Collaborative Project

ASPIRE

Advanced Sensors and lightweight Programmable middleware for Innovative Rfid Enterprise applications

FP7 Contract: ICT-*215417*-CP

WP3 – RFID Middleware Infrastructure

Public report - Deliverable

End-to-End Infrastructure Management

Due date of deliverable:	M38
Actual Submission date:	03.05.11

Deliverable ID:	**WP3/D3.5**
Deliverable Title:	End-to-End Infrastructure Management
Responsible partner:	UJF
Contributors:	John Soldatos – AIT Nikos Kefalakis – AIT Nektarios Leontiadis – AIT Didier Donsez - UJF Kiev Gama – UJF Gabriel Pedraza – UJF Jacky Estublier - UJF Bayu Anggorojati – AAU Simone Frattasi – AAU Neeli R. Prasad – AAU
Estimated Indicative Person Months:	24

Start Date of the Project: 1 January 2008 Duration: 42 Months

Revision: 1.7
Dissemination Level: PU

Document Information

Document Name: **End-to-End Infrastructure Management**
Document ID: **WP3/D3.5**
Revision: **1.7**
Revision Date: **3 May 2011**
Author: **UJF**
Security: **PU**

Approvals

	Name	Organization	Date	Visa
Coordinator	Neeli Rashmi Prasad	CTIF-AAU	03.05.2011	Approved
Technical Coordinator	John Soldatos	AIT	18.04.2011	Approved
Quality Manager	Anne Bisgaard Pors	CTIF-AAU	20.04.2011	Approved

Reviewers

Name	Organization	Date	Comments	Visa
Ramiro Robles	IT	28/03/2001	Consider comments	Approved

Document history

Revision	Date	Modification	Authors
0.1	23 July 2009	ToC	John Soldatos, Nikos Kefalakis
0.2	03 Sep 2009		Didier Donsez, Kiev Gama
0.3	Jan 2011	Merged with T3.6 internal proposal	Didier Donsez, Gabriel Pedraza, Kiev Gama
0.4	4 Feb 2011	Add security	Didier Donsez
0.5	10 Feb 2011	Added Section 4.1.1	Nikos Kefalakis, John Soldatos
0.6	17 Feb 2011	Added Section 5.4.7	Nektarios Leontiadis, Nikos Kefalakis
0.7	21 Feb 2011	Added Section 4.2	Nikos Kefalakis, John Soldatos
0.8	23 Feb 2011	Added section 2.3, Section 6	Nikos Kefalakis, John Soldatos
0.9	26 Feb 2011	Added Section 5.5 (Security and Privacy certification and criteria)	Sofyan M Yousuf Humberto Morán
0.9.1	26 Feb 2011	Annex II – detailed certification criteria	Sofyan M Yousuf Humberto Morán
1.0	28 Feb 2011	Corrections, Added Sections 4.2.5, 4.2.4.1.2	Nikos Kefalakis
1.1	1 Mar 2011	Added Sections 2.2.4 and 3.4	Bayu Anggorojati Simone Frattasi

			Neeli R. Prasad
1.2	17 Mar 2011	Augmented sections 3.1, 4.1.2,	
1.3	22 Mar 2011	Augmented introduction and conclusions	Gabriel Pedraza
1.3.2	23 Mar 2011	Added Section 5.4.7.4	Bayu Anggorojati
1.4	23 Mar 2011	Version to review	Didier Donsez
V1.5	04 Apr 2011	Changed/Added Executive Summary, corrections following internal review comments	John Soldatos, Nikos Kefalakis
V1.5.1	14 Apr 2011	corrections following internal review comments	Didier Donsez, Jacky Estublier
V1.6	18 Apr 2011	Finalization	Didier Donsez
V1.7	03.05.2011	Approved	Neeli Prasad

Content

Section 1 Executive Summary

RFID network and systems management is an essential function of any non trivial RFID deployment. RFID reader vendors provide software that manages the devices which they manufacture. However there are no widespread solutions for managing heterogynous RFID networks. Furthermore there are no OSS (Open Source Software) implementations of end-to-end management suites for RFID systems. One of the objectives of the ASPIRE project is to enable the end-to-end Management of complex and heterogynous RFID infrastructure. In order to reach this objective ASPIRE has specified and implemented a framework facilitating the administration of the elements (RFID readers, middleware modules) of an RFID network, as well as of their combination towards non trivial RFID systems. Based on this Framework ASPIRE has also provided a set of visual tools providing a User Friendly interface to RFID network Manager/Administrators. ASPIRE has provided OSS implementations of the framework and the tools as part of the AspireRFID project (http://wiki.aspire.ow2.org). This deliverable is devoted to the description of the framework and the tools.

The main elements of ASPIRE's end-to-end management framework are:

- A unifying distributed Architecture for end-to-end management enabling management of reader devices and middleware modules. Based on this Architecture RFID network managers can issue management commands to reader modules and middleware libraries from remote using distributed access technologies (such as Web-Services). The Architecture caters also for access control and security management, through a set of authentication and authorization mechanisms. Access control mechanisms can be used for privacy management (i.e. adapting the RFID network functionality to privacy constrains and preferences)
- A set of Mbeans (compliant to the JMX(Java Management Extensions) Architecture) which act as management proxies for the whole range of readers and middleware elements comprising an RFID network. Mbeans are easily amenable by tools, which has facilitated the implementation of graphical tools for executing the supported management commands. VisualVM plug-ins and Jasmine technologies have been exploited for the implementation of the tools. A set of Eclipse based "configurator" tools have also been implemented for managing the middleware modules of the ASPIRE Architecture (as these modules are specified in D2.3 and D2.4 of the ASPIRE project).
- Mechanisms for automated (Plug and Play) configuration of LLRP (Low Level Reader Protocol) compliant RFID Readers. Such readers can be configured without explicit intervention from an administrator.
- Tools and techniques for managing naming and directory services in line with the ONS (Object Name Service) specification of EPC global. Such management capabilities are particularly important for interenterprise RFID applications spanning multiple administrative domains and associated EPCIS (Electronic Product Code Information Service) repositories.

Overall ASPIRE has specified and implemented as OSS a range of useful functionalities for managing RFID devices, elements and networks in a scalable and unified way. Some of the implemented functionalities (e.g. the ONS inter enterprise directory services) are first of a kind implementations. The deliverable provides more details on all the above aspects of ASPIRE's end-to-end management framework.

Section 2 Introduction

2.1 The need for managing an RFID network

Complex applications require suitable techniques in order to be configured and monitored. Currently, management capabilities of RFID applications are very limited. The programming interfaces available for management of RFID applications are limited (or almost inexistent) and there is a lack of suitable management tools. Even if some RFID reader protocols such as RP (Reader Protocol) and LLRP (low level reader protocol) allow communicating with readers for performing configuration and also for retrieving information from the readers, there exists a lack of a higher level protocol as for example SNMP (Simple Network Management Protocols), which is used for managing devices in IP networks.

RFID middleware is composed of a set of physical and logical objects (readers, configurations, sensors, software modules), generally deployed on a LAN or WAN network. This architecture requires both a centralized and integrated access for configuring and monitoring of the RFID middleware as a whole, and a high level protocol for configuration of different middleware devices (readers, sensors, etc.). In addition, administration of IT infrastructures (such as RFID middleware) is today a complex and expensive task, technologies allowing auto-managed systems are leveraged in order to reduce the administration complexity.

Integrated access to application management must be provided by a centralized management application such as control panel. It allows system administrators to have a better perspective of the elements that compose the RFID network, and a single graphical interface for configuring and monitoring the objects that are part of that network.

In addition, it is important to have a common management protocol layer for sending administration commands to the different component and devices (readers, sensors, etc.) that compose middleware architecture. But, it is also important to introduce an intermediate layer for performing translations and adaptations of such administration commands into the appropriate protocol, be it at application level or at reader level.

Finally, introduction of techniques allowing auto-management of RFID middleware leverages a reduction of cost and complexity of administrators' tasks. These techniques go from how plug zero-configuration devices into the RFID architecture to the implementation of autonomic capabilities into the infrastructure.

2.2 Missing Pieces

2.2.1 Configurability

All components of the ASPIRE middleware should provide configuration capabilities in order to arrange in a suitable way all functional units of the middleware.

2.2.2 End to end Management

Aspire middleware consists of a set of components that should be managed and configured in a global way in order to obtain a suitable management vision. Currently, only a subset of the components of Aspire middleware exposes monitoring and management interfaces. Some of these components are managed by stand-alone tools.

It is necessary a unique management tool capable of federate a set of component-specific management tools and present them to the middleware administrator in an integrated interface in order to manage it.

2.2.3 Zero config RFID readers

Currently, in order to use a RFID Reader into RFID middleware (such as Aspire one) its driver must be installed previously, in addition the reader configuration must be registered using an LRSpec (in a EPC-Global vision). This operation is a real burden for the RFID solution installers especially in the context of small shops and small offices (SOHO). This process could take advantage of RFID readers that can join dynamically the network architecture and that can be taken into account dynamically by the RFID middleware (automatically or after validation by the installer). The introduction of a device profile for RFID readers (UPnP[1], DPWS)[2] could allow the exploitation of readers without a previous installation and configuration phase.

