Experiment scenarios, prototypes and report - Iteration 1

Massimo Paolucci, Bertrand Souville, Rachid Saadi, Gordon S. Blair, Paul Grace, Trân Huynh, Pierre Châtel

Emergent Connectors for Eternal Software Intensive Networked Systems

ICT FET IP Project

Deliverable D6.2

Experiment scenarios, prototypes and report – Iteration 1

http://www.connect-forever.eu

Project Number	:	231167
Project Title	:	CONNECT – Emergent Connectors for Eternal Software Intensive Networked Systems
Deliverable Type	:	Other

Deliverable Number	:	D6.2
Title of Deliverable	:	Experiment scenarios, prototypes and report – Iteration 1
Nature of Deliverable	:	R/P
Dissemination level	:	PU
Internal Document Number	:	
Contractual Delivery Date	:	M24
Actual Delivery Date	:	22.02.2011
Contributing WPs	:	WP6
Editor(s)	:	Massimo Paolucci
Author(s)	:	Massimo Paolucci & Bertrand Souville (DOCOMO) Rachid Saadi (INRIA) Gordon S. Blair & Paul Grace (LANCS) Trân Huynh (THA) & Pierre Châtel (THALES)
Reviewer(s)	:	

Abstract

The task of WP6 is to evaluate the CONNECT technologies under realistic situations. To achieve this goal, WP6 concentrated its effort in the development of a main scenario in the context of the GMES, which required the connection of two very different and independently build systems provided by the industry partners. The first one is a video-surveillance system provided by Thales; the second one, is an implementation of the GSMA Rich Communication Suite provided by DOCOMO. The resulting scenario allows to verify the validity of some of the CONNECT claims and to investigate with the introduction of some of the CONNECT technologies in the context of the integration of real systems. In addition, WP6 started the work of evaluating how the overall CONNECT work cycle can be introduced in the context of industrial prototype development.

Keyword list

Interoperability, discovery, ontology, real systems

Document History

Version	Type of change	Author(s)
0	Initial TOC and Terrorist Alert UC description	Trân Huynh (THA), Pierre Châtel (THA)
1.0	New TOC	Pierre Châtel (THA)
1.1	Technical challenges	Souville (DCM)
1.2	Discovery using DPWS technical details	Pierre Châtel (THA)
1.3	Introduction + new structure	Paolucci (DCM)
1.4	Extension of new experiments First version of lessons learned	Paolucci (DCM)
1.5	Review of example	Paolucci (DCM)
1.6	a. Moved text of XMPP Messages to appendix b. Moved commander system SOAP interface c. Reformatted the ToC d. extended a bit RCS description.	Paolucci (DCM)
1.7	Corrected Automata and integration section	Souville (DCM)
1.8	Improved discussion of architectural problems and semantics section.	Paolucci(DCM)
1.9	Improved the structure of the document	Paolucci&Souville(DCM)
1.10	Discussion on generation of connector	Paolucci(DCM)
1.11	Conclusion and Future Work	Paolucci(DCM)
1.12	Added introduction to 3.1 on the description of Thales cameras network system. Added various corrections and modifications to 4.1. Added new 4.1.5 chapter on the interaction between WP1 & WP6 through the concept of “affordance”.	Pierre Châtel (THA)
1.13	Added introduction to 4.1 with links to the Discovery Enabler and Affordance usage. Integration of some of the remarks made by Gordon over chapters 3 and 4. Added introduction to the scenario in 2.1. Various minor corrections.	Pierre Châtel (THA)

1.14_THA	Modifications following 04/02/11 audio-conference with DOCOMO and INRIA • Added many more high level details and figures to the description of Thales cameras network system in 3.1 • Updated 4.1.5 to reflect changes from WP1 to the definition of an affordance Suppression of WS4D mentions in 4.1.3 and addition of JMEDS Addition of connectPrim operation in 4.1.4 Modification of WSDL interfaces in Annex 8.1 and addition of every camera interfaces. Added Table of Figures and unified figures numbering	Pierre Châtel (THA), Youssouf Mhoma (THA)
1.15	Fixed the introduction to answer G. Blair comments	M. Paolucci (DOCOMO)
1.16	Extensive Rewriting up to Section 2.2	M. Paolucci (DOCOMO)
1.17	Included Bertrand's comments and fixes. Specifically fixed references and bibliography Moved some huge WSDLs in appendix	B. Souville (DOCOMO)
1.18	Extensive rewriting to address Valerie's and Gordon's comments	M. Paolucci (DOCOMO)
1.19	CONNECT references	R. Saadi(INRIA)

Document Review

Date	Version	Reviewer	Comment
2.2.2011	1.11	G. Blair (LANCS)	Needs to elaborate on the experimental framework.

Table of Contents

Table of Figures

1 Introduction

The scope of WP6, as reported from the Description of Work, is the following: "The objective of WP6 is to assess the CONNECT architecture and prototypes as generated by WP1 against *actual* scenarios". To this extent, the work performed within WP6 has concentrated on constructing such scenarios by identifying real systems that are available to the project members and that can provide a testbed against which to experiment with CONNECT technologies, defining in this way the *actual* scenarios that are expected in the Description of Work.

The starting point of the work performed has been the following:

1. Identify two systems that are available to the industry partners that were (1) independently developed, (2) industry strength, and that (3) could provide an initial vehicle toward the transition of the technology.
2. Given the two systems, WP6 identified a scenario in which the two systems could be in principle used, and identified how the two systems could interoperate in a real situation.
3. Finally WP6 proceeded to identify how CONNECT technologies fit in the scenario and it initiated to integrate the technology starting from the systems discovery.

In the process, WP6 identified problems that were not addressed by other WPs specifically with respect to the architectural constraints that derive from the use of CONNECTors. These architectural constraints have been fed back to WP1 and they are reflected in deliverable D1.2 (P. Grace, 2011).

1.1 The Role of Work Package WP6

The objective of this workpackage is to practically assess the integrated CONNECT prototype implementation of WP1 based on actual scenarios.

Task 6.1 that aims at defining, implementing and experimenting with a collection of scenarios that are intended to help the project along the research process.

Task 6.2 which will concentrate on the assessment in the systems of systems area.

Task 6.3 that involves a technical and industrial evaluation of the proposed solutions

1.2 Summary of Achievements

1. Identification of the systems that will form the evaluation platform
2. Definition of the evaluation scenario
3. Initial implementation of the evaluation scenario
4. Initial integration of CONNECT technologies in the implementation
5. Feedback to our partners on the obstacles encountered during the initial implementation

1.3 First Review Recommendations

Last year's review suggested two goals for the work to be performed this year within WP6.

1. Select one scenario that could work across the whole project
2. Pay particular attention to non-functional properties that were neglected by the scenarios proposed last year.

The core of the work in WP6 concentrated around the first goal, with the objective of defining a scenario that would be challenging for all WPs. Furthermore, the scenario selected will develop further next year towards a fully integrated GMES scenario.

Although no particular attention was devoted to non-functional properties over other aspects of the project, their importance emerged from the scenario, and indeed, their use is crucial as discussed more in details in Section 3.2.2.

1.4 Challenges for Year 2

The main challenge for Year 2 has been the definition of the evaluation platform, which required the definition of an evaluation scenario that satisfied a number of conflicting constraints. First and foremost the scenario had to be realistic, this means that it needed to involve complex systems that are independent of CONNECT and that may need to interoperate under some conditions. Second, the scenario needs to highlight challenging interoperability constraints at different level of abstraction: from the low level messaging, to protocol level interoperability, to high-level architectural constraints. Third, while complex, the scenario needs to be implementable and it should offer the possibility to plug in the results of the other work packages. In the process, the goal of this year's work has been to identify problems in the current vision of connectors and provide feedbacks to the other work-packages.

1.5 Achievements in Year 2

The main goal of Year 2 has been moving form explorative and hypothetical cases as the "popcorn scenario" described in (Huynh, et al., 2010) to realistic cases that challenge the CONNECT technology. Along these lines, the main achievements of WP6 in Year 2 have been the following.

1. The definition of the scenario that satisfies the requirements highlighted above. It is realistic in the sense that it exploits systems that are available to the industry partners to develop new services that their companies may commercialize. Furthermore, the scenario highlights the interoperability challenges that are of interest of CONNECT. Among them, there is a protocol level mismatch between XMPP and SOAP; there are application level protocol mismatches between the different applications used; at the architecture/deployment level there are problems to be addressed.
2. WP6 moved toward an initial implementation of the scenario with the connection between the two systems.
3. In addition to the scenario above, CONNECT ideas have been used in a completely unrelated scenario with the goal of interfacing two systems that were available to one of the industry partners for pre-commercialization. This work, which is briefly discussed in section 1, highlighted problems and new issues that need where not directly raised in the experimental scenario.
4. Task 6.2, on the assessment of the evaluation in CONNECT, has been kicked off.

1.6 Structure of the Deliverable

The rest of the deliverable is organized as follows. In section 2 we will highlight the experimental scenario and the two systems that are to be integrated to implement the scenario; in section 3 we will address the technical challenges and describe the connected system. In section 1 we will describe an additional experiment that has been conducted as an initial exploitation of CONNECT; and finally in section 5 we will elaborate on the lessons learned and ways to address the problems; and conclude highlighting the challenges ahead.

2 The Terrorist Alert Scenario

The goal of WP6 is to evaluate the CONNECT technology with respect to actual scenarios, or in other words, with respect to conditions that are as close as possible to the conditions in which CONNECT technology would be deployed. The approach followed by WP6 is necessarily bottom up: starting from systems that are defined outside the project, developed a challenging scenario in which these systems are expected to work together, to verify whether and how the CONNECT technology eases integration and achieves run-time interoperation. Following this approach it is possible to identify when the CONNECT technology fails to address (some of) the problems that occur during system development, and to identify additional requirements that are not addressed by CONNECT, but crucial for the deployment of CONNECTed systems.

The search for the scenario was driven by the two industrial partners and hinged on the technologies that they could provide to the project and the products and potential transfer that they could do with the project results. Thales has a long standing business in security systems, and to this extent one of the product lines at Thales is the development of surveillance systems that can be deployed for crowd control in environments such as stadiums. DOCOMO's core business is to provide mobile network. To this extent, DOCOMO is interested in systems that enhance the intelligence of the network, and in particular in systems that integrate multi-media capabilities in the network. Furthermore, DOCOMO is interested in novel uses of the network that exploit the increasing bandwidth and transmission speed that the network provides. Video surveillance is one such application, which supports a number of interesting use cases including remote monitoring and rescue operations during natural disasters. Given these two concerns, it has been natural to converge on the *Terrorist Alert* use case that is described more in details in section 2.1.

The systems involved are also provided by the two industrial partners, and they were developed and deployed completely independently of the project. To this extent, Thales contributed the video-surveillance system, which is based on state of the art Web service technology. The system is deployed here over DPWS (Device Profile for Web Services) (Driscoll, et al., 2009), which is a standard that supports secure and decentralized web service discovery, to provide remote control of the cameras as well as video delivery. DOCOMO contributed the access to a industry grade test-bed implementing the Rich Communication Suite (hereafter RCS) that is under standardization at GSMA (GSMA, 2010) (GSMA-FD, 2010) The advantage of using RCS is twofold. First, it provides a piece of software that has been implemented by a third party and therefore cannot be modified to facilitate the CONNECT project, rather it forces CONNECT technology to create a viable connector. Second, RCS provides a blueprint for the deployment of advanced services on the mobile network, and therefore it provides a venue toward the exploitation of the CONNECT technology.

In addition to the systems, the scenario of course comes with a storyboard that explains the conditions under which the two systems interact. The storyboard of the scenario was developed starting from the business considerations and the systems provided. The aim while developing the storyboard was to describe a hypothetical situation in which the two systems could work together to form a system that achieves goals that are beyond what can be achieved by each one of the systems independently. Since Thales and DOCOMO do not have a history of working together, the storyboard was invented ad hoc for the project and therefore it is inevitably somewhat artificial; but it is also rich enough to allow us to experiment with different CONNECT technologies, and to move towards a more complex GMES system to be constructed in the remaining time of the project.

Purposely, the scenario was build around the general problem addressed by CONNECT, but not around the CONNECT architecture or any of the CONNECT enablers. There are two reasons for that.

1. Concretely evaluate the ideas that were developed within the CONNECT project against a one specific complex, and as real as possible, scenario. With the goal

to analyze how the different enablers interact to address the problems in the scenario.

2. Provide a stepping-stone towards a general approach to the evaluation of CONNECTed systems.

The rest of the deliverable will concentrate on the first reason, namely the concrete evaluation, with the *experimental hypothesis* that all the enablers described in the architecture proposed in D1.2 are required to construct the CONNECTor required to address the scenario. This experimental hypothesis assumes three violations:

1. some of the enablers in the CONNECT architecture will not be required at all;
2. some enablers will have to be used in ways that are different than the use foreseen by the general CONNECT architecture;
3. the scenario requires additional requirements that are not satisfied by CONNECT.

In such cases we will have to register a deviation from the experimental hypothesis. Such undesired results will anyway provide important information on the impact of CONNECT in solving interoperability problems and indeed it starts posing questions of when and how CONNECT technology may be utilized,

This section is organized in the following way. First, in section 2.1, we will introduce the scenario to provide the background of the evaluation; then in section 2.2, we will briefly introduce the two systems; finally in section 2.3, we will review the technical challenges that are involved in the scenario.

2.1 Storyboard Description

The storyboard is centered around the search of a suspected terrorist in a crowded stadium where video surveillance is running during a show. It describes all the actions that took place from the moment a terrorist alert is detected by the stadium security center, to its arrest thanks to an efficient exchange of information (alerts, video data, coordinates, etc.) between all the involved parties. As such, it includes:

- **3+ locations**: stadium, police control center, outside stadium and other zones of interest (ZOI)
- **8 parties**: operators of the stadium security center, suspect, stadium guards, mobile law enforcers, private security guards, police captain, police/private chiefs
- **4 steps**, detailed in the sections below:
    1. receiving alert (see 2.1.1) which acts as the opening scene setting
    2. dispatching alert (see 2.1.2) where the CONNECT discovery is actively used
    3. video surveillance (see 2.1.3) where a CONNECTor is constructed to connect the two systems to connect
    4. capturing the suspect (see 0) which acts as closing step.

2.1.1 Step 1: Receive Alert

The Terrorist Alert scenario starts in a stadium. People are enjoying the show and the operators of the stadium security center are on the alert for any problems that could happen in a mob such as hooligans or health emergencies.

The show is running smoothly when suddenly an operator from the security center spots an individual carrying a weapon wandering suspiciously in the corridors of the stadium, and triggers an alert (Figure 1).

This triggers an alert event. The security guards located in the stadium organize the capture of the suspect while the snapshot of the suspect is immediately sent to the assigned police control center.

The suspect notices the threat of being captured coming and manages to evade out of the stadium.

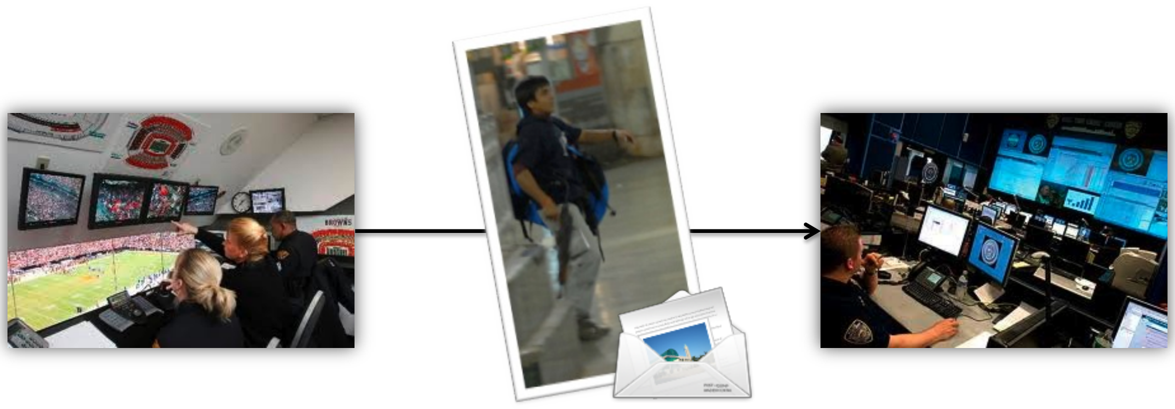


Figure 1: Receiving an alert (Step1)

2.1.2 Step 2: Dispatch Alert

On reception of the snapshot and once the threat is acknowledged by the captain, the police control center:

1. sets a zone of interest (ZOI) around the stadium (Figure 2),
2. dispatches the alert as well as the snapshot of the suspect to the mobile law enforcers within the ZOI,
3. also dispatches the information to all the private security guards within the ZOI because of the high threat level.

The private security guards report to their respective chiefs while the latter reports to the assigned chief officers.

The ZOI is now being put under strict surveillance.

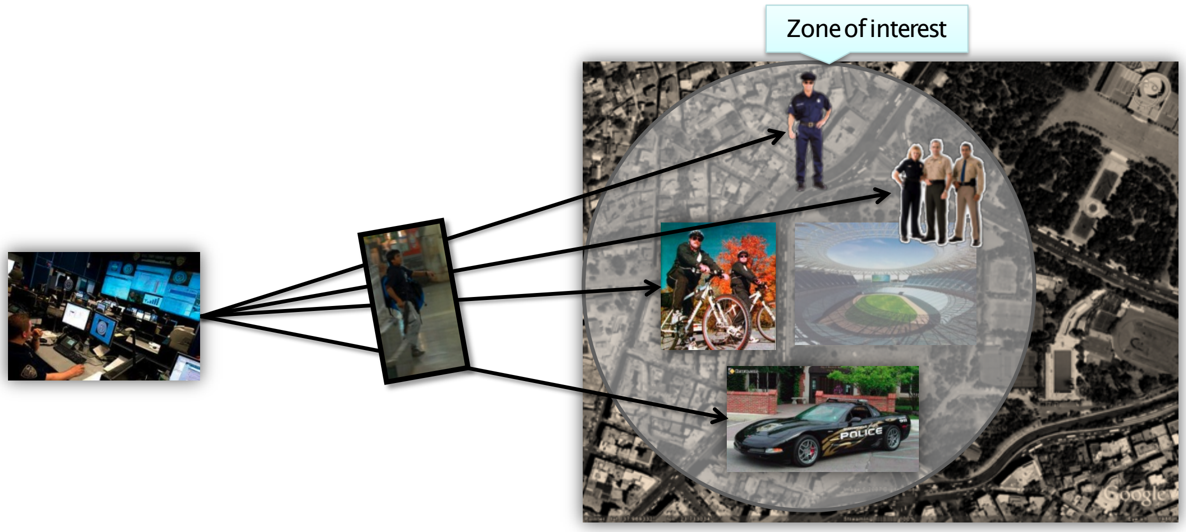


Figure 2: Dispatch alert (Step2)

After some time, notifications of individuals looking alike the suspect from different locations within the ZOI are being relayed to the police control center.

The captain decides to modify the zone of interest accordingly to the updated information about the suspect. There are now two smaller ZOIs.

From the technical point of view, Step 2 requires the discovery of cameras that corresponds to the ZOIs, and potentially the discovery of other devices that allow the captain to monitor a given Zones of Interest.

2.1.3 Step 3: Video Surveillance

In addition to the mobile law enforcers and the private security guards, the captain commands the operators at the control center to put the ZOIs under video surveillance (Figure 3).

To increase the chances of finding the suspect, the police will also keep watch: on the videos from the public surveillance cameras within the ZOIs, and on additional videos collected through the mobile operators RCS system (see 2.2.2). Potentially, other sources of information could be exploited such as social networks, video sharing sites, etc.

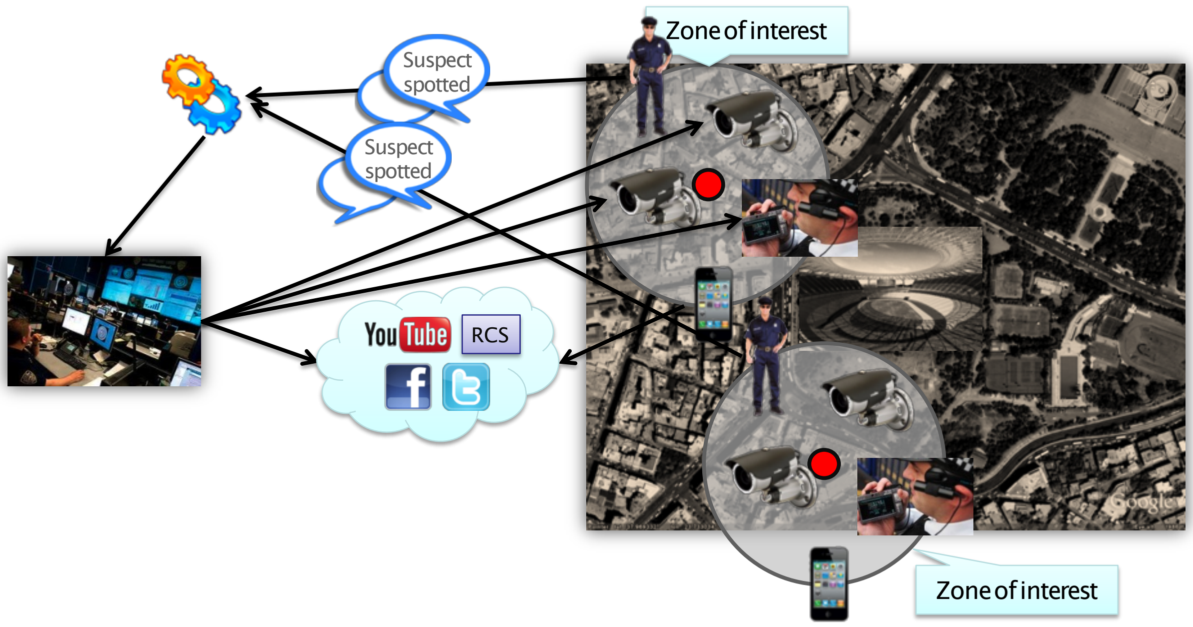


Figure 3: Keep watch on the zones of interest (Step 3)

Technically, step 3 is responsible to interface the commander interface with the video distribution within RCS. Since the command center has been constructed to work with the cameras independently of RCS, this step would require some integration effort. The objective here is to replace such work with the construction of a CONNECTor. Essentially, on this step the control center acts as a client of the camera while RCS acts as a "degenerate camera".

2.1.4 Step 4: Suspect Capture

The tight surveillance of the ZOIs has been a success. The suspect has been clearly indentified and his capture been ordered by the captain (Figure 4).

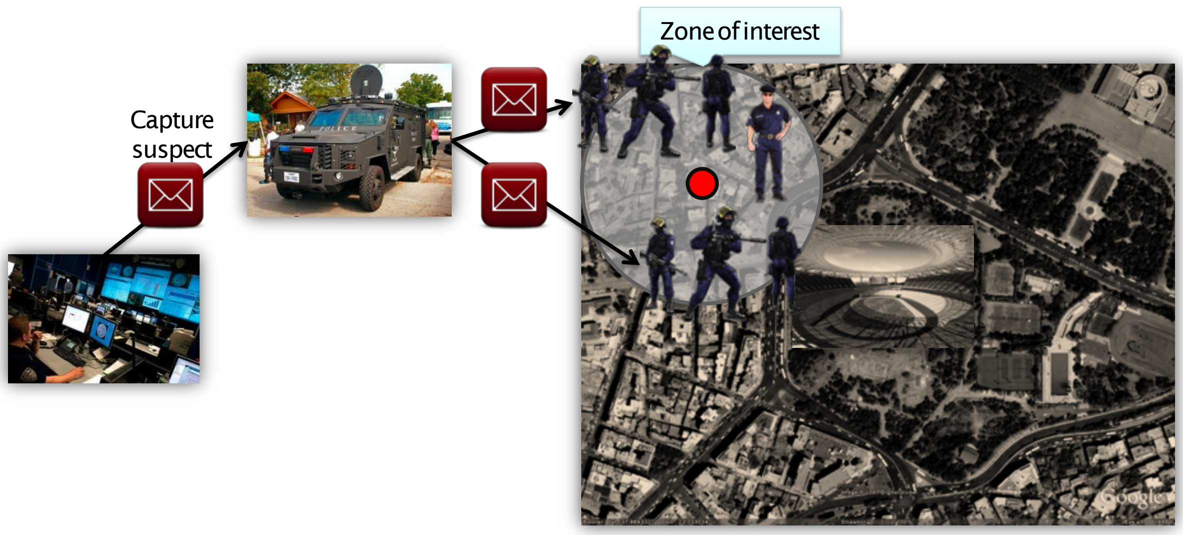


Figure 4: Suspect capture (Step 4)

Overall, as discussed more in details in section 3, the scenario allows us to explore two aspects of the CONNECT technology: the discovery mechanism and the construction of the connector. In addition, the overall setup allows us to investigate other issues in CONNECT. Specifically, with respect to the work in WP5 on reliability, the construction of a connector allows the experimentation with different models of dependability, performance, security and trust.

2.2 Description of Involved Systems

As discussed in the introduction, one of the main features of the work performed by WP6 has been to exploit existing systems that are available at the two companies that are partners in the project. In this section, we briefly review both of them.

2.2.1 Description of Thales Cameras Network System

One of the product lines at Thales is the development of surveillance systems that can be deployed in a broad range of applications: from crowd control in dynamic environments and open spaces, to more classical “closed” contexts such as public or private buildings and their direct surroundings. The stadium use case described above fits in the first category due to the extent of its video surveillance coverage that goes well over the stadium itself.

From the start, this system was designed with this “broad spectrum” goal in mind: design choices were made to accommodate multiple deployment environment and scenarios: the Terrorist Alert use case being one specific instantiation that was defined for CONNECT. For instance, the cameras network system is currently being used for indoor video surveillance and movement detection, combined with seismic detection (see Figure 5).

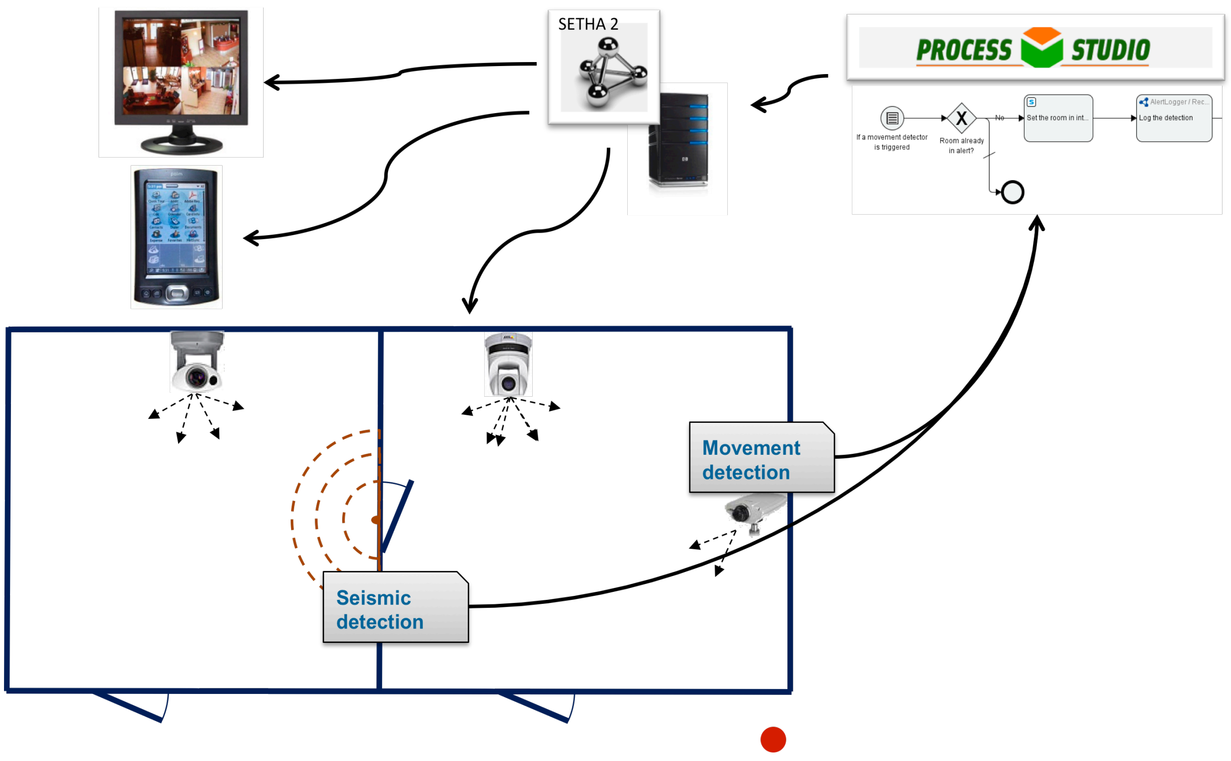


Figure 5: Another scenario for the cameras network system

This surveillance system is currently deployed at Thales, the Thales SETHA 2 semantic service framework being its key part. This innovative framework integrates all features and software components needed to deal with semantically annotated[1] (Web) services:

- Support of semantically annotated business **process modeling** (using BPEL) and **execution** (based on specific BPEL engine)
- Support of **semantic services registration** (using a specifically developed registry) and **execution**. Services are described externally through semantically annotated WSDL files.
- Support of **message exchanges** between services consumers and producers, through a dedicated bus (the ServiceMix ESB)
- Support of dynamic **data adaption** to deal with discrepancies between semantic service consumers and producers.

These components are organized thereafter in SETHA 2 architecture (see Figure 6):

[1] Semantic annotations are based on available business and technical OWL ontologies

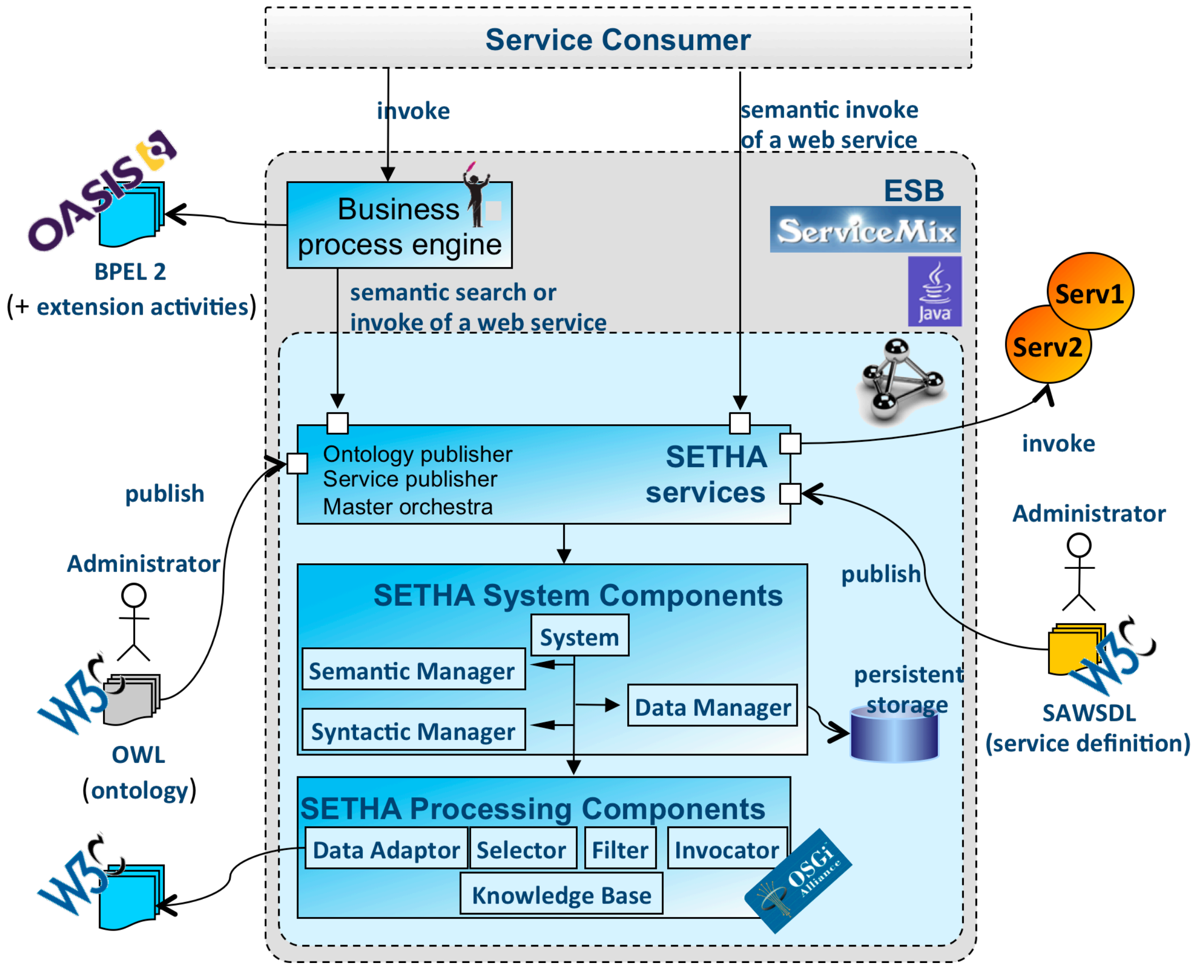


Figure 6: SETHA 2 architecture for camera support at Thales

In CONNECT, we try to implement a new unobtrusive method of communication between network actors that may, in many ways, supplants traditional ESBs, like the one currently used in SETHA 2. This is one of the many reasons why it is interesting for the consortium to try to deploy this cameras network system in this new context and see how integration problems can be solved with CONNECT technology.