2.2.4 Security Management

Security management is a very critical requirement for a system like RFID middleware. Indeed, security management is a big task and covers big area of security. Therefore, apart from the implementation specific of all security aspects in the system, we will focus on the authentication and access control in the context of RFID middleware.

An RFID middleware should at least be able to manage user credentials, e.g. add, modify, and delete user credential such as basic username and password, client certificate, and so on. This capability could be realized by means of a management console. Furthermore, management framework in an RFID middleware should be able to collect information and give notification upon any event related to security or access to middleware, such as notification of (un)successful authentication of client. Other important functionality is notification of mal-usage of RFID middleware by any unauthorized client or client that is not supposed to access a certain resource in RFID middleware.

2.2.5 Autonomic Management

Since IT infrastructures (such as RFID middleware) grow rapidly in complexity of systems management, autonomic management aims to reduce it and its cost (TCO) by alleviate the system administrator from basic tasks. Autonomic management aim is to develop computer systems capable of self-management such as self-configuration, self-healing, self-tuning, etc. [50].

OW2 JASMINe is an autonomic manager for the SOA[3] platform aiming to alleviate the administrator's job and reduce the management costs. Jasmine targets clusters of JavaEE servers and/or OSGi platforms. System sensors and actuators are JMX MBeans and the control loop is a set of rules written in Drools[4].

2.3 Supply Chain Management Using RFID Middleware

Since industry wide application is by nature distributed, and the amount of information generated is impossible to store and manage centrally – or locally, being distributed implies that often there is no prior knowledge of location or available interfaces between all partners, and such information has to be discovered Figure 1. This raises the need for an 'Object Loop-up Service'. This is an incarnation of the directory service found in all distributed systems, this time associating object names (the unique RFID tag the object is carrying) with

[1] Universal Plug and Play [29]
[2] OASIS Device Profile for Web Services, http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01
[3] Service Oriented Architecture
[4] http://www.jboss.org/drools

sources of information regarding the particular object. EPCGlobal does provide the specification for such a service called Object Naming Service (ONS) [49], along with its position in the overall EPCGlobal framework and the interaction with the rest of the components. At the moment there are only a few implementations of the service, and all of them are commercial and proprietary.

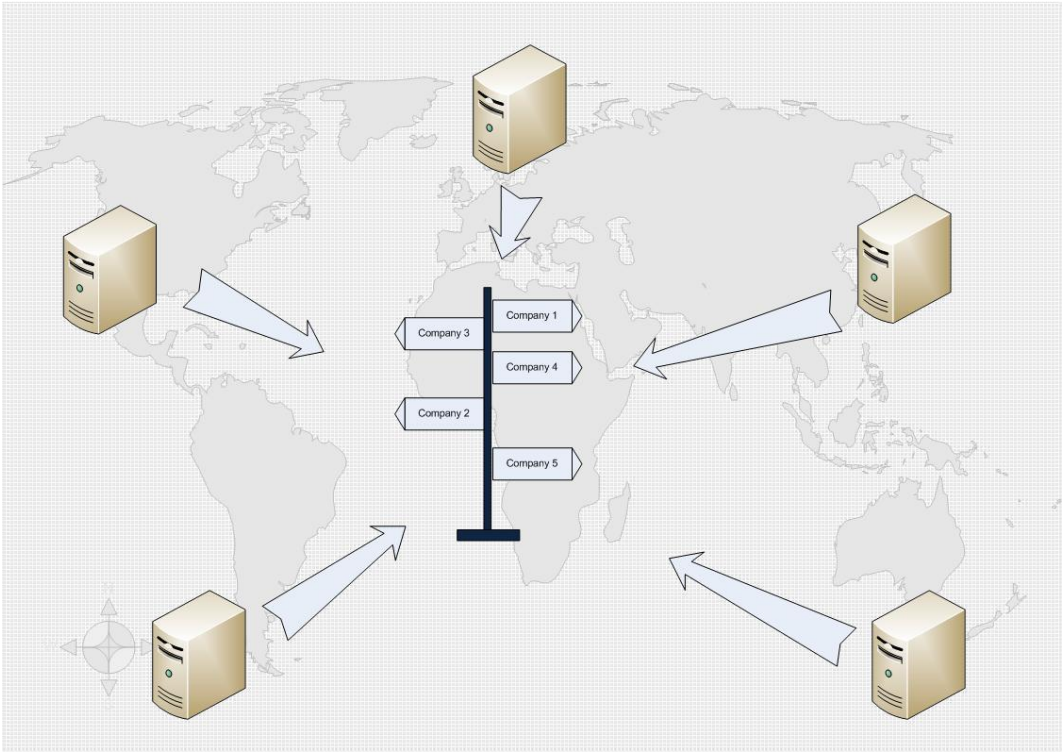

Figure 1 Inter-enterprise applications pose the challenge of discovery additional partners

ASPIRE introduce an open source implementation of a distributed look-up / service for RFID tags. The implementation lends itself on the ONS standards and provides the core directory service for EPCIS accessing applications, as defined in the EPCGlobal framework. Using the ONS server, enterprise application can discover and communicate to or retrieve data from partner applications running on foreign domains. Besides the core server we provide an API (in Java) that allows application to perform lookup for EPC tags on the ONS server. While the server, and the API is based on the standard proposed by EPCGlobal, both can be easily adapted to serve as basis for directory services in the Internet of Things era of applications.

Along with the implementation ASPIRE introduce a representative application that manifests the concept and the added-value of inter-enterprise applications. The application allows the automated tracking of company products, as they move between business partners. Each partner has an already established RFID infrastructure (an EPCIS repository as defined by EPCGlobal standards) and the application works in tandem with it as an add-on. Functionality wise, for every item entering the company domain, it uses the ONS server to find an appropriate endpoint and report that information to the item owner. Naturally it also collects data reported by other business partners for the company's own products that enter their domains. Finally, an interface is provided for accessing applications to request all available data about a specific EPC tag. Data about a specific tag can be plotted on a map presenting a visual course and history of the product in question. In Section 6 we describe the Directory Server Architecture and its Implementation and the Tracking Application and Interfaces. Finally at the same section we describe the presentation tier and provide a sample application.

Section 3 Enabling Management technologies

Aspire middleware has been built using the Java technology and a big portion of it the OSGi technology as execution platform for advanced services (sensors, dynamic readers, etc.). Moreover the whole middleware architecture proposes interfaces using SOAP Web services technology following a Service Oriented Architecture.

JMX is the de facto framework for providing management capabilities for Java applications. Web services technology is an alternative to JMX, providing interoperability advantages because it is platform agnostic, not depending on implementation technology.

3.1 JMX

Java Management Extensions (JMX) [15] is a Java technology for the management and monitoring of Java applications. System objects, or reified external physical objects like printers, sensors, RFID readers, etc., can be represented in the system as a Manageable Bean (MBean). These MBean objects would expose functionality (e.g. reading/writing a property, invoking a method) that can be remotely called via JMX, enabling the instrumentation of the system objects and resources at runtime.

The functionality achieved with JMX is similar to what SNMP (Simple Network Management Protocols) provides for the management and monitoring of network equipment.
As depicted the Figure 2,The JMX architecture has basically three levels:

- The Probe level, which are constituted by the MBeans previously described for accessing the resources.
- The Agent level, that corresponds to an MBeanServer of the target application. MBeans are registered in the MBeanServer which acts as an intermediary between external management applications and the MBean instances.
- The Remote Management level, which provides connectors and adaptors implemented using different types of object communication technologies (RMI, JMS, WebServices[5], HTTP, SMTP, SNMP, XMPP, … etc). Management applications (such as administration consoles) should use these connectors for accessing a Java application MBeanServer.