2.2.2 Rich Communication Suite

The Rich Communication Suite (RCS) (GSMA, 2010) (GSMA-FD, 2010)[2] is an industrial initiative within GSMA to create new services on top of IMS/IP networks (3GPP, 2010). IMS (IP Multimedia Subsystem) is a set of 3GPP standards that specify the details of the telco operators' infrastructure. Essentially, IMS provides all capabilities that support the construction of a communication channel across two or more phones using the IP network. Such capabilities include user authentication, routing of SIP messages, charging, 99.999% up-time reliability, and other functions that are essential to run the telephone system.

RCS defines a layer of added value services, which allows customers to gain more value from their phones and operators and other parties to increase their revenues. At this time RCS standardized the following set of service enablers.

[2] On 15 February 2011, during the World Mobile Congress, five major European Operators announced the adoption of RCS within the end of 2011.

http://www.mobileeurope.co.uk/news/news-anaylsis/8570-mwc-big-five-operators-commit-to-qnewq-rcs-approach

- **Presence:** to allow the user to specify whether she is available to take the call
- **Voice call:** VOIP standard
- **IM:** Messaging based on the SIP protocol
- **Video Share:** Sharing of videos
- **Image Share:** Sharing of images

One crucial aspect of the RCS standards is that the service enablers can be combined in new ways to create new services. The three minimal services that are envisioned by the RCS standardization effort are the following (GSMA-FD, 2010).

- **Enhanced Address Book:** mobile phonebook enhanced with contact presence and status
- **Rich Call:** mobile voice calls enriched with multimedia content such as image files and video
- **Rich Messaging:** mobile messaging enhanced through a conversational experience

The implicit value of RCS is that it provides a blueprint for a new layer of service provisioning on top of the telephone network. This layer has been always implied by IMS and intelligent networks before, but never clearly specified. While it is not the task of RCS to define such layer, RCS provides a "template" for the creation of other services.

As a consequence, all RCS implementations include a number of non-standard extensions that allow creating and deploying services that are beyond the strict RCS standard. One of the most important of such extensions is a number of gateways toward legacy systems and Internet services. These extensions make all RCS implementations somewhat idiosyncratic, and mostly serve the vendors to distinguish their offers and to customize their offer toward the needs of their customers.

3 Addressing the Challenges of the Scenario

In this section we discuss more deeply the work done toward the automatic construction of CONNECTors. Specifically, we will concentrate on two orthogonal issues: discovery and constructing the CONNECTor itself. Specifically, in section 3.1 we will discuss discovery in the Thales system; such system has been initially defined on the bases of Devices Profile for Web Services (DPWS) (Driscoll, et al., 2009), and in the curse of the project extended to include the notion of affordance discussed in D1.2. In section 3.2, we report on the progress toward the construction of the CONNECTor. Finally, in section 3.3 we will relate this experiment with the use of enablers in the CONNECT Architecture described in D1.2.

3.1 Discovery of the Cameras

The first issue to address in the scenario is to define a technical mean to discover cameras. Indeed, as has previously been described in 2.2, it is a crucial part of second step of the scenario. Due to its versatile nature, the “business” context of the discovery does not allow the prior “offline” identification of most required cameras before the launch of the system: for instance, this dynamicity can be seen in mobile cameras attached to policemen that can, or cannot, be available in our zones of interest at the time of a specific alert.

This versatility, a preeminent part of Thales’s cameras network system, as well as its agility in pervasive environments, is here implemented through Devices Profile for Web Services (DPWS). As explained below in sections 3.1.1 to 3.1.3.

As a matter of fact, since CONNECT deals with interoperability and transcending different service discovery protocols, other discovery protocols could be used. But, for the purposes of this experiment, we are interested in DPWS since it offers legacy support of existing Web services (which do exist for cameras in Thales cameras network system) and distributed service discovery features (through “broadcast” discovery requests, as opposed to centralized via an all-inclusive registry).

Discovery of the cameras by way of DPWS in this experiment is also linked to the *Discovery Enabler* component developed in WP1, and more specifically to the concept of *Affordance* that encompasses the annotated XML schema of a service data, as well as its annotated WSDL interface, process model and non-functional properties. How the affordance is leveraged in the scenario and Thales cameras network system is explained in 3.1.5.

3.1.1 Devices Profile for Web Services

With a purpose similar to the one of the Universal Plug and Play (UPnP) specification, Devices Profile for Web Services (DPWS) defines implementation constraints to enable secure and *decentralized Web Service discovery*, but also messaging, description and eventing (subscribing to, and receiving events from a Web Service), while keeping in mind the challenge of embedded services on resource-constrained devices. DPWS has also been called Web Services on Devices (WSD).

DPWS provides the following functionality between conforming devices:

- Discovering DPWS-capable devices on the network and the services they offer;
- Sending messages to DPWS-capable devices and receiving replies;
- Describing a Web service by providing a WSDL file;
- Interacting with a service using its description;
- Subscribing to and receiving events from a Web service.

As a standard, DPWS 1.1 was submitted to OASIS in July 2008 and approved, together with WS-Discovery 1.1, on June 2009. In fact, DPWS has been developed mainly by Microsoft and

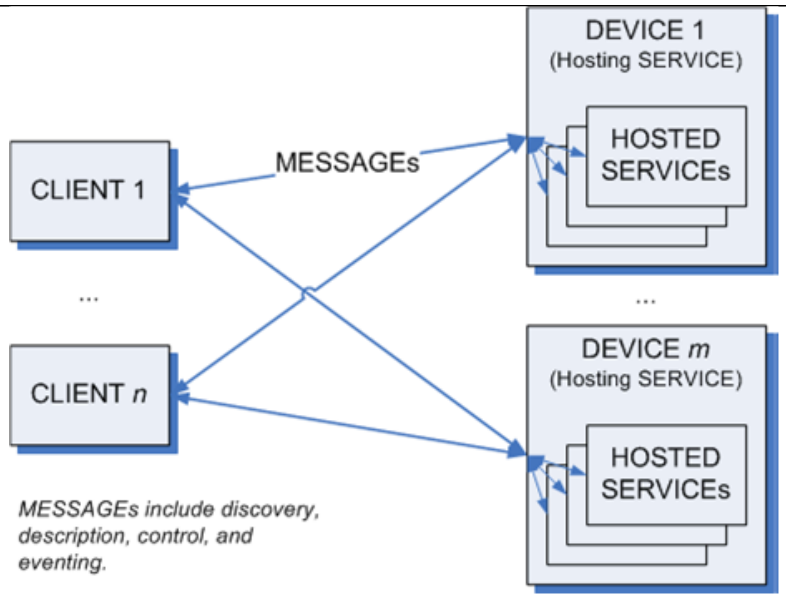


Figure 7: DPWS, Arrangement of clients and devices

some printer device manufacturers. Indeed, at the moment, DPWS is part of the "Network-connected Devices" program of Microsoft and integrated in their latest operating system with a stack called WSDAPI. Microsoft uses DPWS to ease the integration of networked hardware like printers or beamers into Windows.

As a profile, DPWS defines a set of guidelines for how to use Web Services technologies. Indeed, Web Services standards in general allow implementers to choose from a variety of message representations, text encodings, transport protocols, and other options, some of which are not interoperable. By constraining these decisions, DPWS ensure that conforming implementations will work well together. As such, it is based on several other Web Services specifications:

- WS-Addressing for advanced endpoint and message addressing;
- WS-Policy for policy exchange;
- WS-Security for managing security;
- WS-Discovery for device discovery. It is a technical specification that defines a multicast discovery protocol to locate services on a local network. As the name suggests, the actual communication between nodes is done using web services standards, notably SOAP-over-UDP;
- WS-Transfer / WS- MetadataExchange for device and service description;
- WS-Eventing for managing subscriptions for event channels.

DPWS is also partially based on W3Cs Web Services Architecture, which includes SOAP, WSDL and XML-Schema.

As a matter of fact, one of the main characteristics of DPWS, which is very useful in the context of the WS Camera scenario of CONNECT, is that it has been designed from the ground up with Web Services integration in mind, and also includes numerous extension points allowing for seamless integration of device-provided services in enterprise-wide application scenarios.

With DPWS, devices available on a network can run two types of services: **hosting services** and **hosted services**, which are accessed through messages by the clients (see Figure 7).

Hosting services are directly associated to a device, and play an important part in the device discovery process. Hosted services, on the other hand, are mostly functional sub-entities that are discovered through their hosting device.

In addition to user-defined services, a typical DPWS infrastructure also includes a set of built-in services:

- **Discovery services**: used by a device connected to a network to advertise itself and to discover other devices;
- **Metadata exchange services**: provide dynamic access to a device's hosted services and to their metadata;
- **Publish/subscribe eventing services**: allowing other devices to subscribe to asynchronous event messages produced by a given service.

Figure 8 looks into the details of the typical architecture of a DPWS-compliant (physical) device and sums up some of the aforementioned concepts: user-defined services and events are shown in yellow, predefined services in green and, finally, the two network interfaces (*primary* over HTTP and *discovery* using UDP and multicast address) are displayed in the bottom.

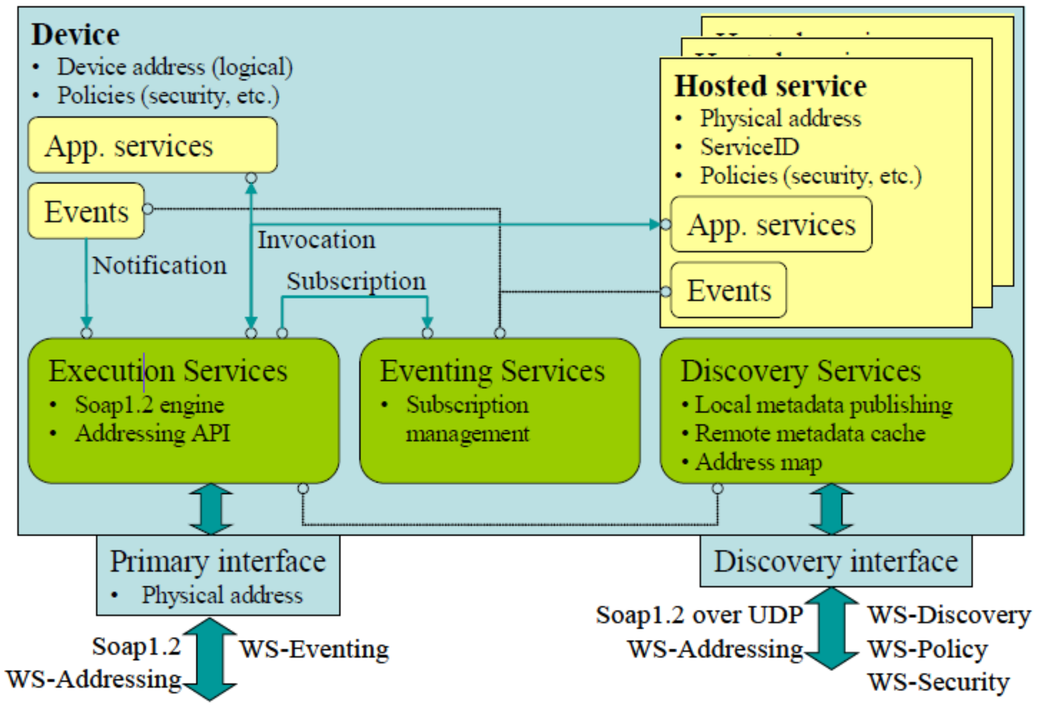


Figure 8: A DPWS-compliant device architecture

3.1.2 DPWS Implementations

Multiple DPWS implementations (or stacks) exist in both C and Java. The following is given as a non-exhaustive list:

- Java MultiEdition DPWS-Stack (JMEDS)[3];
- Life|ware's DPWS Stack for domotics[4];

[3] http://sourceforge.net/projects/ws4d-javame

[4] http://www.life-ware.com/products/dpws.php

- SOA4D - Service-Oriented Architecture for Devices[5], Open source and free-of-charge implementations in C and Java of an embeddable DPWS stack. It has originally been implemented as part of the SIRENA project, under the auspices of the European research initiative ITEA. For instance, the The DPWS4J Core project[6] provides a Java Web Services stack for the J2ME CDC platform, compliant with the Device Profile for Web Services (DPWS) specification;
- WS4D - Web Services for Devices[7], an Open source Java/OSGi and C implementations of DPWS whose roots can also be traced dows to the SIRENA project.

3.1.3 Discovering a Camera

DPWS uses the WS-Discovery protocol for plug-and-play device discovery. WS-Discovery defines a multicast discovery protocol to search for and locate network-connected resources. The primary mode of discovery is a client searching for one or more Hosting services (see Figure 7). In the context of DPWS, these are (physical) devices. Hosted services do not participate in the discovery process, but can be individually addressed (through their respective End Point References (hereafter EPRs) once the hosting device has been discovered.

A DPWS discovery request can either specify the *type* of a device or a *scope* in which the device resides or both. As such, during the discovery process, a device exposes the following metadata:

- Its EPR, which allows to determine the device's physical network address;
- Types: a set of messages the device can send and/or receive; these can be either functional WSDL port-types, or **abstract types (as in the following DPWS example)** grouping several port types and/or hosted services
- Scopes: a set of attributes that may be used to organize devices into logical or hierarchical groups, e.g. according to their location or access rights;

Note: we use JMEDS Java implementation for connecting camera into the DPWS network.

3.1.4 Interacting with a Camera Web Service

After its discovery, interaction with a camera is based on its interface. In this network system, Web Services Description Language (WSDL) v1.1[8] has been chosen for the definition of service interfaces.

The WSDL interfaces of every camera are given in Annex 1 in section 7.1. The following extract focuses on the business operations that have been defined for interaction on PTZ (*Point*, *Tilt*, *Zoom* capabilities) cameras, leaving aside the more technical details.

Camera214 WSDL interface extract:

```
<portType name="Camera214Impl">
            <operation name="goToPresetPosition">
                  <input message="tns:goToPresetPosition" />
                  <output message="tns:goToPresetPositionResponse" />
                  <fault message="tns:CameraException" name="CameraException" />
            </operation>
```

[5] https://forge.soa4d.org/

[6] https://forge.soa4d.org/projects/dpws4j/

[7] http://www.ws4d.org/

[8] http://www.w3.org/TR/wsdl

```
              <operation name="setZoomFactor">
                     <input message="tns:setZoomFactor" />
                     <output message="tns:setZoomFactorResponse" />
                     <fault message="tns:CameraException" name="CameraException" />
              </operation>
              <operation name="getVideo">
                     <input message="tns:getVideo" />
                     <output message="tns:getVideoResponse" />
                     <fault message="tns:CameraException" name="CameraException" />
              </operation>
              <operation name="connectPrim">
                     <input message="tns:connectInput" />
                     <output message="tns:fileString" />
                     <fault message="tns:IOException" name="connectException" />
              </operation>
</portType>

(…)

<xs:complexType name="getVideo">
       <xs:sequence>
              <xs:element name="resolution" type="xs:string" minOccurs="0"/>
              <xs:element name="duration" type="xs:int" minOccurs="0"/>
       </xs:sequence>
</xs:complexType>

<xs:complexType name="getVideoResponse">
       <xs:sequence>
              <xs:element name="video" type="xs:any"/>
       </xs:sequence>
</xs:complexType>
```

Since we consider cameras with pan, tilt and zoom capabilities, the aforementioned `goToPresetPosition` operation is used to move the camera over a predefined position on its axes. The operation takes a `string` as its unique parameter that should match the name of any position that has been predefined using the administration interface of the camera. As such, these positions can be tailored to any business scenario. The operation doesn't have any return parameter.