[5] JSR 262: Web Services Connector for Java Management Extensions (JMX) Agents http://jcp.org/en/jsr/detail?id=262 https://ws-jmx-connector.dev.java.net

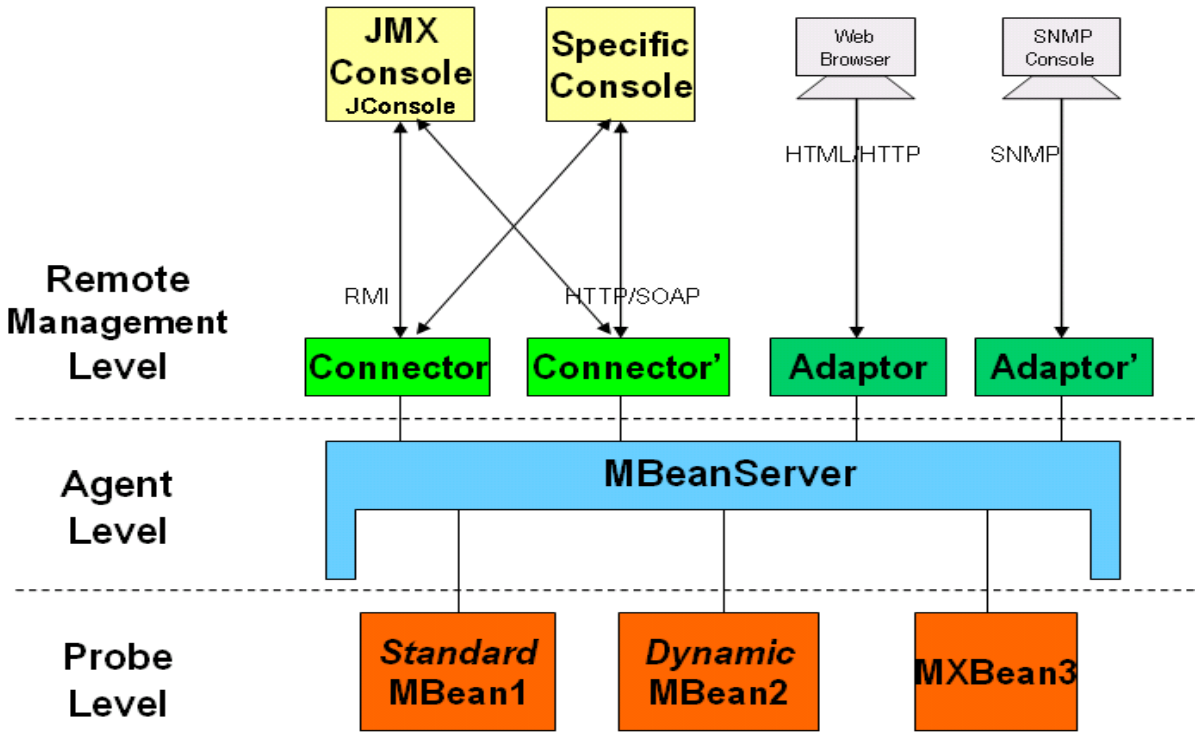

Figure 2 the JMX architecture (from Wikipedia, D. Donsez (contributor)).

By developing custom JMX probes (MBeans) and installing them in a Java application, management interfaces can be made available to external applications. The usage of JMX technology is seamless in Java applications, since JMX support is built-in. Standardized MBeans has be proposed for the management of JavaEE servers (JSR 77), OSGi platforms (RFC 139[6])

An MBeanServer instance interacts with the MBeans registered with it via their ObjectNames. In order to register an MBean with an MBeanServer, both the MBean object instance and the corresponding MBean ObjectName instance are passed as parameters to the registration method.
The object name is an instance of the JMX class ObjectName, and must conform to the syntax defined by the JMX specification, namely it must comprise a domain, and a list of key-properties. For example, an MBean proposing access to configuration of the ECSpec module into ALE server component should be named as follows: *"org.ow2.aspirerfid:type=aleserver,name=ecspec"*.

Currently some generic consoles are proposed in order to manage applications through JMX technology. We can find management consoles as Eclipse plugins, (Figure 2,) but the most used ones are those shipped with the Oracle Java Development Kit (formerly Sun JDK): JConsole and VisualVM (Figure 4). The VisualVM [28] is a visual tool that instruments Java Virtual Machines (JVM), providing different types of information at the VM level and making available default profilers and data visualizers. The VisualVM supersedes JConsole, (JDK's previous monitoring console generation), being extensible through a plug-in mechanism since it is built on top of the Netbeans platform. Plug-ins that extends VisualVM functionality easily integrate to the instrumenting infrastructure provided by that platform. No additional components or connection establishing is necessary to access JMX (Java Management Extensions) functionality since the VisualVM API allows plugins to retrieve the JXM connection object that gives access to the instrumented JVM. Figure 4 illustrates the

[6] http://www.osgi.org/javadoc/r4v42/org/osgi/jmx/framework/package-summary.html

VisualVM with a custom plugin that provides a management console to OSGi platforms [39], developed in the context of the Aspire project enables to automatically deployed on a monitored OSGi platform, the probes (i.e. MBeans) required by the plugins.

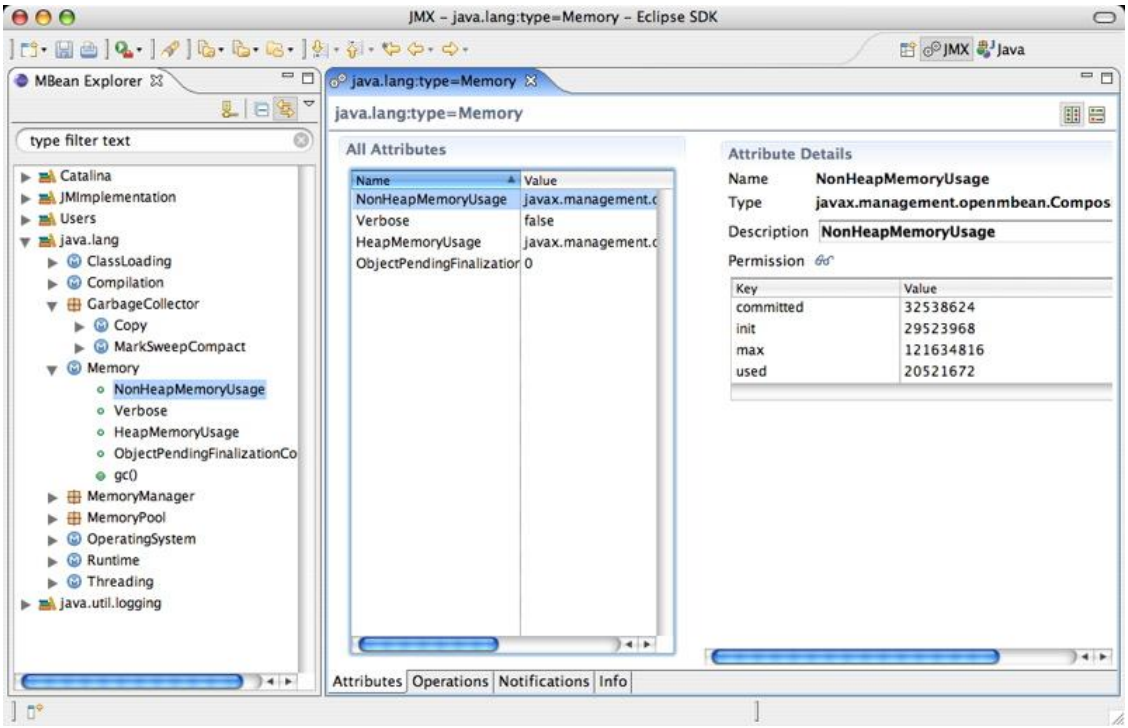

Figure 3 JMX Eclipse Console

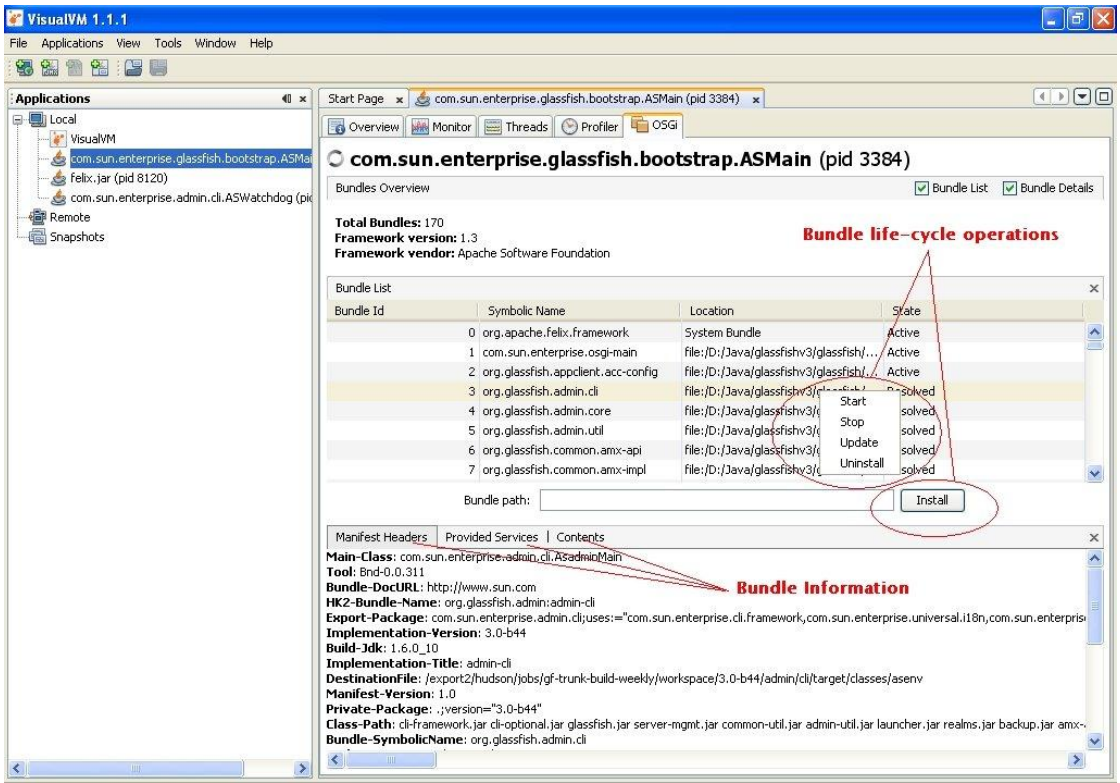

Figure 4 VisualVM Console inspecting an OSGi platform

3.2 WebServices

Web services are software applications that enable the communication of machines over a network, similar to remote procedure calls. Since they are built using web technologies like HTTP and XML, Web Services have been a good alternative for the communication of heterogeneous applications. The exchange of messages is text-based (XML) which implies in a high communication overhead if compared to binary formats.

A common utilization of Web Services for integration is wrapping existing functionality in a system and exposing it as a Web Service. Due to its loosely coupled nature, Web Services is commonly used in architectural approaches like Service Oriented Architecture (SOA).

The web services technology has not been created to be used as a management technology, however Distributed Management Task Force (DMTF)[7] promotes the Web-Based Enterprise Management (WBEM)[8] which is a set of systems management technologies based on Web Services technologies, to unify the management of distributed computing environments.

3.3 UPnP (Universal Plug and Play)

UPnP and its successor DPWS[9] are distributed, open architecture protocols based on established standards such as the Internet Protocol Suite (TCP/IP), HTTP, XML, and SOAP. Those technologies allow the establishment of devices profiles, for example profiles for Printers, Internet Gateways, Media Servers, Security Camera, HVAC[10] appliances etc. The main idea behind these technologies is to leverage a Plug-and-Play architecture, where devices that join a network can be discovered automatically and be used by other devices without a previous configuration phase. UPnP is now the de-facto standard for SOHO (Small Office Small Home) appliances, even some alternatives exist such as IGRS[11] promoted by the Chinese manufacturers and EchoNet[12] by Japanese ones. Moreover, "Device Management"13 a recent specification of the UPnP Forum, addresses the remote configuration and the software maintenance of SOHO appliances by remote operators.

3.4 Security

Security management in the context of AspireRFID middleware is related to authentication and authorization, as well as access control to each AspireRFID middleware module. Please notice that security as well as privacy are very broad issues, which may involve different components in the whole system and most likely to be system dependent or implementation specific, e.g. network security, server security configuration (Tomcat, Felix, etc), client side security, etc. Therefore we are going to focus on security in the AspireRFID middleware specification as a platform, and try to avoid any dependency as much as possible, e.g. independent of deployment server type.

3.4.1 Authentication and authorization

AspireRFID middleware uses Web Service technology as its communication interface between middleware and client application, as well as between different modules in the

[7] http://www.dmtf.org/
[8] http://www.dmtf.org/standards/wbem
[9] OASIS Device Profile for Web Services, http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01
[10] Heat, Ventilation and Air Conditioning http://upnp.org/specs/ha/hvac/
[11] Intelligent Grouping and Resource Sharing http://www.igrs.org/
[12] http://www.echonet.gr.jp/english/index.htm
[13] http://upnp.org/specs/dm/dm1/

middleware itself. By considering such specification, authentication and authorization in the AspireRFID middleware context is based on Web Service security specification. By having this, we can avoid dependency with different security requirements and configurations in different deployment server, e.g. Tomcat or Felix server.

In general, authentication and authorization of Web Service technology consists of two methods, namely HTTP authentication and SOAP message embedded authentication. The first method requires security configuration in the deployment server of middleware as mentioned earlier, therefore this method is out of the middleware scope. On the other hand, the second method makes use of SOAP message to carry authentication credentials from the challenger (in this case middleware's client is the challenger), and uses another SOAP message in response to authentication results. This results in more flexible authentication implementation in AspireRFID middleware which uses Web Service technology, although the real implementation might require dependencies from the implemented web service framework in the system, i.e. CXF[14], Axis2[15], or Metro[16], which is left for the developer to decide which framework to be used.

The implementation of authentication and authorization in AspireRFID middleware follows the specification of CXF as the implemented web service framework. Furthermore, AspireRFID middleware supports basic username token and user certificate authentication, which again will be left to the system implementer of AspireRFID middleware.

3.4.2 Access control

Access control in AspireRFID middleware is implemented on each available middleware module, e.g. ALE server, EPCIS repository, and BEG. Particularly in the ALE server, the access control implementation is done based on EPCglobal specification of ALE server. Moreover, access control implementation in the other middleware's modules is adapted from the EPCglobal specification of access control in ALE server.

- Referring to EPCglobal specification, access control consists of four main classes, namely: ACClientCredential, ACClientIdentity, ACRole, and ACPermission. Short description of each above mentioned class is as follow:
- ACClientCredential: manages all clients' credentials.
- ACClientIdentity: manages identity of client and map them with particular roles.
- ACRole: manages all roles that can access the middleware and map them with particular permissions.
- ACPermission: manages all kind of permissions to be accessed in the middleware

The first bullet point above is highly related to the authentication and authorization that was discussed in the previous sub-section. In order to ease the development and testing process of access control in particular, and middleware in general, only basic username token has been implemented in the current access control development.

Simple illustration of Access Control in AspireRFID middleware is depicted in Figure 5. The top part of Figure 5 shows the Middleware Access Control configuration done by System Admin, by assuming that System Admin has permission to access the Access Control API in the middleware and does any operation related to access control. Furthermore, System Admin is able to do all the operations related to ACClientCredential, ACClientIdentity, ACRole, and ACPermission, as defined in the EPCglobal specification, such as define,

[14] http://cxf.apache.org/
[15] http://axis.apache.org/axis2/java/core/
[16] http://metro.java.net/

update, undefined, delete, etc. The access control configuration then will be stored in the Access Control repository once System Admin finishes with the configuration.

The bottom part of Figure 5 illustrates how the Client Application will access any resource in any middleware's module through Access Control mechanism. Resources in the middleware's module can be a certain method, API, or the module itself, which is defined by System Admin in the configuration process. The process that involves Client Application in accessing middleware's module as illustrated in Figure 5 is briefly explained as follows:

1. Client application asks Access Control module to access a certain resource in a Middleware's module.
2. If the authentication of client application is successful, the Access Control will check its credentials with its assigned role and its permission to access the resource in the Access Control repository.
3. Access Control module receives the result from point 2.
4. Access Control module sends the response upon the operation in point 1 to Client application.
5. Client application will be able to access a resource in the middleware module if it receives positive access control result.

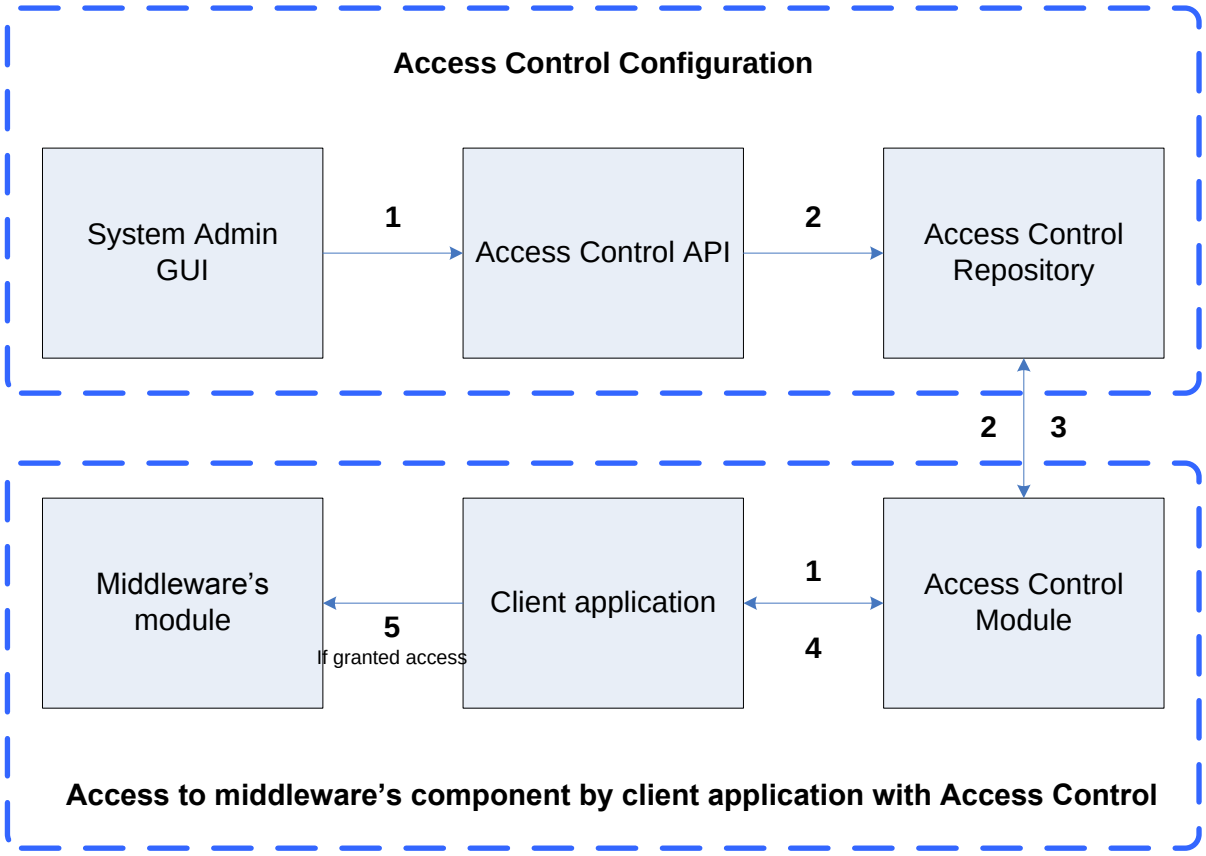

Figure 5: Simple illustration of access control in AspireRFID middleware

Section 4 Classic ASPIRE End to End management Architecture

4.1 RFID Reader Management

The existence of interfaces that enable fine grained application control during runtime by changing state and calling operations of system objects (e.g. readers, middleware modules, sensors) allows the construction of applications such as a control panel. Such sort of application where all system objects and configuration could be centralized would significantly simplicity application management.