The `setZoomFactor` operation is used to change the zoom factor of the camera. It uses an integer ranging from 0 to 10 as parameter that matches the camera min and max zoom capabilities (from large angle to telephoto). The operation doesn't have any return parameter.

The getVideo operation allows obtaining a video file from the camera by its return parameter. The main characteristics of this video file will be defined by the operation input parameters, as represented, in the previous interface extract, by the `getVideo` XSD complex type: the resolution of the video (in a *AxB* format, as a string), its duration (in ms, as an integer).

Finally, each aforementioned operation can trigger a `CameraException` in case of invalid argument or if something goes wrong during its execution.

In addition to PTZ cameras, fixed cameras will also be used. The only difference between them is that fixed cameras miss the `goToPresetPosition` operation.

3.1.5 Using CONNECT Affordance-based Discovery Mechanism

WP1 defined the concept of *Affordance* for describing available services features looked-for during the discovery phase. As the cornerstone of the CONNECT Networked System Model, an affordance is a macroscopic view, or the quality of a feature, of a networked system. Essentially, the affordance describes the high-level roles a networked system plays, e.g., 'prints a document', or 'sends and e-mail'. This allows semantically equivalent action-relationships/interactions with another networked system to be matched.

An example of the affordance is provided below. It shows that the macroscopic capability of the camera is to retrieve videos (as defined by http://www.connect.com/ontology/media#RetrieveVideo) and that their outputs is a Video (as defined by http://www.connect.com/ontology/media#Video).

Definition of the cameras affordance:

```
<?xml version="1.0" encoding="UTF-8"?>
<Affordances name="DPWSCamera" >
  <Affordance name="Camera" kind="provided">
      <FunctionalConcept>
       http://www.connect.com/ontology/media#RetrieveVideo
      </FunctionalConcept>
      <Inputs>
        <Input>owl:Nothing</Input>
      <Inputs>
      <Outputs>
        <Output>
           http://www.connect.com/ontology/media#Video
        </Output>
      <Outputs>
  </Affordance>
</Affordances>
```

The affordance defined above will be used in this scenario since we integrated both DOCOMO's RCS system and Thales's Cameras Network System together, thanks to this matching technique. For cameras, depending on the granularity of the semantic modeling, affordances could reflect their ability to *pan*, *tilt* and *zoom* at a lower level; or simply to provide a video feed/file from a higher level point-of-view.

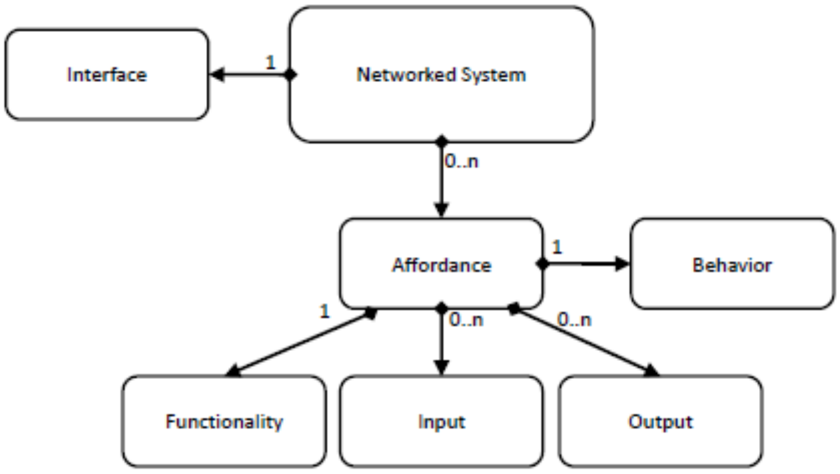


Figure 9: CONNECT Networked System Model

Moreover, as shown in Figure 9, the Networked System Model also defines other service meta-data related to the affordance that can be used for further service matchmaking and execution support: its *Behavior*, *Functionality*, *Input* and *Output*. This includes the annotated XML schema of a service data, as well as its annotated WSDL interface, process model and non-functional properties.

The *Behavior* is the description of the valid "call sequences" of operations over these services. These sequences are modeled in the BPEL business process language, from the point-of-view of the service provider. As such, they differ from a classical BPEL file, which usually describes the required operation calls and sequence flow needed by a service consumer.

In the CONNECT framework, a *Discovery Enabler* component has been implemented and includes an affordance repository and matching engine. It will be used for the deployment of this scenario. Furthermore, for the implementation of the affordance-based discovery mechanism by services, a dedicated *connectPrim* operation may be implemented, as an alternative to learning, by each system available in the network. Its role is to provide a standard meta-data retrieving point for every service consumer to use, as shown in Figure 10:

1. First of all, a DPWS compliant service discovery is made in order to discover available (camera) service, whether physical, from the Cameras Network System, or “virtual” (a data proxy), from RCS.
2. Then, since available services implement the connectPrim operation, services consumers and the CONNECT framework, can retrieve and exploit the Networked System Model information provided by services.
3. Finally, data exchanges can freely happen between consumers and producers thanks to the generated CONNECTors.

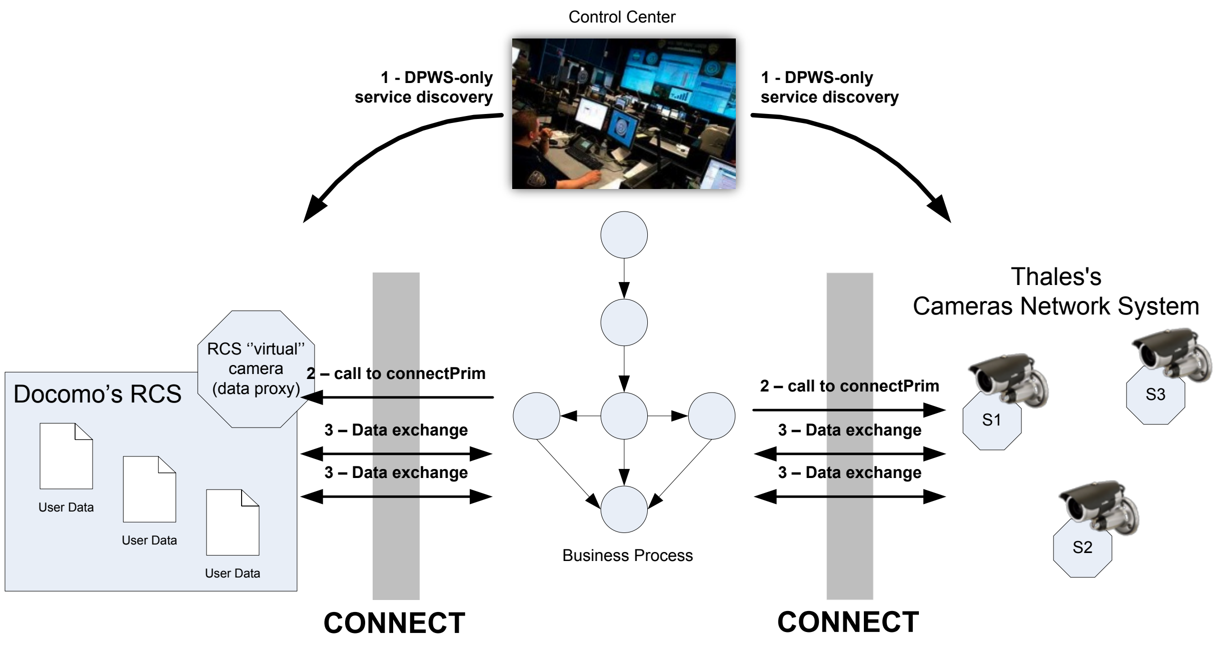


Figure 10: Use of connectPrim to retrieve services meta-data

The signature of the *connectPrim* operation and associated data types is described bellow and should be added to every camera WSDL interfaces, as well as their service implementations.

Definition of the *connectPrim* service operation:

```
<xs:element name="connectInput">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="GetAffordance"/>
      <xs:enumeration value="GetSemInterface"/>
      <xs:enumeration value="GetBehaviour"/>
      <xs:enumeration value="GetNFP"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>


<xs:element name="fileString" type:xsd:string/>

<xs:element name="IOException" type:xsd:string/>

<operation name="connectPrim">
      <input message="tns:connectInput" />
      <output message="tns:fileString" />
      <fault message="tns:IOException" name="connectException" />
</operation>
```

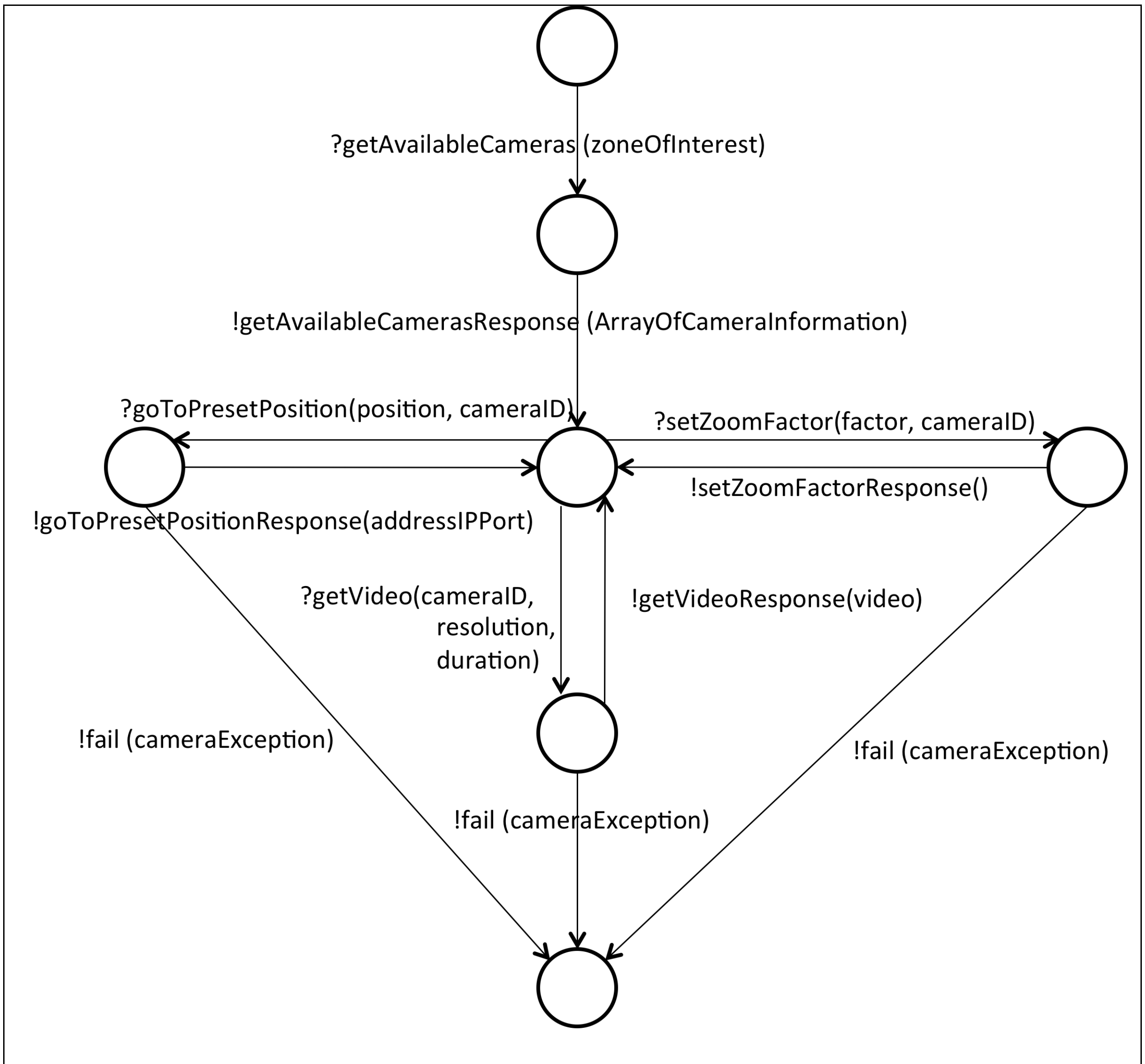

?getAvailableCameras (zoneOfInterest)
!getAvailableCamerasResponse (ArrayOfCameraInformation)
?goToPresetPosition(position, cameraID)
?setZoomFactor(factor, cameraID)
!setZoomFactorResponse()
!goToPresetPositionResponse(addressIPPort)
?getVideo(cameraID, resolution, duration)
!getVideoResponse(video)
!fail (cameraException)
!fail (cameraException)
!fail (cameraException)

Figure 11: Interaction behavior model of the SOAP Interface

It should be noted that for each parameter (*GetAffordance*, *GetSemInterface*, *GetBehaviour*, *GetNFP*), an associated data is returned. We are here mostly concerned by the *GetAffordance* parameter that returns the affordance as a whole, but it should be noted that other types of data that are included in the concept of affordance can be directly retrieved, such as the non-functional properties (NFP) of the service.

3.2 Towards the Construction of CONNECTors

After discovery is completed, the next problem to address is the construction of the CONNECTors so that the commander at the control center can extract additional videos that have been stored in the RCS video repository. Specifically, in section 3.2.1 we will concentrate on the detailed description of the different components from which to construct the CONNECTor; and in section 3.2.2 we will analyze additional problems that emerge during the construction of the CONNECTor.

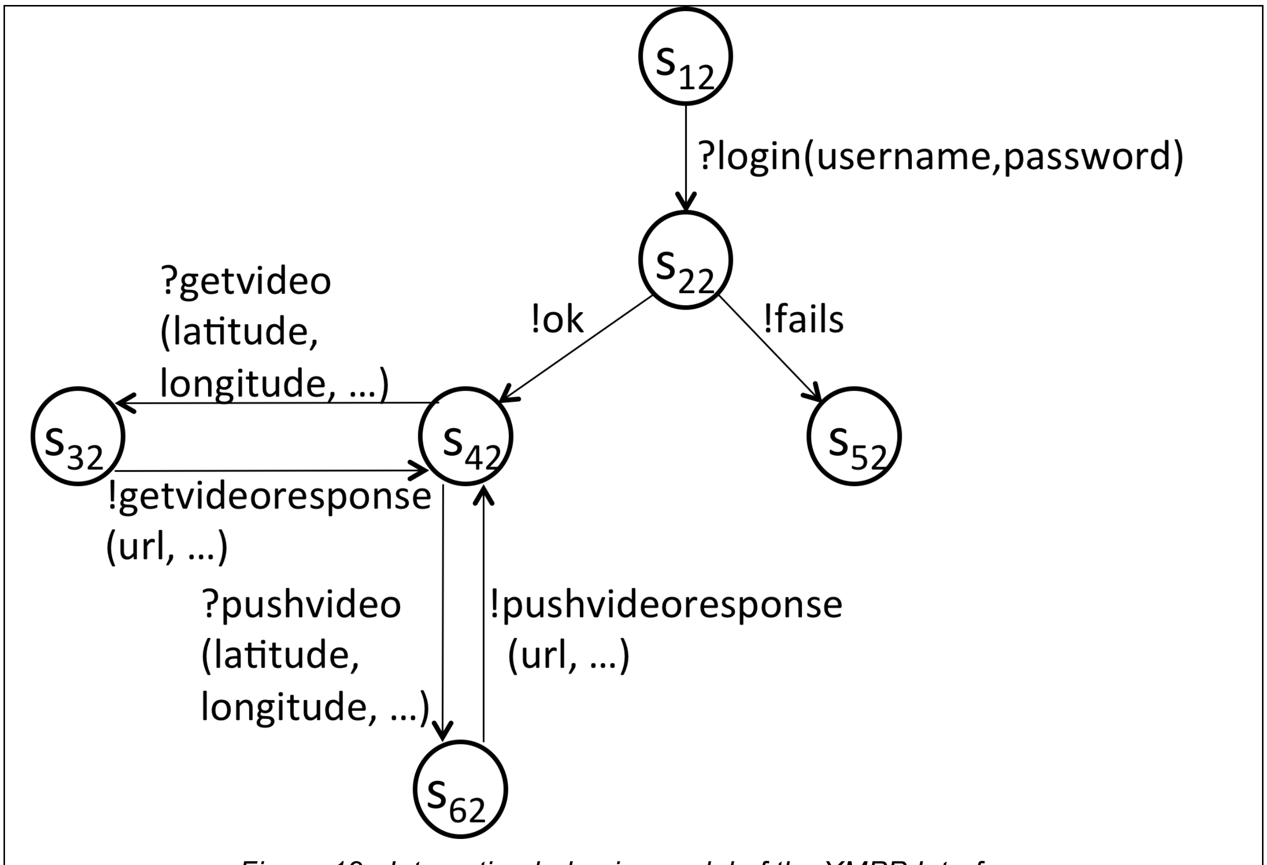
3.2.1 System Components

In this case, we have two very different systems that need to be connected. The first one is the Control Center, which is based Thales system, while the second one is the Video Server on the RCS side. As shown below, they follow different processes, they use different protocols, and the objective is to make them work together.