4.1.1 Autonomic configuration of LLRP Readers

One of the most important and crucial part of RFID technology is the configuration and management of RFID readers. Readers are operating based on their settings. Each reader prior to its use is set up according to the needs of the information system within it is used. For example, a reader in a supply chain management system should be properly configured so as to identify and record the specific information the company wants.

Thus, characteristics like the reader's manufacturer, the maximum number of antennas it can support, its IP address and port number are all very important. A bad configuration (i.e. use of wrong values to attributes like the aforementioned) may lead to unexpected and bad results.

Previously, due to settings' importance, the readers' configuration was made manually for every single reader being part of the system. However, we can understand that in systems having a lot of readers it can become extremely time consuming.

The current work aims at filling this "gap". An application was created that is doing the aforementioned configuration automatically for all the LLRP readers being in a sub-network. The user still has to provide some necessary information (e.g. the range of readers' IP addresses) and leave the rest to our application. Such an application may be useful in the progress of AspireRFID project.

4.1.1.1 Low Level Reader Protocol (LLRP)

LLRP specifies an interface between RFID readers and clients. Under this protocol, the user can interact with an RFID reader: query the reader for its characteristics, set up the reader or change some of its settings.

All of the above are done by exchanging the so-called "LLRP messages". Those messages have a specific format and depending on the values of their attributes can ask information from the reader or configure the reader.

4.1.1.2 Automatic Reader Management – Design and Specification

The application aims at providing a convenient way of automatically configuring all the RFID readers being in a network. The first step is to query for the present RFID readers in our network. Every reader in our network will receive a specific LLRP message asking for its characteristics. Based on the data we are going to receive, we create a new Logical Reader Specification, the so called LRSpec. For every LRSpec we create, we should inform the Filter and Collection Layer, which is responsible for keeping the system's readers in its records. The above procedure should be done periodically so as to keep an up to date record of the readers being in the network every moment.

4.1.1.3 System Implementation

The whole application is written in Java. It is obvious though, that in order to succeed in communicating with the readers and exchanging messages with them we had to make use of some “extra” libraries containing useful classes and methods.

The main class of our application is in “*SearchReaders.java*”. The application initializes by “reading” the necessary configuration XML file. The file parsing is done using JDOM[17] library which provides helpful methods for identifying specific tags in an XML file and get their values make thinks simpler. This file contains the range of IP addresses in which the application has to search for RFID readers as well as some important constant values needed in the rest of the program.

It would be extremely convenient and more efficient in terms of resource management if we had the opportunity to broadcast a message to all possible recipients. In order to use multicasting in Java, the recipients must be registered in a group, but this is not supported from LLRP RFID readers. As a consequence of that, our application spawns one thread for every single IP address in the range provided. To be more specific, our application starts one instance of the “*ReaderConfig.java*” for every IP address.

After establishing a socket connection with the reader, the application uses the LLRP toolkit [29] in order to send and receive LLRP messages from the reader, depicted in Figure 6 below. So, by using the aforementioned toolkit we are able to create an LLRP message at first. Thus, we create a “GET_READER_CAPABILITIES” LLRP message and send it via the output stream of the socket connection. Once the reader receives such a message, it responds with an additional LLRP message containing its characteristics such as its manufacturer name, the maximum number of antennas it can support etc. In order to be able to receive and read messages from the reader, once we send the LLRP message, we spawn a “*ReadThread. Java*” thread. Its goal is to set up the connection’s input stream and start waiting for new messages to be received.

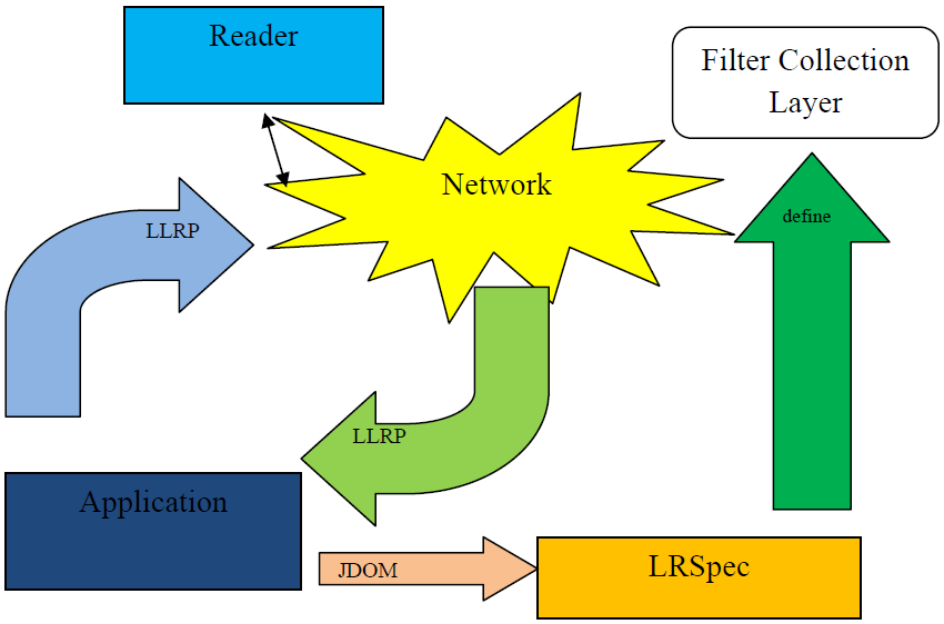

Figure 6 Automatic LLRP Reader Configuration lifecycle

[17] http://www.jdom.org/

Once that message is received, only the useful data are selected. In order to do so, we firstly transform the message to an XML string. Then, by using JDOM library we are parsing through the string and collect only the information we want. In our case, we are interested in the manufacturer's name and the maximum number of antennas the reader can support. It is easily understood, that the data we collect from the message depend on the system's and reader's characteristics. Thus, if the system's needs require additional or other data, we can just edit this part.

In the next step, we have to create a Logical Reader Specification (LRSpec). By using the data we collected in the previous step as well as some of the data provided in the configuration XML file we create the corresponding LRSpec. This is done by using "*LRSpecBuilder.java*" class. Once we have created the LRSpec, we are ready to inform the Filter and Collection Layer for its existence. This is done by "defining" the LRSpec to the Filtering and Collection Layer.

The above procedure is repeated in a preconfigured interval of time e.g. 3-5 minutes so as for the Filtering and Collection Layer to be up to date to any changes that may have arisen in the mean time.

4.1.1.4 Testing

At the testing phase our application gave us good results. There was no problem with the reader communication or with the messaging exchanging. The only disadvantage was the fact that due to the "problem" we mentioned regarding multicasting we initially spawn a lot of useless threads, especially if the readers' IP's are not consequent. This can somehow harm our system's efficiency.

4.1.2 Adhoc discovery of readers

Several protocols [48] have be developed and standardized to discover appliances in local networks such as home and warehouse area networks. Most popular ones are DNS-SD[18] (aka mDNS, Rendez Vous, Apple' Bonjour), UPnP[29] and DPWS. Since most of professional-range RFID readers are connected thru IP over Ethernet and WiFi, reader discovery features will alleviate the configuration burden of readers scattered across the network..

4.2 ASPIRE Middleware Modules Web Service Management/Configuration

The Aspire middleware, see Figure 7 below, has adopted the EPC Global Framework as the reference architecture and used the Fosstrak [1] project as the base implementation for the Reader, Filtering & Collection and EPCIS modules. EPC framework described various API's for an RFID middleware configuration and management. The most important ones, which are also used from the ASPIRE implementation are the following:

[18] http://www.dns-sd.org/

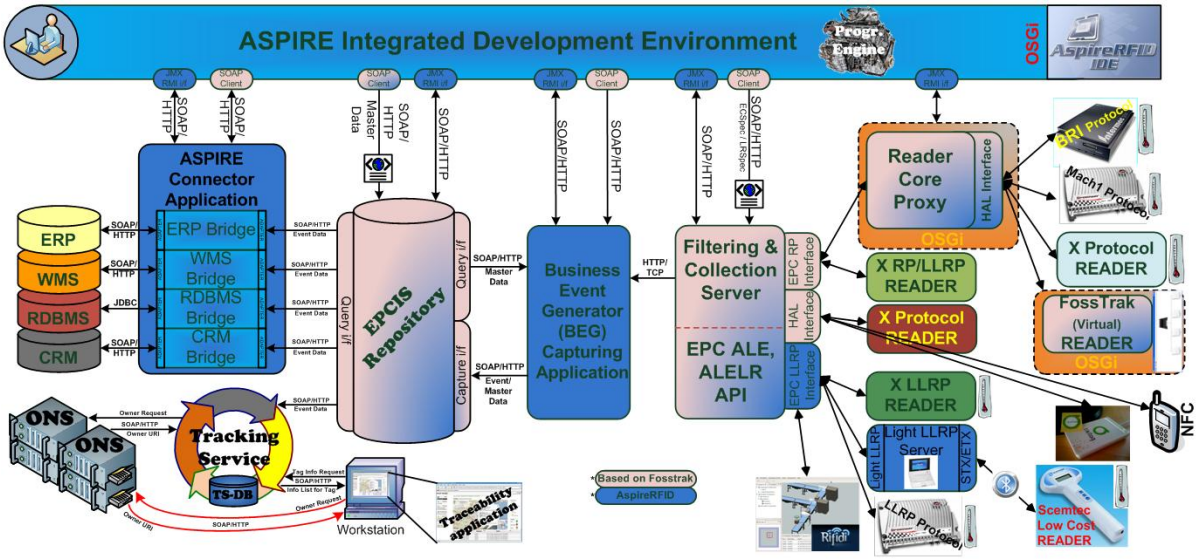

Figure 7 ASPIRE Architecture

4.2.1 ALE Reading API (Fosstrak Implementation)

The Filtering & Collection Interface (ALE), see Figure 8 below, provides a standard interface to the Filtering & Collection role that applies to a large collection of use cases in which RFID Tags are inventoried (i.e., where the EPCs carried on the tags are read).

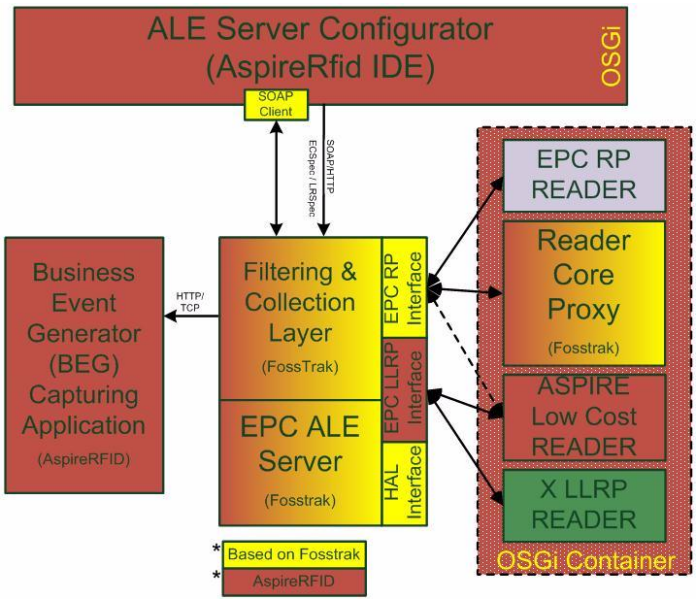

Figure 8 ALE & ALELR Interfaces [2]

Responsibilities:

- Provides means for one or more client applications to request EPC data from one or more Tag sources.
- Provides means for one or more client applications to request that a set of operations be carried out on Tags accessible to one or more Tag sources. Such operations including writing, locking, and killing.
- Insulates client applications from knowing how many readers/antennas, and what makes and models of readers are deployed to constitute a single, logical Tag source.

- Provides declarative means for client applications to specify what processing to perform on EPC data, including filtering, aggregation, grouping, counting, and differential analysis.
- Provides a means for client applications to request data or operations on demand (synchronous response) or as a standing request (asynchronous response).
- Provides means for multiple client applications to share data from the same reader or readers, or to share readers' access to Tags for carrying out other operations, without prior coordination between the applications.
- Provides a standardized representation for client requests for EPC data and operations, and a standardized representation for reporting filtered, collected EPC data and the results of completed operations.

The ALE API methods are shown in Table 1below.

<<interface>> ALE
---
define(specName : String, spec : ECSpec) : void
undefine(specName : String) : void
getECSpec(specName : String) : ECSpec
getECSpecNames() : List<String>
subscribe(specName : String, notificationURI : String) : void
unsubscribe(specName : String, notificationURI : String) : void
poll(specName : String) : ECReports
immediate(spec : ECSpec) : ECReports
getSubscribers(specName : String) : List<String>
getStandardVersion() : String
getVendorVersion() : String
<<extension point>>

Table 1 EPC ALE Interface [2]

4.2.2 ALE Logical Reader API (Fosstrak Implementation)

The Logical Reader API (ALELR), see Figure 8 above, provides:

- A standardized way for an ALE client to define a new logical reader name as an alias for one or more other logical reader names
- For manipulating "properties" (name/value pairs) associated with a logical reader name
- A means for a client to get a list of all of the logical reader names that are available, and to learn certain information about each logical reader

In summary, there are three ways that logical readers may come into existence:

- Composite Reader: A composite reader is a logical reader that is defined as an alias for other logical reader names
- Externally-defined Base Reader: An externally-defined base reader is a logical reader that is an actual channel for manipulating Tags

- API-defined Base Reader: An API-defined base reader is a logical reader that is an actual channel for manipulating Tags

The ALELR API methods are shown in Table 2 below.

<<interface>> ALELR
---
define(name : String, spec : LRSpec) : void
update(name : String, spec : LRSpec) : void
undefine(name : String) : void
getLogicalReaderNames() : List<String>
getLRSpec(name : String) : LRSpec
addReaders(name : String, readers : List<String>) : void
setReaders(name : String, readers : List<String>) : void
removeReaders(name : String, readers : List<String>) : void
setProperties(name : String, properties : List<LRProperty>) : void
getPropertyValue(name : String, propertyName : String) : String
getStandardVersion() : String
getVendorVersion() : String
<<extension point>>

Table 2 EPC ALELR Interface [2]

4.2.3 BEG API

The role of the BEG (Business Event Generator), depicted in Figure 9 below, is to automate the mapping between reports stemming from F&C and IS events. Instead of requiring developers to implement the mapping logic, the BEG enables application builders to configure the mapping based on the semantics of the RFID application.

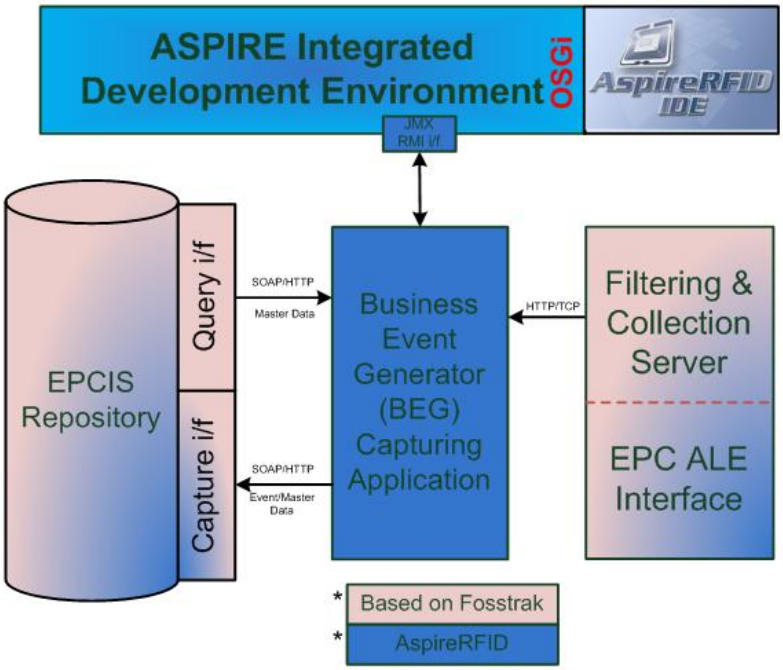

Figure 9 BEG Interfaces

Responsibilities:

- Recognizes the occurrence of EPC-related business events, and delivers these as EPCIS data.
- May coordinate multiple sources of data in the course of recognizing an individual EPCIS event. Sources of data may include filtered, collected EPC data obtained through the Filtering & Collection Interface, other device-generated data such as barcode data, human input, and data gathered from other software systems.
- May control the carrying out of actions in the physical environment, including writing RFID tags and controlling other devices. (When tag writing and related features are addressed in a future version of the Filtering & Collection Interface, the EPCIS Capturing Application may use the Filtering & Collection Interface to carry out some of these responsibilities.)