3.2.1.1 Control Center

The Control Center is the client of the Security Cameras. It is used to decide which camera to activate, to point the cameras to the right position and to get video out of them. This is a model of the system that a commander uses while interacting with the cameras and decides the zone of interest.

The process of the Control Center is shown in Figure 11[9]. It is composed of two phases: first two transitions of the automata control the discovery of cameras; the remaining transitions control the usage of the camera such as its position, the zoom level, and the transmission of videos.



s_{12}
?login(username,password)
s_{22}
!ok
!fails
?getvideo (latitude, longitude, …)
s_{32}
s_{42}
s_{52}
!getvideoresponse (url, …)
?pushvideo (latitude, longitude, …)
!pushvideoresponse (url, …)
s_{62}

Figure 12: Interaction behavior model of the XMPP Interface

3.2.1.2 Video Server

The process of the video server provided by RCS is shown in the automata in Figure 12. It shows first a login phase: if the login succeeds, it is followed by either getting a video or by sending a video, depending on the operation invoked by the client.

[9] In Figure 11as well as in all other pictures, the "?" indicates the sending of a message, while "!" indicates the reception of a message

3.2.1.3 Towards a CONNECTor Automata

Although the RCS video server and the Control Center client are two systems that have been designed independently from each other, the objective of this effort is to show that the Control Center can get videos from the video server. In a sense, it is as if, from the point of view of the Control Center, the video server is a camera with that cannot zoom and cannot change position.

A sketch of the automata of the CONNECTed system is shown in Figure 13. Essentially, it shows that only some of the capabilities of the control center and of the video server can be matched, and specifically the *getVideo* function on both sides, which is shaded in Figure 13. On the other hand, both systems provide functionalities that do not need to be CONNECTed since they do not contribute to the final CONNECTed system. Still, the login process of the video server triggered in state S_{12} is needed but does not have a counterpart with respect to the functionalities of the control center.

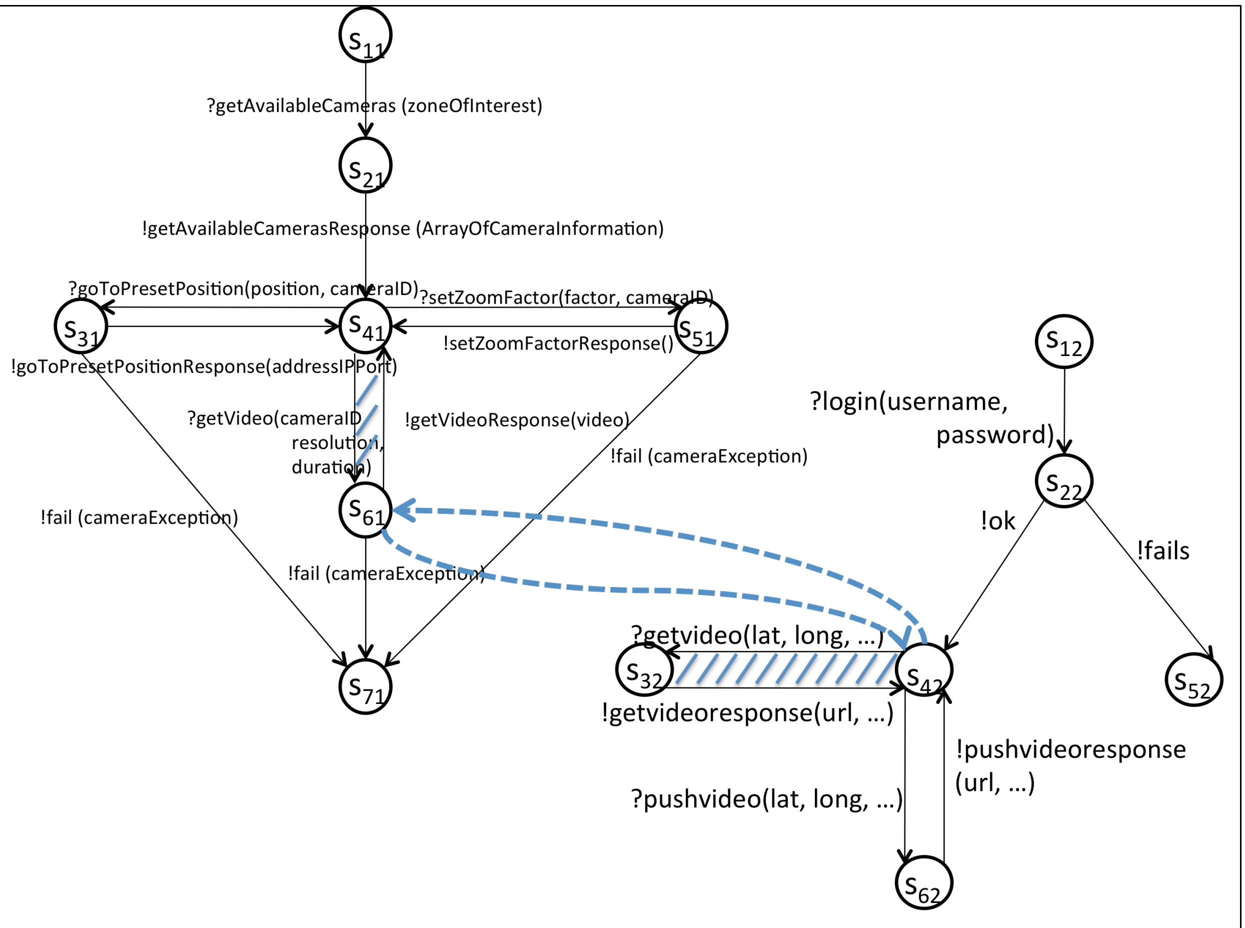

S11
?getAvailableCameras (zoneOfInterest)
S21
!getAvailableCamerasResponse (ArrayOfCameraInformation)
?goToPresetPosition(position, cameraID)
?setZoomFactor(factor, cameraID)
S31
S41
S51
!setZoomFactorResponse()
!goToPresetPositionResponse(addressIPPort)
?getVideo(cameraID, resolution, duration)
!getVideoResponse(video)
!fail (cameraException)
!fail (cameraException)
S61
!fail (cameraException)
S71
S12
?login(username, password)
S22
!ok
!fails
?getvideo(lat, long, …)
S32
S42
S52
!getvideoresponse(url, …)
!pushvideoresponse (url, …)
?pushvideo(lat, long, …)
S62

Figure 13: Automata of the CONNECTed system

The second aspect of the transformation is the transformation of the payload of the messages. The approach, in this case, is to annotate messages with the semantic annotation of the type of data transmitted, and to use this annotation to derive a data transformation function. The link annotation has been already developed and it is presented in D1.2, but its use on this example has not been tested yet.

3.2.2 Satisfy Architectural Requirements

In principle, the problem to construct a connector is just an issue of mediation among two systems, therefore it can be expected that once the automata sketched in Figure 13 is completed, then the process is complete. But the experience that we have made shows that the problem is more complex. Indeed, there are a number of requirements that need to be satisfied before the connector can be constructed. Below we analyze the problem of modeling additional constraints, namely: how to produce a correct architecture, and then how to construct a connector after that.

An overview of the DOCOMO RCS Testbed that will be used in the demonstration is shown in Figure 14. The figure shows that the testbed has two ports: the first one, shown on the left of Figure 14, is the standard SIP port used by any RCS client and any other device within the operator network. The second one, shown on the right of the figure and indicated as XMPP Gateway, which exploits the XMPP protocol (Saint-Andre, 2004) and then performs a translation to SIP (Saint-Andre, 2008) to communicate with the other components in the testbed. It is worth noticing that the XMPP Gateway is a non-standard XMPP port that gives access to devices on the Internet.

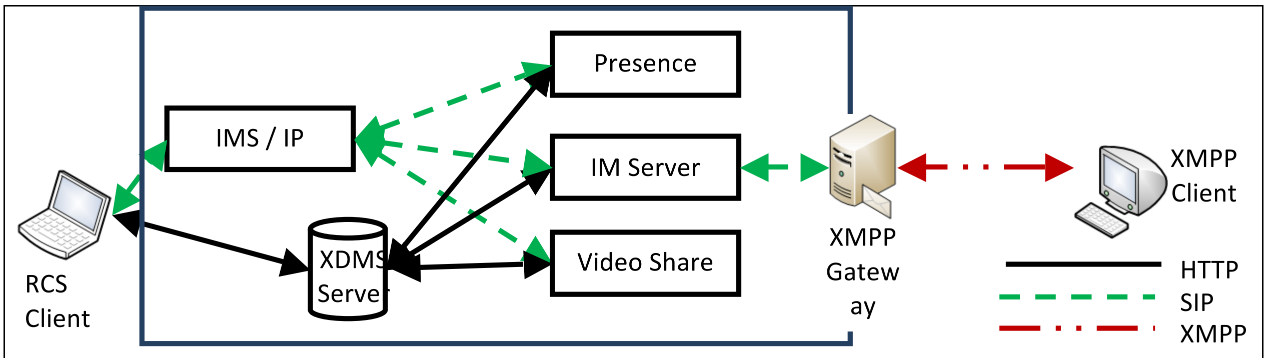


Figure 14: Overview of DOCOMO RCS Testbed Functional Architecture

As described more in details below, the construction of the CONNECTor is based on two different decisions: first which port to use, the SIP port vs XMPP Gateway, second, validating the overall infrastructure dealing with additional constraints.

3.2.2.1 Port Selection

The two ports provide exactly the same function; as a consequence their affordances are equivalent. The decision of which one of the two ports the CONNECTor needs to use hinges on other conditions that go beyond the functional properties of the two ports.

The use of the standard SIP port is more complex since it requires a previous registration to the mobile network, which in turns requires an exchange of keys and secrets that are typically stored in the SIM card. But the exchange of those secrets and the connection to the network provides also a number of guarantees such as communication security and identity as well as all the capabilities that are afforded to mobile phones such as roaming, billing, and a high level reliability, with a telco-grade guaranteed up time of 99.999%.

The use of the XMPP Gateway is instead much simpler. It is based on the standard XMPP protocol, and it avoids the registration with the core network since it is performed by the Gateway. On the other hand, it does provide lower security guarantees, no identity, no roaming, and a best-effort network as opposed to the high availability of the mobile network.

Ultimately the difference between the two ports is not only which protocol to match, i.e. SIP vs XMPP, but more importantly, the two ports differ by their own non-functional properties. The port selection therefore hinges on which keys the connecting system can provide, whether the 99.999% is worth the billing costs, or whether a free best effort is at the end good enough.

3.2.2.2 Validating the Infrastructure

Once the port is decided and the connector is available, it is also necessary to verify that no additional constraints have been introduced in the overall architecture as a consequence of the creation of the connected system.

For example, if the CONNECTor uses the XMPP port, an additional constraint is introduced; specifically, the XMPP gateway accepts connections only from clients that are in the same XMPP domain of the Gateway, specifically the DOCOMO Domain. Clients from other domains can connect only through their own server as long as this server is white listed by the gateway[10]. The corresponding architecture is shown in Figure 15, where the dotted lines represent the domain boundaries.

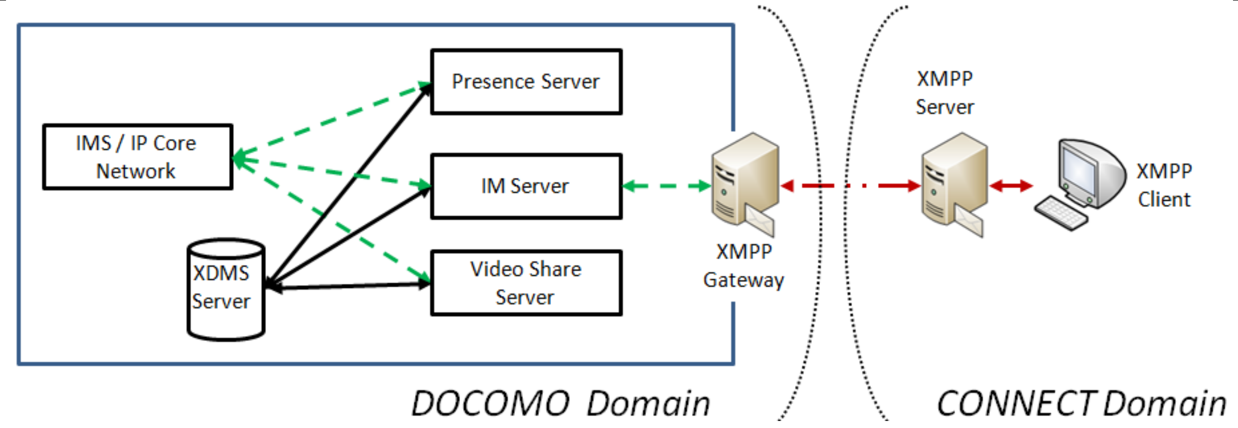


Figure 15: Functional Architecture with domain boundaries

The domain constraint is a new constraint that emerges from the way the two systems are connected. During the construction of the CONNECTor, there is therefore a need to verify these architectural constraints, and find solutions in case these constraints are not satisfied. .

3.2.2.3 Addressing the Selection and Infrastructure Problems

In summary the construction of CONNECTors between the Thales camera system and the DOCOMO RCS system depends on two decisions that have to be made on the systems to connect. Specifically:

1. A decision should be made on the port to use, this affects the protocols that have to be used, SIP vs. XMPP, keeping into account the security requirements that are involved.
2. A decision should be made on the validity of the overall architecture and specifically, if the XMPP port is selected, to address the XMPP domain issue.

To make these two decisions, we implemented an architecture ontology[11] that formalizes the required constraints. The purpose of implementing the ontology is twofold first we can describe the infrastructure that we have available in a declarative form, in other words we can create a data structure consistent with the ontology that describes the characteristics of each component, and how the different components relate. Second, we can exploit the reasoning that comes with the logic that supports the ontology so that we can derive conclusions on the requirements, what is missing and what kind of structure we need to construct.

[10] The constraint is slightly more complex: an external client should connect to a server which in turn is connected to another server that is white listed with the XMPP Gateway.

[11] A definition of Ontology in CONNECT is provided in D1.2..