The BEG component interface provides five methods for interaction with the BEG client which are all communicating with the BEG client by exchanging SOAP messages.

- The first method is the `getEpcListForEvent (EventStatus getEpcListForEvent(String eventID))` which is used for returning to the BEG client an `EventStatus` object which contains the real time list of EPC ids and the transaction ID of a chosen Event (String eventID) from the list of events that the BEG component is already serving. So with the help of this method one can observe at real time the incoming IDs as they are reported to the BEG by the F&C component and are related with a specific transaction Event.
- The second method is the `stopBegForEvent (boolean stopBegForEvent(String eventID))` which is used by the BEG client to stop serving a predefined `Event` by sending to it its specific `EventID`.
- The third method is the `getStartedEvents (List<String> getStartedEvents())` which returns a list of Event IDs that the BEG component is serving.
- The fourth method is the `startBegForEvent (boolean startBegForEvent(VocabularyElementType vocabularyElementType, String repositoryCaptureURL, String begListeningPort))` which is

used to set up the BEG component for start serving a specific Event. More specifically this method takes the already pre described Elementary Business Transaction Event described at the Information Sharing repository's Master Data and uses it for configuring the Business Event Generator to create Business Events from the ECReports received from the port given as variable to the `startBegForEvent` method. If the method is successful it will return `true` otherwise it will return `false`.

- Finally, the fifth method is the `getEventList (List<VocabularyElementType> getEventList( String repositoryQueryURL))` which is used for returning a list of all the available defined Events from a Company's EPCIS Master Data repository.

4.2.4 EPCIS API

4.2.4.1 EPCIS Capture Interface

4.2.4.1.1 Event Data Capture Interface

The EPCIS Capture Interface, see Figure 10 below, responsibilities is to provide a path for communicating EPCIS events generated by EPCIS Capturing Applications to other roles that require them, including EPCIS Repositories, internal EPCIS Accessing Applications, and Partner EPCIS Accessing Applications.

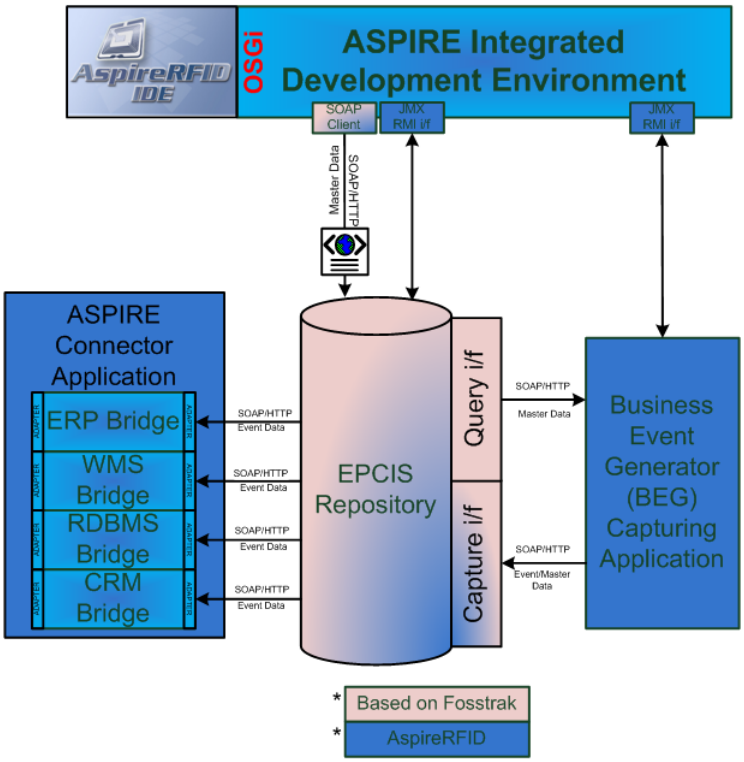

Figure 10 EPCIS Interfaces

4.2.4.1.2 Master Data Capture Interface

Additionally to the EPC defined Event Data capture interface a new EPCIS interface was defined and implemented from ASPIRE for capturing Master Data information. This new interface supports eleven control commands which are briefly described in the followinglist:

1. **alterVocElem** (String vocabularyType, String oldVocabularyElementURI, String newVocabularyElementURI)

- Which is used to alter a vocabulary's Element URI.

2. **insertVocElem** (String vocabularyType, String vocabularyElementURI)
   - Which is used to insert a vocabulary's Element.
3. **massInsertVocElem** (String vocabularyType, ArrayList<String> vocabularyElementURIs)
   - Which is used to insert many vocabulary's Elements.
4. **deleteVocElem** (String vocabularyType, String vocabularyElementURI)
   - Which is used to delete a vocabulary's Element (When using single delete only the element with its attributes will be deleted).
5. **massDeleteVocElem** (String vocabularyType, ArrayList<String> vocabularyElementURIs)
   - Which is used to delete many vocabulary's Elements (When using mass delete only the listed elements with their attributes will be deleted).
6. **deleteWithDescendantsVocElem** (String vocabularyType, String vocabularyElementURI)
   - Which is used to delete a vocabulary's Element with its direct or indirect descendants (The element with its attributes and with all of its children elements and its children's attributes will be deleted).
7. **massDeleteWithDescendantsVocElem** (String vocabularyType, ArrayList<String> vocabularyElementURIs)
   - Which is used to delete many vocabulary's with their Elements and with their direct or indirect descendants. The elements with its attributes and with all of its children elements and its children's attributes will be deleted.
8. **insertOrAlterVocElemAttr** (String vocabularyType, String vocabularyElementURI, String vocabularyAttributeName, String vocabularyAttributeValue)
   - Which is used to insert or Update a vocabulary Element's Attribute. If the Vocabulary is not inserted yet it will be inserted. The vocabularyElementURI, vocabularyAttributeName pair should be unique so if it already exists it will be changed to the vocabularyAttribute entered or simply rewrite it.
9. **massInsertOrAlterVocElemAttr** (String vocabularyType, String vocabularyElementURI, HashMap<String, String> vocabularyAttributes)
   - Which is used to mass Insert or Update a vocabulary Element's Attributes. If the Vocabulary is not inserted yet it will be inserted. The vocabularyElementURI, vocabularyAttributeName pair (the vocabularyAttributeName is the key of the vocabularyAttributes HashMap) should be unique so if it already exists it will be changed to the vocabularyAttributeValue (which is the value of the vocabularyAttributes HashMap) entered or simply rewrite it.
10. **deleteVocElemAttr** (String vocabularyType, String vocabularyElementURI, String vocabularyAttributeName)
    - Which is used to delete a vocabulary Element's Attribute.
11. **massDeleteVocElemAttr** (String vocabularyType, String vocabularyElementURI, ArrayList<String> vocabularyAttributeNames)
    - Which is used to delete many vocabulary Element's Attributes.

The above work was contributed to the Fosstrak project and it is available online at http://www.fosstrak.org/epcis.

4.2.4.2 EPCIS Query Interface (Fosstrak Implementation)

The EPCIS Query Interface, see Figure 10 above, responsibilities are to provide:

- Means whereby an EPCIS Accessing Application can request EPCIS data from an EPCIS Repository or an EPCIS Capturing Application, and the means by which the result is returned.

- Means for mutual authentication of the two parties.
- The result of authorization decisions taken by the providing party, which may include denying a request made by the requesting party, or limiting the scope of data that is delivered in response.

4.2.5 Programmable Engine API

The programmable engine exposes two Interfaces. The first one allows the Clients to "encode" the AspireRFID middleware with the use of an APDL document. And the second one gives the ability to the client to "decode" and retrieve an already configured APDL document from the AspireRFID middleware by giving the ID of the Parent Business Process.

4.2.5.1 Encode API

The First Interface offered by the Programmable Engine implementation is the Encode interface. The Encode API has one method that requires as input the APDL XML document with the whole RFID Business Process description (see Table 3 below). This method returns an integer code that indicates whether the execution of this specific command is successful or not. Within this method the PE configures a running instance of the middleware, such instance implementing the Business Processes described by the given OLCBProc (Open Loop Composite Business Process) Object.

Method	Argument/Result	Type	Description
encode	openLoopCBProc	OLCBProc	This method configures the AspireRFID middleware to serve the described Business Processes within the given APDL XML document. If the encode is successful the reply ID will be "400" if not the reply ID will be "425".
	[result]	Integer	

Table 3 PE's Encode Interface methods

The primary Datatypes associated with the PE Encode API are the "Integer" type, which denotes the success id of the "encode" execution, and the "OLCBProc" type which contains the APDL object and the Business Process Description.

4.2.5.2 Decode API

The second Interface offered by the Programmable Engine is the Decode Interface. The Decode API has one method "decode" that requires as input a String of an OLCBProc ID originally used to configure a middleware instance (see Table 4 below). This service is for being used by clients to alter a previously encoded Business Process by retrieving it ("decode"), making the required changes, and then reconfiguring back the middleware ("encode").