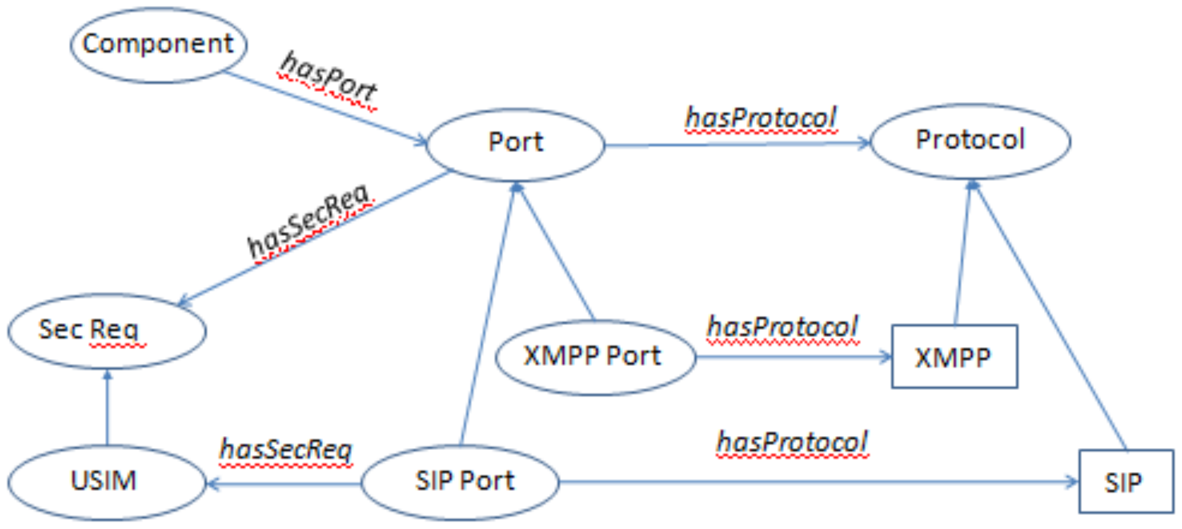


Figure 16: Fragment of the Architecture Ontology

The ontology fragment shown in Figure 16 highlights the concepts and relations used to address the problem of the port selection. The ontology specifies the concept of *Component*, *Port,* and the different types of ports such as *XMPP Port* and *SIP Port,* as well as the different types of protocols *XMPP* and *SIP.* Given this ontology, we can automatically identify that the different types of ports and the security requirements that are associated with them. For instance, the SIP port requires a valid *USIM*[12] while the XMPP port, for representation simplicity, does not require any security mechanism[13]. Therefore, the only way to recognize a port as a SIP port is to provide evidence of a valid USIM; conversely, if a port is declared to be a SIP Port and no USIM is provided, the inference engine would detect an error.

In our case, we can use the ontology to model the two CONNECTors that we can construct between the Control Center and the RCS system. More precisely, we can define the model in the following way: CONNECTors as instances of *Component;* for each of them we can specify the type of ports and the type and specify additional requirements. Upon exploitning the inference, the CONNECTor to the SIP port will be considered invalid since it is missing the required USIM; on the opposite, all the requirements for connecting to the XMPP port are satisfied, and therefore the connection succeeds.

The second problem, namely the construction of an architecture that bridges across the different domains is addressed by the ontology fragment shown in Figure 17.

[12] USIM stands for Universal Subscriber Identity Module ; it contains a secret key that allows to identify the telephone with the operator (3GPP_TS_11.11) (3GPP_TS_11.14, 2007).

[13] XMPP may require X509 certificates for message security, and this is represented in the ontology.

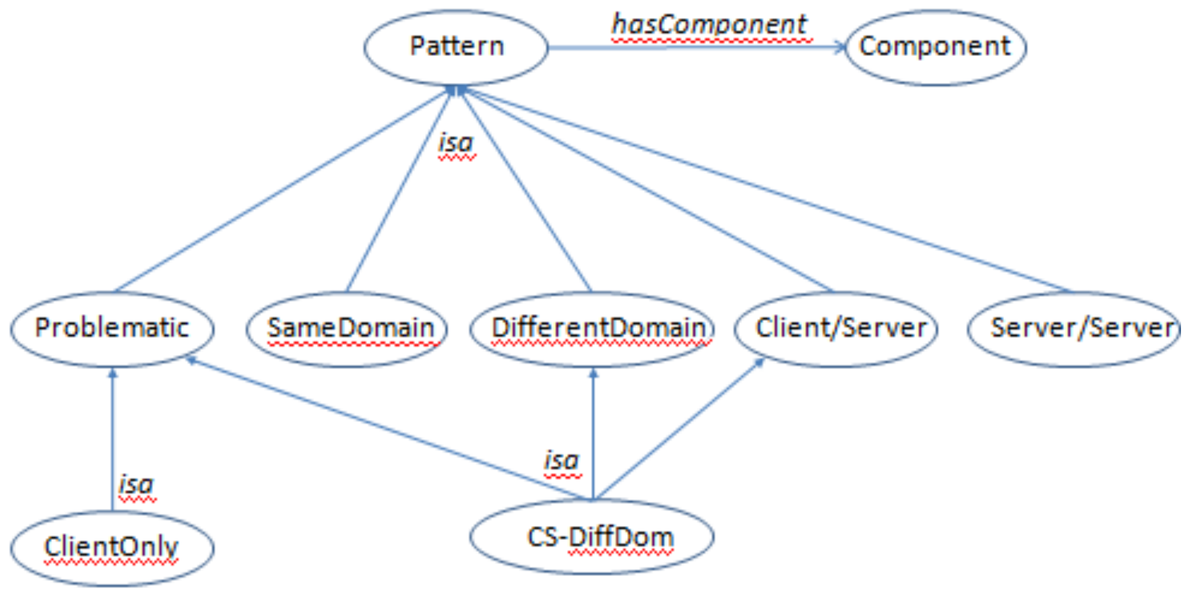


Figure 17: Ontology fragment specifying components domain

This ontology allows us to reason about patterns of components, i.e. whether they are in a client server relation, and whether the two systems are in the same domain or not. The class *Problematic* is defined to detect patterns that are problematic in the domain. Specifically, two components in a client/server relation but in different domains would be recognized as problematic. It is therefore possible to define remedial rules such as specifying how to recover from the problem. For example, such a rule could specify to add a new client/server pattern with a new server in the domain of the client and then connect the two servers directly to complete the system. Crucially, the CONNECT discovery process can be utilized here to find an XMPP server that satisfy the architectural requirements.

The result of the declarative modeling of the systems architecture in conjunction with the reasoning about such specifications allows us to identify what constraints are violated and how to address them. As a result of this reasoning process, we can verify that the use of the SIP port has requirements that cannot be satisfied, while the XMPP Gateway can be used as long as we can identify a XMPP server that can bridge the client with the Gateway. Forthcoming work

3.3 Evaluation

At the beginning of this section, we specified an experimental hypothesis "that all the enablers described in the architecture proposed in D1.2 are required to construct the CONNECTor required to address the scenario." Furthermore, we described 3 possible violations of this hypothesis: (1) some of the enablers in the CONNECT architecture will not be required; (2) that some enablers will have to be used in ways that are different than the use foreseen by the general CONNECT architecture; and that (3) additional requirements that are not satisfied by CONNECT than we will have to register a deviation from the experimental hypothesis.

The discovery enabler was explicitly used in section 3.1 for both the discovery of the cameras that were required in the system, as well as for the RCS video server to retrieve pictures from different points of view. Furthermore, the discovery enabler could have been used to find an XMPP server to address the architectural problems highlighted in section 3.2.2.2.

The synthesis enabler is clearly required to construct the CONNECTor in Figure 13.

The Dependability and Security and Trust enablers were required to select the RCS port to be used by the connector since that decision ultimately hinges on what security credential can be satisfied by the CONNECTor and on the quality of service that is desired for the CONNECTed system.

The deployment enabler was never mentioned, but clearly the CONNECTor has to be deployed and therefore its use is assumed.

The Learning enabler has not be used anywhere in the scenario. As a consequence, the initial experimental hypothesis in its whole has not been satisfied. Indeed, we had knowledge about all the systems involved; we knew all the details and documentation that allowed us to construct behavioral models. Despite this, it is also true that none of the behavioral models that we had to define for the experiment were given ahead of time. And some of them, for example the RCS XMPP port, proved to be quite time consuming and difficult to make work. Arguably, had we used the learning enabler, we would have achieved our result faster. Furthermore, the *Monitoring Enabler* was never used, but this is a fault of the experimental design that did not make space for such an enabler, and therefore this point is somewhat mute.

As for violations of type (2), since CONNECT is evolving quite fast is difficult to make any claim about unexpected uses. Still, two aspects of the process described above may generate new requirements for the rest of the project. The first one is that the synthesis of CONNECTors depends on the functionalities that are expected by the CONNECTed system, therefore given two systems, different CONNECTors can be constructed as function of what is expected from them. This issue is further stressed by the discussion in section 1. The second aspect is the use of non-functional properties to decide among different ports to be utilized during the construction of connectors. Both issues are already under the consideration of WP3 and WP5.

As for violations of type (3), namely additional requirements that are not satisfied by CONNECT, the need to model additional architectural constraints that emerge from the construction of connectors seems to fit in this category. This issue is already reflected in D1.2.

4 Bringing CONNECT to Development

In addition to the main effort carried out above, DOCOMO started a process of using CONNECT ideas and technologies for internal projects. RCS as described above highlights how in the vision of the mobile operators the future mobile network will contain a number of service enablers, some standardized others not, that will support applications that deeply integrate with the mobile network. One key ingredient of this vision is the ability to support the easy integration of enablers to create rich services. Clearly CONNECT technology should contribute to this vision, and our attempt was to start the process.

Given that the complete CONNECT cycle is not in place yet, this section is inevitably preliminary; furthermore, the goal of the attempt was mostly to identify how to include CONNECT in our internal development process, rather than make any technological progress. Nevertheless, this attempt uncovered issues that were only partially highlighted by the previous efforts.

The goal of the task was to build a new advanced service given two network enablers that have been implemented by different companies and that where proposed to DOCOMO to evaluate their possible commercial exploitation. For confidentiality reasons, we cannot disclose details about these services; nevertheless, we can highlight some of their basic characteristics.

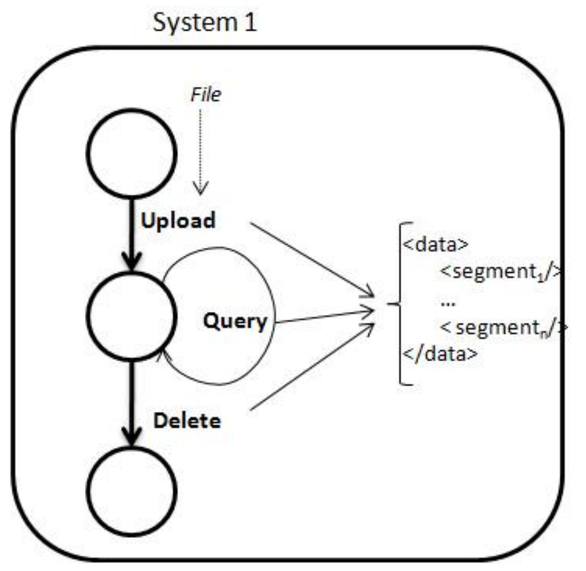


Figure 18: System 1 Automata

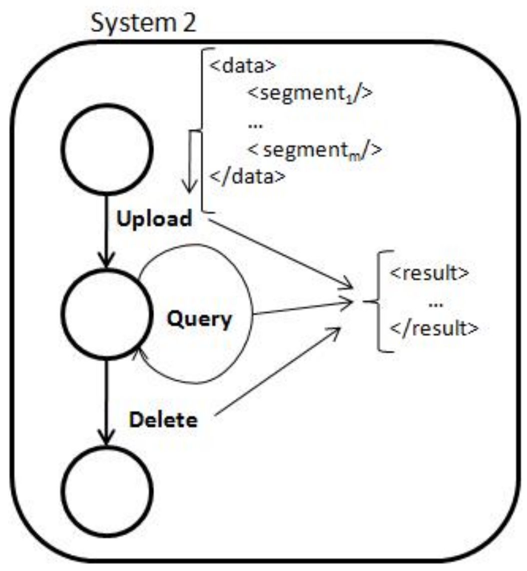


Figure 19: System 2 Automata

System 1, whose automata is shown in Figure 18, would take a file as input through the Up*load* operation and return a segmentation of the file; in addition it allows the repeated access to the list of segments to select the ones that have some desired features; finally, the delete operation supports the deletion of one or more segments. What is crucial here is that all three main operations of the system, namely *upload, query* and *delete,* despite their very different semantics would result in list of segments as output. In the first case, such list is the list of segments loaded, in the second is the list of segments selected, in the third is the list of segments deleted.

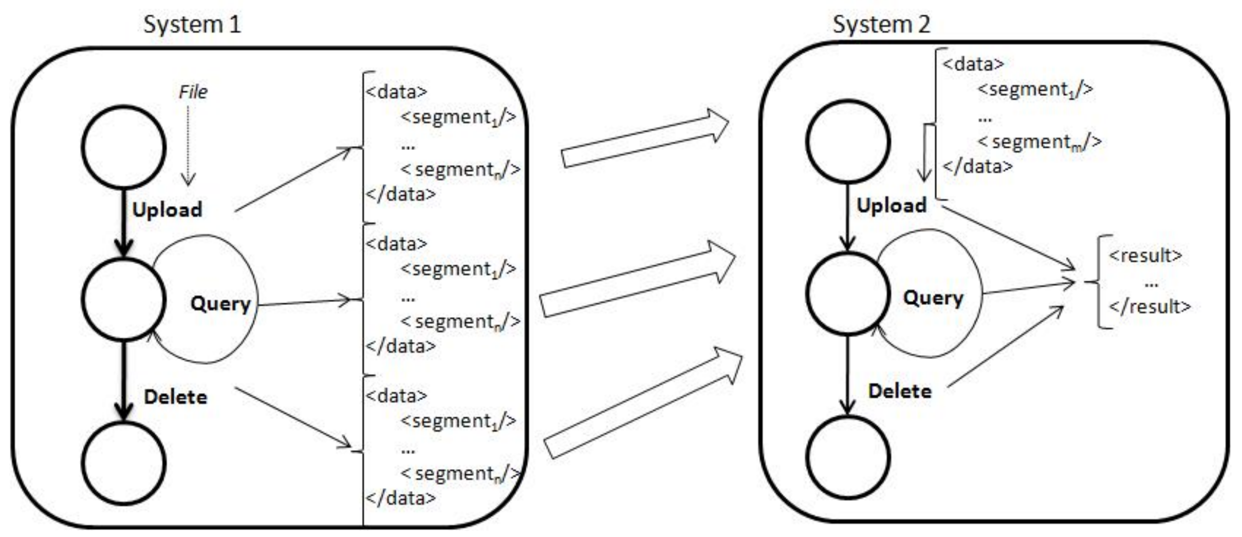


Figure 20: Different ways to combine the two Systems

The second system, whose automata is shown in Figure 19, system allows a list of segments to be uploaded and then manipulated. In this system there is also a similar query and delete operation, as in the first system.

Figure 20 shows how the two systems could be connected. Essentially we have three ways to connect the systems with three very different results. The first one is to use the list of segments generated by loading the file in the first system, as input for the second system. The second way is to take the list of segments selected by the first system and manipulate those. The third way is to take the list of segments deleted from the first system and pass them to the second system.

In the three cases, we will have three very different systems resulting from the connection of the two original ones. In the first case, the connector would be through the loaded scenes, in the second one through the viewed, and possibly selected, scenes, and in the third case, through the deleted scenes. Whereas the distinction between the three different systems is somewhat trivial, and further trivialized by the simplifications that we made in the description of the systems, this case also highlights there is no single connector among two systems. Indeed in the simplified case above we found 3 different connectors that we could build.

Ultimately, the construction of connectors depends on the goals of the system integrator, who needs to be able to specify which connector she/he intends to construct. At this point, somewhat not surprisingly, the task of CONNECT blurs into the task of service (or component) composition. But to the extent that the objective of CONNECT is to achieve run-time interoperation, then care should be taken to address problems like the one highlighted in this section.

As a final note, though very preliminary, attempts to discuss with other developers the use of CONNECT technology, which implies the construction of automata of the different systems was met with great skepticism. The argument being: "*By the time I build the automata and make it work, I already implemented the connector in java*". Since behavioral models are rarely provided, the work on learning performed in WP4 may prove a key technology toward the real deployment of CONNECT-type systems.

5 Conclusion and Future Work

The main lesson learned from looking at the connectors of different systems is that in general there are multiple connectors that can be built for the same two systems. In the experiments above we identified two different dimensions along which to organize these differences.

In section 3.2, above, we showed as different connectors could be built across the RCS and Thales video surveillance systems, and that these systems could strike different tradeoffs. Using the XMPP port facilitates the connector implementation, but at the cost of losing the close security and identity management that is performed within the mobile network. On the other side, exploiting the SIP port, while it could in principle give us high security and identity requirements, it would also increase the complexity because we would have for one gather the credentials, and for two, we would have to connect to two different systems: the first one to register with the network core; the second one to connect to the system we would like to exploit.

In section 1 we analyzed a different dimension along which connectors distinguish each other. This dimension is the goals of the connecting party. Indeed depending on these goals, different connectors can be built. The dependence of the connector on the affordance that is desired is also highlighted by the main scenario where the mapping of the two automata is only partial, and many capabilities of cameras have to be disregarded when the connector between the video server and the control center is constructed.

The future work will develop along three different directions. First, WP6 will continue the development of the scenario towards a definition of a more comprehensive GMES system, which will include many different devices and allow us to experiment with different forms of interoperability. Second, this initial scenario provided us with an initial approach to think about the evaluation of CONNECTed systems. In the third year of the project this effort should become more principled abstracting from specific cases toward a framework that allows in depth experimentation with the underlying architecture, middleware and enablers. Third, we will contribute closely with WP1 and all other WPs to produce an implemented CONNECT system. Indeed now we are in the position to integrate the different components.

In addition, two important issues are crucial and so far only partially addressed. The first one is the modeling of the data-level transformations between the two systems. Initial work in this direction has been done in WP1, but there is a more important problem of relating the semantics of the data to the message mapping proposed in D1.2. The second one is the performance consequences of different CONNECTor constructions. Specifically, a CONNECTor may result in a single point of failure, or in a performance bottleneck. We will exploit as much as possible results from WP5 to address these problems.

6 Bibliography

OMA. 2010. *OMA SIMPLE IM V1.0.* [Online] 2010 March. http://www.openmobilealliance.org/Technical/release_program/simple_im_v1_0.aspx.

3GPP. 2010. *3GPP TS 23.228: IP Multimedia Subsystem; Stage 2, v10.3.0.* [Online] 2010 December. http://www.3gpp.org/ftp/Specs/html-info/23228.htm.

3GPP_TS_11.11. 2010. Specification of the Subscriber Identity Module - Mobile Equipment (SIM-ME) Interface. *Linking Open Data Wiki.* [Online] [Cited: 2010 18-1.] http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData.

3GPP_TS_11.14. 2007. *Specification of the SIM Application Toolkit (SAT) for the Subscriber Identity Module - Mobile Equipment (SIM-ME) interface.* 2007.

Driscoll, Dan and Mensch, Antonine. 2009. *Devices Profile for Web Services Version 1.1.* s.l. : OASIS, 2009.

GSMA. 2010. *GSMA RCS Project.* [Online] 2010. http://www.gsmworld.com/our-work/mobile_lifestyle/rcs/gsma_rcs_project.htm.

GSMA-FD. 2010. *GSMA RCS Functional Description Release 1.* [Online] 2010 February. http://www.gsmworld.com/documents/FunctionalDescriptionv_v1.2.pdf.

Huynh, Trân, et al. 2010. *Deliverable D6.1 Experiment scenarios.* s.l. : CONNECT ICT FET IP Project, 2010.

P. Grace, G. Blair, et al. 2011. *CONNECT Deliverable 1.2: Intermediate CONNECT Architecture.* 2011.

Saint-Andre, P. 2004. *IETF RFC 3920, Extensible Messaging and Presence Protocol (XMPP).* [Online] 2004 October. http://www.ietf.org/rfc/rfc3920.txt.

—. 2004. *IETF RFC 3921, Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence.* [Online] 2004 October. http://www.ietf.org/rfc/rfc3921.txt.

—. 2008. *XEP-0078: Non-SASL Authentication.* [Online] 2008 October. http://xmpp.org/extensions/xep-0078.html.

—. 2008. *Interworking between the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP).* [Online] 2008 January. http://xmpp.org/internet-drafts/draft-saintandre-sip-xmpp-media-00.html.

7 Annexes

7.1 Annex 1 - WSDL Interfaces of Cameras Services

7.1.1 Camera214 Interface (PTZ)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<definitions targetNamespace="http://Camera213.demo.connect.thalesgroup.com/"
      name="Camera213ImplService" xmlns="http://schemas.xmlsoap.org/wsdl/"
      xmlns:tns="http://Camera214.demo.connect.thalesgroup.com/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:sawsdl="http://www.w3.org/2007/01/sawsdl#"
      xmlns:connect="http://connect.thalesgroup.com"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
      <types>
            <xs:schema version="1.0"
                  targetNamespace="http://Camera214.demo.connect.thalesgroup.com/"
                  xmlns:tns="http://Camera214.demo.connect.thalesgroup.com/"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
                  <xs:element name="CameraException" type="tns:CameraException" />
                  <xs:element name="goToPresetPosition"
type="tns:goToPresetPosition" />
                  <xs:element name="goToPresetPositionResponse"
type="tns:goToPresetPositionResponse" />
                  <xs:element name="setZoomFactor" type="tns:setZoomFactor" />
                  <xs:element name="setZoomFactorResponse"
type="tns:setZoomFactorResponse" />
                  <xs:element name="getVideo" type="tns:getVideo" />
                  <xs:element name="getVideoResponse" type="tns:getVideoResponse"
/>
                  <xs:complexType name="getVideo">
                        <xs:sequence>
                              <xs:element name="resolution" type="xs:string"
                                    minOccurs="0" />
                              <xs:element name="duration" type="xs:int"
minOccurs="0" />
                        </xs:sequence>
                  </xs:complexType>
                  <xs:complexType name="getVideoResponse">
                        <xs:sequence>
                              <xs:element name="video" type="xs:anyType" />
                        </xs:sequence>
                  </xs:complexType>
                  <xs:complexType name="setZoomFactor">
                        <xs:sequence>
                              <xs:element name="factor" type="xs:int" />
                        </xs:sequence>
                  </xs:complexType>
                  <xs:complexType name="setZoomFactorResponse">
                        <xs:sequence />
                  </xs:complexType>
                  <xs:complexType name="CameraException">
                        <xs:sequence>
                              <xs:element name="message" type="xs:string"
minOccurs="0" />
                        </xs:sequence>
                  </xs:complexType>
                  <xs:complexType name="goToPresetPosition">
                        <xs:sequence>
                              <xs:element name="position" type="xs:string"
minOccurs="0" />
                        </xs:sequence>
                  </xs:complexType>
                  <xs:complexType name="goToPresetPositionResponse">
                        <xs:sequence />
                  </xs:complexType>

                  <!-- Data for CONNECT discovery -->
                  <xs:element name="connectInput">
                        <xs:simpleType>
```

```
                                <xs:restriction base="xs:string">
                                        <xs:enumeration value="GetAffordance" />
                                        <xs:enumeration value="GetSemInterface" />
                                        <xs:enumeration value="GetBehaviour" />
                                        <xs:enumeration value="GetNFP" />
                                </xs:restriction>
                        </xs:simpleType>
                </xs:element>
                <xs:element name="fileString" type="xs:string" />
                <xs:element name="IOException" type="xs:string" />
        </xs:schema>
</types>
<message name="goToPresetPosition">
        <part name="parameters" element="tns:goToPresetPosition" />
</message>
<message name="goToPresetPositionResponse">
        <part name="parameters" element="tns:goToPresetPositionResponse" />
</message>
<message name="CameraException">
        <part name="fault" element="tns:CameraException" />
</message>
<message name="setZoomFactor">
        <part name="parameters" element="tns:setZoomFactor" />
</message>
<message name="setZoomFactorResponse">
        <part name="parameters" element="tns:setZoomFactorResponse" />
</message>
<message name="getVideo">
        <part name="parameters" element="tns:getVideo" />
</message>
<message name="getVideoResponse">
        <part name="parameters" element="tns:getVideoResponse" />
</message>

<message name="connectInput">
        <part name="parameters" element="tns:connectInput" />
</message>
<message name="fileString">
        <part name="parameters" element="tns:fileString" />
</message>
<message name="IOException">
        <part name="parameters" element="tns:IOException" />
</message>


<portType name="Camera214Impl">
        <operation name="goToPresetPosition">
                <input message="tns:goToPresetPosition"

        wsa:Action="http://Camera214.demo.connect.thalesgroup.com/Camera214Impl/goToPr
esetPosition" />
                <output message="tns:goToPresetPositionResponse"

        wsa:Action="http://Camera214.demo.connect.thalesgroup.com/Camera214Impl/goToPr
esetPositionResponse" />
                <fault message="tns:CameraException" name="CameraException"

        wsa:Action="http://Camera214.demo.connect.thalesgroup.com/Camera214Impl/Camera
Exception" />
        </operation>
        <operation name="setZoomFactor">
                <input message="tns:setZoomFactor"

        wsa:Action="http://Camera214.demo.connect.thalesgroup.com/Camera214Impl/setZoo
mFactor" />
                <output message="tns:setZoomFactorResponse"

        wsa:Action="http://Camera214.demo.connect.thalesgroup.com/Camera214Impl/setZoo
mFactorResponse" />
                <fault message="tns:CameraException" name="CameraException"

        wsa:Action="http://Camera214.demo.connect.thalesgroup.com/Camera214Impl/Camera
Exception" />
        </operation>
        <operation name="getVideo">
                <input message="tns:getVideo"
```

```
	wsa:Action="http://Camera214.demo.connect.thalesgroup.com/Camera214Impl/getVid
eo" />
				<output message="tns:getVideoResponse"

	wsa:Action="http://Camera214.demo.connect.thalesgroup.com/Camera214Impl/getVid
eoResponse" />
				<fault message="tns:CameraException" name="CameraException"

	wsa:Action="http://Camera214.demo.connect.thalesgroup.com/Camera214Impl/Camera
Exception" />
			</operation>
			<operation name="connectPrim">
				<input message="tns:connectInput"

	wsa:Action="http://Camera214.demo.connect.thalesgroup.com/Camera214Impl/connec
tPrim" />
				<output message="tns:fileString"

	wsa:Action="http://Camera214.demo.connect.thalesgroup.com/Camera214Impl/fileSt
ring" />
				<fault message="tns:IOException" name="connectException"

	wsa:Action="http://Camera214.demo.connect.thalesgroup.com/Camera214Impl/connec
tException" />
			</operation>
		</portType>
		<binding name="Camera214ImplPortBinding" type="tns:Camera214Impl">
			<soap12:binding style="document"
				transport="http://schemas.xmlsoap.org/soap/http" />
			<operation name="goToPresetPosition">
				<soap12:operation soapAction="" />
				<input>
					<soap12:body use="literal" />
				</input>
				<output>
					<soap12:body use="literal" />
				</output>
				<fault name="CameraException">
					<soap12:fault name="CameraException" use="literal" />
				</fault>
			</operation>
			<operation name="setZoomFactor">
				<soap12:operation soapAction="" />
				<input>
					<soap12:body use="literal" />
				</input>
				<output>
					<soap12:body use="literal" />
				</output>
				<fault name="CameraException">
					<soap12:fault name="CameraException" use="literal" />
				</fault>
			</operation>
			<operation name="getVideo">
				<soap12:operation soapAction="" />
				<input>
					<soap12:body use="literal" />
				</input>
				<output>
					<soap12:body use="literal" />
				</output>
				<fault name="CameraException">
					<soap12:fault name="CameraException" use="literal" />
				</fault>
			</operation>
			<operation name="connectPrim">
				<soap12:operation soapAction="" />
				<input>
					<soap12:body use="literal" />
				</input>
				<output>
					<soap12:body use="literal" />
				</output>
				<fault name="connectException">
					<soap12:fault name="connectException" use="literal" />
				</fault>
			</operation>
```

```
       </binding>
       <service name="Camera214ImplService">
              <port name="Camera214ImplPort" binding="tns:Camera214ImplPortBinding">
                     <soap12:address
location="http://localhost:8080/CONNECT/Camera214" />
              </port>
       </service>
</definitions>
```

7.1.2 Camera213 Interface (PTZ)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<definitions targetNamespace="http://Camera213.demo.connect.thalesgroup.com/"
       name="Camera213ImplService" xmlns="http://schemas.xmlsoap.org/wsdl/"
       xmlns:tns="http://Camera213.demo.connect.thalesgroup.com/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:sawsdl="http://www.w3.org/2007/01/sawsdl#"
       xmlns:connect="http://connect.thalesgroup.com"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
       <types>
              <xs:schema version="1.0"
                     targetNamespace="http://Camera213.demo.connect.thalesgroup.com/"
                     xmlns:tns="http://Camera213.demo.connect.thalesgroup.com/"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
                     <xs:element name="CameraException" type="tns:CameraException" />
                     <xs:element name="goToPresetPosition"
type="tns:goToPresetPosition" />
                     <xs:element name="goToPresetPositionResponse"
type="tns:goToPresetPositionResponse" />
                     <xs:element name="setZoomFactor" type="tns:setZoomFactor" />
                     <xs:element name="setZoomFactorResponse"
type="tns:setZoomFactorResponse" />
                     <xs:element name="getVideo" type="tns:getVideo" />
                     <xs:element name="getVideoResponse" type="tns:getVideoResponse"
/>
                     <xs:complexType name="getVideo">
                            <xs:sequence>
                                   <xs:element name="resolution" type="xs:string"
                                          minOccurs="0" />
                                   <xs:element name="duration" type="xs:int"
minOccurs="0" />
                            </xs:sequence>
                     </xs:complexType>
                     <xs:complexType name="getVideoResponse">
                            <xs:sequence>
                                   <xs:element name="video" type="xs:anyType" />
                            </xs:sequence>
                     </xs:complexType>
                     <xs:complexType name="setZoomFactor">
                            <xs:sequence>
                                   <xs:element name="factor" type="xs:int" />
                            </xs:sequence>
                     </xs:complexType>
                     <xs:complexType name="setZoomFactorResponse">
                            <xs:sequence />
                     </xs:complexType>
                     <xs:complexType name="CameraException">
                            <xs:sequence>
                                   <xs:element name="message" type="xs:string"
minOccurs="0" />
                            </xs:sequence>
                     </xs:complexType>
                     <xs:complexType name="goToPresetPosition">
                            <xs:sequence>
                                   <xs:element name="position" type="xs:string"
minOccurs="0" />
                            </xs:sequence>
                     </xs:complexType>
                     <xs:complexType name="goToPresetPositionResponse">
                            <xs:sequence />
                     </xs:complexType>
```

```
				<!-- Data for CONNECT discovery -->
				<xs:element name="connectInput">
					<xs:simpleType>
						<xs:restriction base="xs:string">
							<xs:enumeration value="GetAffordance" />
							<xs:enumeration value="GetSemInterface" />
							<xs:enumeration value="GetBehaviour" />
							<xs:enumeration value="GetNFP" />
						</xs:restriction>
					</xs:simpleType>
				</xs:element>
				<xs:element name="fileString" type="xs:string" />
				<xs:element name="IOException" type="xs:string" />
			</xs:schema>
	</types>
	<message name="goToPresetPosition">
		<part name="parameters" element="tns:goToPresetPosition" />
	</message>
	<message name="goToPresetPositionResponse">
		<part name="parameters" element="tns:goToPresetPositionResponse" />
	</message>
	<message name="CameraException">
		<part name="fault" element="tns:CameraException" />
	</message>
	<message name="setZoomFactor">
		<part name="parameters" element="tns:setZoomFactor" />
	</message>
	<message name="setZoomFactorResponse">
		<part name="parameters" element="tns:setZoomFactorResponse" />
	</message>
	<message name="getVideo">
		<part name="parameters" element="tns:getVideo" />
	</message>
	<message name="getVideoResponse">
		<part name="parameters" element="tns:getVideoResponse" />
	</message>

	<message name="connectInput">
		<part name="parameters" element="tns:connectInput" />
	</message>
	<message name="fileString">
		<part name="parameters" element="tns:fileString" />
	</message>
	<message name="IOException">
		<part name="parameters" element="tns:IOException" />
	</message>


	<portType name="Camera213Impl">
		<operation name="goToPresetPosition">
			<input message="tns:goToPresetPosition"

	wsa:Action="http://Camera213.demo.connect.thalesgroup.com/Camera213Impl/goToPr
esetPosition" />
			<output message="tns:goToPresetPositionResponse"

	wsa:Action="http://Camera213.demo.connect.thalesgroup.com/Camera213Impl/goToPr
esetPositionResponse" />
			<fault message="tns:CameraException" name="CameraException"

	wsa:Action="http://Camera213.demo.connect.thalesgroup.com/Camera213Impl/Camera
Exception" />
		</operation>
		<operation name="setZoomFactor">
			<input message="tns:setZoomFactor"

	wsa:Action="http://Camera213.demo.connect.thalesgroup.com/Camera213Impl/setZoo
mFactor" />
			<output message="tns:setZoomFactorResponse"

	wsa:Action="http://Camera213.demo.connect.thalesgroup.com/Camera213Impl/setZoo
mFactorResponse" />
			<fault message="tns:CameraException" name="CameraException"

	wsa:Action="http://Camera213.demo.connect.thalesgroup.com/Camera213Impl/Camera
Exception" />
		</operation>
```

```
              <operation name="getVideo">
                     <input message="tns:getVideo"

       wsa:Action="http://Camera213.demo.connect.thalesgroup.com/Camera213Impl/getVid
eo" />
                     <output message="tns:getVideoResponse"

       wsa:Action="http://Camera213.demo.connect.thalesgroup.com/Camera213Impl/getVid
eoResponse" />
                     <fault message="tns:CameraException" name="CameraException"

       wsa:Action="http://Camera213.demo.connect.thalesgroup.com/Camera213Impl/Camera
Exception" />
              </operation>
              <operation name="connectPrim">
                     <input message="tns:connectInput"

       wsa:Action="http://Camera213.demo.connect.thalesgroup.com/Camera213Impl/connec
tPrim" />
                     <output message="tns:fileString"

       wsa:Action="http://Camera213.demo.connect.thalesgroup.com/Camera213Impl/fileSt
ring" />
                     <fault message="tns:IOException" name="connectException"

       wsa:Action="http://Camera213.demo.connect.thalesgroup.com/Camera213Impl/connec
tException" />
              </operation>
       </portType>
       <binding name="Camera213ImplPortBinding" type="tns:Camera213Impl">
              <soap12:binding style="document"
                     transport="http://schemas.xmlsoap.org/soap/http" />
              <operation name="goToPresetPosition">
                     <soap12:operation soapAction="" />
                     <input>
                            <soap12:body use="literal" />
                     </input>
                     <output>
                            <soap12:body use="literal" />
                     </output>
                     <fault name="CameraException">
                            <soap12:fault name="CameraException" use="literal" />
                     </fault>
              </operation>
              <operation name="setZoomFactor">
                     <soap12:operation soapAction="" />
                     <input>
                            <soap12:body use="literal" />
                     </input>
                     <output>
                            <soap12:body use="literal" />
                     </output>
                     <fault name="CameraException">
                            <soap12:fault name="CameraException" use="literal" />
                     </fault>
              </operation>
              <operation name="getVideo">
                     <soap12:operation soapAction="" />
                     <input>
                            <soap12:body use="literal" />
                     </input>
                     <output>
                            <soap12:body use="literal" />
                     </output>
                     <fault name="CameraException">
                            <soap12:fault name="CameraException" use="literal" />
                     </fault>
              </operation>
              <operation name="connectPrim">
                     <soap12:operation soapAction="" />
                     <input>
                            <soap12:body use="literal" />
                     </input>
                     <output>
                            <soap12:body use="literal" />
                     </output>
                     <fault name="connectException">
```

```
                            <soap12:fault name="connectException" use="literal" />
                     </fault>
              </operation>
       </binding>
       <service name="Camera213ImplService">
              <port name="Camera213ImplPort" binding="tns:Camera213ImplPortBinding">
                     <soap12:address
location="http://localhost:8080/CONNECT/Camera213" />
              </port>
       </service>
</definitions>
```

7.1.3 Camera211 Interface (Fixed)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<definitions targetNamespace="http://Camera211.demo.connect.thalesgroup.com/"
       name="Camera211ImplService" xmlns="http://schemas.xmlsoap.org/wsdl/"
       xmlns:tns="http://Camera211.demo.connect.thalesgroup.com/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:sawsdl="http://www.w3.org/2007/01/sawsdl#"
       xmlns:connect="http://connect.thalesgroup.com"
xmlns:wsa="http://www.w3.org/2005/08/addressing">
       <types>
              <xs:schema version="1.0"
                     targetNamespace="http://Camera211.demo.connect.thalesgroup.com/"
                     xmlns:tns="http://Camera211.demo.connect.thalesgroup.com/"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
                     <xs:element name="CameraException" type="tns:CameraException" />
                     <xs:element name="goToPresetPosition"
type="tns:goToPresetPosition" />
                     <xs:element name="goToPresetPositionResponse"
type="tns:goToPresetPositionResponse" />
                     <xs:element name="setZoomFactor" type="tns:setZoomFactor" />
                     <xs:element name="setZoomFactorResponse"
type="tns:setZoomFactorResponse" />
                     <xs:element name="getVideo" type="tns:getVideo" />
                     <xs:element name="getVideoResponse" type="tns:getVideoResponse"
/>
                     <xs:complexType name="getVideo">
                            <xs:sequence>
                                   <xs:element name="resolution" type="xs:string"
                                          minOccurs="0" />
                                   <xs:element name="duration" type="xs:int"
minOccurs="0" />
                            </xs:sequence>
                     </xs:complexType>
                     <xs:complexType name="getVideoResponse">
                            <xs:sequence>
                                   <xs:element name="video" type="xs:anyType" />
                            </xs:sequence>
                     </xs:complexType>
                     <xs:complexType name="setZoomFactor">
                            <xs:sequence>
                                   <xs:element name="factor" type="xs:int" />
                            </xs:sequence>
                     </xs:complexType>
                     <xs:complexType name="setZoomFactorResponse">
                            <xs:sequence />
                     </xs:complexType>
                     <xs:complexType name="CameraException">
                            <xs:sequence>
                                   <xs:element name="message" type="xs:string"
minOccurs="0" />
                            </xs:sequence>
                     </xs:complexType>


                     <!-- Data for CONNECT discovery -->
                     <xs:element name="connectInput">
                            <xs:simpleType>
                                   <xs:restriction base="xs:string">
```

```
                                        <xs:enumeration value="GetAffordance" />
                                        <xs:enumeration value="GetSemInterface" />
                                        <xs:enumeration value="GetBehaviour" />
                                        <xs:enumeration value="GetNFP" />
                                </xs:restriction>
                        </xs:simpleType>
                </xs:element>
                <xs:element name="fileString" type="xs:string" />
                <xs:element name="IOException" type="xs:string" />
        </xs:schema>
    </types>
    <message name="CameraException">
        <part name="fault" element="tns:CameraException" />
    </message>
    <message name="setZoomFactor">
        <part name="parameters" element="tns:setZoomFactor" />
    </message>
    <message name="setZoomFactorResponse">
        <part name="parameters" element="tns:setZoomFactorResponse" />
    </message>
    <message name="getVideo">
        <part name="parameters" element="tns:getVideo" />
    </message>
    <message name="getVideoResponse">
        <part name="parameters" element="tns:getVideoResponse" />
    </message>

    <message name="connectInput">
        <part name="parameters" element="tns:connectInput" />
    </message>
    <message name="fileString">
        <part name="parameters" element="tns:fileString" />
    </message>
    <message name="IOException">
        <part name="parameters" element="tns:IOException" />
    </message>


    <portType name="Camera211Impl">
        <operation name="setZoomFactor">
            <input message="tns:setZoomFactor"

    wsa:Action="http://Camera211.demo.connect.thalesgroup.com/Camera211Impl/setZoo
mFactor" />
            <output message="tns:setZoomFactorResponse"

    wsa:Action="http://Camera211.demo.connect.thalesgroup.com/Camera211Impl/setZoo
mFactorResponse" />
            <fault message="tns:CameraException" name="CameraException"

    wsa:Action="http://Camera211.demo.connect.thalesgroup.com/Camera211Impl/Camera
Exception" />
        </operation>
        <operation name="getVideo">
            <input message="tns:getVideo"

    wsa:Action="http://Camera211.demo.connect.thalesgroup.com/Camera211Impl/getVid
eo" />
            <output message="tns:getVideoResponse"

    wsa:Action="http://Camera211.demo.connect.thalesgroup.com/Camera211Impl/getVid
eoResponse" />
            <fault message="tns:CameraException" name="CameraException"

    wsa:Action="http://Camera211.demo.connect.thalesgroup.com/Camera211Impl/Camera
Exception" />
        </operation>
        <operation name="connectPrim">
            <input message="tns:connectInput"

    wsa:Action="http://Camera211.demo.connect.thalesgroup.com/Camera211Impl/connec
tPrim" />
            <output message="tns:fileString"

    wsa:Action="http://Camera211.demo.connect.thalesgroup.com/Camera211Impl/fileSt
ring" />
            <fault message="tns:IOException" name="connectException"
```

```
	wsa:Action="http://Camera211.demo.connect.thalesgroup.com/Camera211Impl/connec
tException" />
		</operation>
	</portType>
	<binding name="Camera211ImplPortBinding" type="tns:Camera211Impl">
		<soap12:binding style="document"
			transport="http://schemas.xmlsoap.org/soap/http" />
		<operation name="setZoomFactor">
			<soap12:operation soapAction="" />
			<input>
				<soap12:body use="literal" />
			</input>
			<output>
				<soap12:body use="literal" />
			</output>
			<fault name="CameraException">
				<soap12:fault name="CameraException" use="literal" />
			</fault>
		</operation>
		<operation name="getVideo">
			<soap12:operation soapAction="" />
			<input>
				<soap12:body use="literal" />
			</input>
			<output>
				<soap12:body use="literal" />
			</output>
			<fault name="CameraException">
				<soap12:fault name="CameraException" use="literal" />
			</fault>
		</operation>
		<operation name="connectPrim">
			<soap12:operation soapAction="" />
			<input>
				<soap12:body use="literal" />
			</input>
			<output>
				<soap12:body use="literal" />
			</output>
			<fault name="connectException">
				<soap12:fault name="connectException" use="literal" />
			</fault>
		</operation>
	</binding>
	<service name="Camera211ImplService">
		<port name="Camera211ImplPort" binding="tns:Camera211ImplPortBinding">
			<soap12:address
location="http://localhost:8080/CONNECT/Camera211" />
		</port>
	</service>
</definitions>
```

7.2 Annex 2: Commander System SOAP Interface

```
<xs:complexType name="latitudeLongitude">
	<xs:sequence>
		<xs:element name="Latitude" type="xs:double"/>
		<xs:element name="Longitude" type="xs:double"/>
	</xs:sequence>
</xs:complexType>

<xs:complexType name="zoneOfInterest">
	<xs:sequence>
		<xs:element name="latitudeLongitude"
	type="tns:latitudeLongitude"/>
		<xs:element name="Radius" type="xs:double"/>
			<xs:annotation>
				<xs:documentation>Radius of the zone of interest
			expressed in meters
```

```
                        </xs:documentation>
                  </xs:annotation>
            </xs:element>
      </xs:sequence>
</xs:complexType>

<xs:complexType name="addressIPPort">
      <xs:sequence>
            <xs:element name="ip" type="xs:string" minOccurs="0"/>
            <xs:element name="port" type="xs:int"/>
      </xs:sequence>
</xs:complexType>

<xs:complexType name="cameraInformation”>
      <xs:sequence>
            <xs:element name="Manufacturer" type="xs:string"/>
            <xs:element name="CameraId" type="xs:string">
                  <xs:annotation>
                        <xs:documentation>The cameraId corresponds to the
                  hardware ID of the device.
                        </xs:documentation>
                  </xs:annotation>
            </xs:element>
            <xs:element name="addressIPPort" type="tns:addressIPPort"/>
            <xs:element name="latitudeLongitude"
      type="tns:latitudeLongitude"/>
      </xs:sequence>
</xs:complexType>

<xs:complexType name="getAvailableCameras">
      <xs:sequence>
            <xs:element name="zoneOfInterest" type="tns:zoneOfInterest"/>
      </xs:sequence>
</xs:complexType>

<xs:complexType name="getAvailableCamerasResponse">
      <xs:sequence>
            <xs:element name="cameraInformation"
type="tns:cameraInformation"/>
      </xs:sequence>
</xs:complexType>

<xs:complexType name="setZoomFactor">
      <xs:sequence>
            <xs:element name="factor" type="xs:int"/>
            <xs:element name="cameraID" type="xs:string"/>
      </xs:sequence>
</xs:complexType>

<xs:complexType name="setZoomFactorResponse">
      <xs:sequence />
</xs:complexType>

<xs:complexType name="goToPresetPosition">
      <xs:sequence>
            <xs:element name="position" type="xs:string" minOccurs="0"/>
            <xs:element name="cameraID" type="xs:string">
      </xs:sequence>
</xs:complexType>

<xs:complexType name="goToPresetPositionResponse">
```

```
      <xs:sequence>
            <xs:element name="return" type="tns:addressIPPort"
            minOccurs="0"/>
      </xs:sequence>
</xs:complexType>

<xs:complexType name="CameraException">
      <xs:sequence>
            <xs:element name="message" type="xs:string" minOccurs="0" />
      </xs:sequence>
</xs:complexType>
…
<operation name="getAvailableCameras">
      <input message="tns:getAvailableCameras " />
      <output message="tns:getAvailableCamerasResponse" />
</operation>
<operation name="setZoomFactor">
      <input message="tns:setZoomFactor" />
      <output message="tns:setZoomFactorResponse" />
      <fault message="tns:CameraException" name="CameraException" />
</operation>
<operation name="goToPresetPosition">
      <input message="tns:goToPresetPosition" />
      <output message="tns:goToPresetPositionResponse" />
      <fault message="tns:CameraException" name="CameraException" />
</operation>
```

7.3 Annex 3: XMPP Messages

Step 1: Client to Server (Initial stream)

<stream:stream

 to=”connect.eu” xmlns=”jabber:client” xmlns:stream=”http://etherx.jabber.org/streams”>

Step 2: Server to Client (Response with a stream header)

<stream:stream

xmlns:stream="http://etherx.jabber.org/streams" xmlns="jabber:client"

from="connect.eu" id="e66745af" xml:lang="en" version="1.0">

Step 3: Server to Client (List of stream features supported by the server)

<stream:features>

<starttls xmlns=”urn:ietf:params:xml:ns:xmpp-tls”>

</starttls>

<mechanisms xmlns="urn:ietf:params:xml:ns:xmpp-sasl">

<mechanism>DIGEST-MD5</mechanism>

<mechanism>PLAIN</mechanism>

<mechanism>ANONYMOUS</mechanism>

<mechanism>CRAM-MD5</mechanism>

</mechanisms>

```
<compression xmlns="http://jabber.org/features/compress">
<method>zlib</method>
</compression>
<auth xmlns="http://jabber.org/features/iq-auth"/>
<register xmlns="http://jabber.org/features/iq-register"/>
</stream:features>
```

Step 4: Client to Server (Request for non-SASL authentication)

```
<iq
id="nql1w-0"
type="get">
<query xmlns="jabber:iq:auth">
<username>alice</username>
</query>
</iq>
```

Step 5: Server to Client (Response from server with authentication fields)

```
<iq
 type="result"
id="nql1w-0">
<query xmlns="jabber:iq:auth">
<username>alice</username><password/><digest/><resource/></query></iq>
```

Step 6: Client to Server (Client provides its credentials (e.g. username, digest/encrypted password)

```
<iq
id="nql1w-1"
type="set">
<query xmlns="jabber:iq:auth">
<username>alice</username>
<digest>f66c3c9df16f3c6aaf21e8e16613f4308b30ee41</digest>
<resource>Smack</resource></query></iq
```

Step 7A: Server to Client (Authentication was successful)

```
<iq type="result" id="nql1w-1" to="alice@connect.eu/Smack"/>
```

Step 7B:Server to Client (Authentication failed)

```
<iq type="error" id="nql1w-1" to="connect.eu/37a6d498">
```

```
<query xmlns="jabber:iq:auth">
 <username>alice</username>
 <digest>ddedf27c1f194e9dd60bc02a656914aa55e54a90</digest>
 <resource>Smack</resource>
 </query>
 <error code="401" type="AUTH">
 <not-authorized xmlns="urn:ietf:params:xml:ns:xmpp-stanzas"/>
 </error>
</iq>
<error code="401" type="auth">
<not-authorized xmlns="urn:ietf:params:xml:ns:xmpp-stanzas"/></error></iq
```

Step 8: Client to Server (Send presence information)

```
<presence id="nql1w-3"></presence>
```