Method	Argument/Result	Type	Description
decode	openLoopCBProcID	String	This method returns an OLCBProc Object which is retrieved from an AspireRFID middleware running instance by a prior configured (encoded) OLCBProc by giving that object's ID.
	[result]	OLCBProc	

Table 4 PE's Decode Interface methods

The primary Datatypes associated with the PE are the "*String*" type, which is the ID of a prior configured OLCBProc, and the OLCBProc type which is constricted and returned from the decode service.

Section 5 Homogenizing the ASPIRE Management Environment

As the AspireRFID existing middleware components are all based on Java technology, Java Management Extentions (JMX) are a reasonable choice for providing management functionality.
The architecture of Aspire's End to End management would consist of basically three layers as depicted in Figure 11, which provides a high level perspective. A management console provides a Graphical User Interface (GUI) which administration functionality that accesses the manageable objects through probes that must be under a common Aspire Management API.

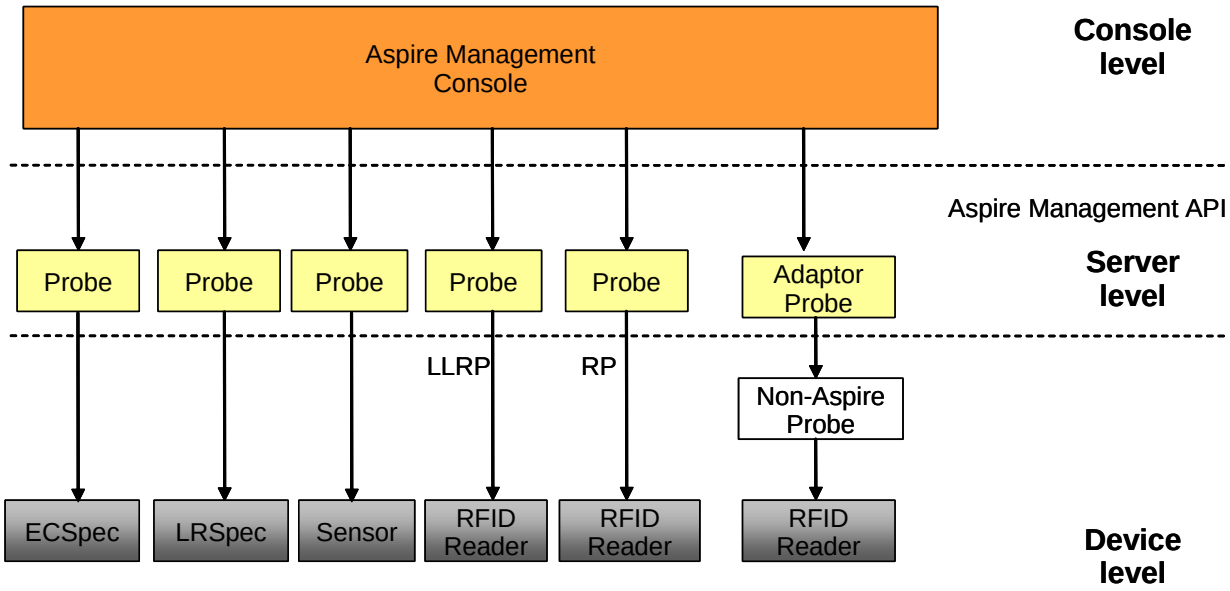

Figure 11 AspireRFID End-to-End Management Architecture

In the realization of the proposed architecture, we use the uniform management API based on JMX MBeans that provide access to the manageable resources. The VisualVM is used as a management platform, where the customization is introduced in the form of VisualVM plugins that provide specific AspireRFID functionality.

The different modules of the Aspire middleware can provide manageable functionality separately. It is only necessary to provide MBeans that comply with the requirements specified in this document.

New elements adapted to the system (e.g. a new RFID reader) would have to provide an implementation of the specified MBean interface so it could be controlled through the management console. If an RFID reader that already presents a management interface (e.g. a web services interface) is to be integrated to Aspire, an adaptor MBean would do the role of delegating the JMX calls to the underlying interface. Moreover, the deployment of the MBean required by a console plugin is automatically managed by the MBeanDeployer utility by using the OSGi Bundle Repository service (OSGi Alliance RFC112) when the monitored ASPIRE module is running on an OSGi platform. More information is detailed in the TOPI paper [47].

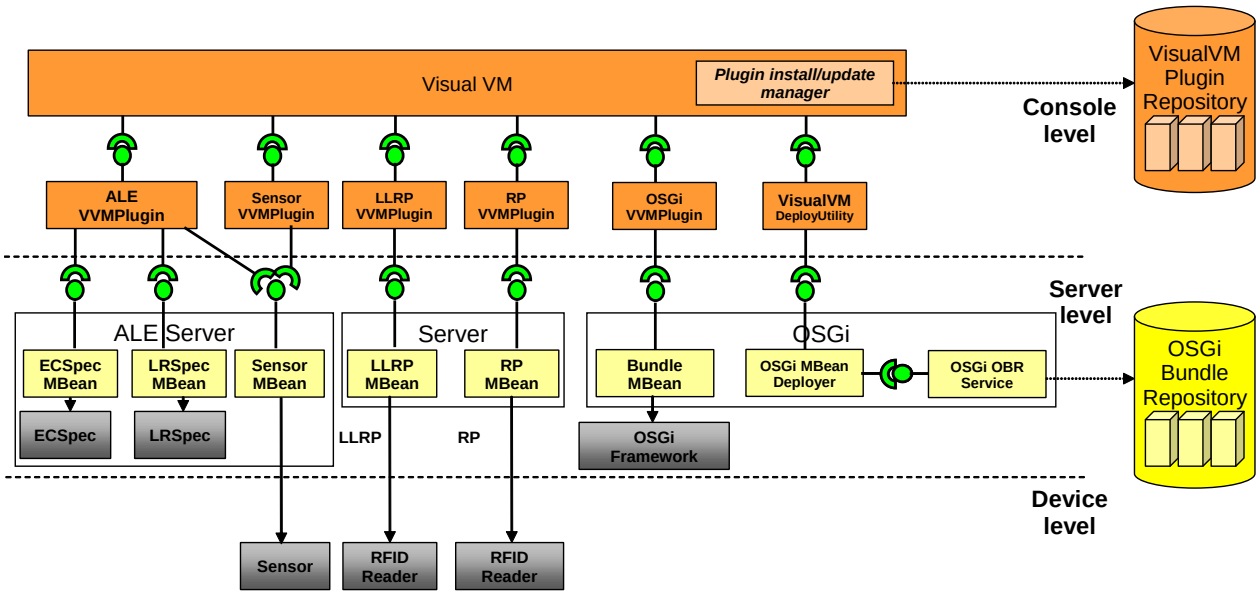

Figure 12. Implementation of the architecture using MBeans as probes and the VisualVM as the management console

5.1 Readers

Readers made available through the AspireRFID middleware should provide a management interface based as JMX MBeans. A minimal set of functionality must be provided in order to allow the configuration of the reader during application execution.

5.2 Middleware modules

AspireRFID middleware modules should provide a management interface based as JMX MBeans. .Manageable AspireRFID modules are Reader Core Proxy, ALE and EPCIS. The set of functionality must be provided in order to allow basic configuration of the module during initialization and to allow operations for the RFID integrator (i.e. ECSpec creation on the ALEServer modules).

5.3 Configurators

The management functionality available in the Aspire middleware is currently decentralized and scattered over different layers. The objective of to unify the access to the different manageable elements through a single application, which would bring to users the simplification of configuration tasks, as well as an overall view of the system elements.

5.4 ASPIRE JMX End-to-End Implementation Design

In order to provide the E2E management, we have to

1. to specify some use cases involving the **RFID middleware administrator**
2. to define the model
3. to implement it with MBeans
4. to provide VisualVM plugins for the central administration and monitoring
5. to provide autonomous rules for autonomic management
6. to provide plug-and-play features for readers in the context of SOHO

5.4.1 Models

We propose a hierarchical model of the management:

- The top level represents the components : ALE, BEG, EPCIS, APE
- The level 2 represents the probes (For instance, the ECSpec). **Probes represent important information for the RFID MW administrator.** Probes can provide metrics

(for instance, number of the generated reports during the last ten minutes for all ECSpecs). Those metrics can aggregate metrics from the level 3 (Instance). Probes can have operations. The operation can be a subset (or a superset) of the operations provided by the RCP tools. The Probe can instantiate an indirect factory of instances. For instance (ECSpec add ECSpec have the consequence of load and start a new ECSpec. The creation of the ECSpec registers an instance of the Probe).

- The level 3 represents the instances of the probes. Instances can have metrics (for instance, number of the generated reports during the last ten minutes). Instances can have operations. They can be created as the consequence of level 2 operations but they can be also created by external events (for instance, deployment of a Plug and Play sensor).

The naming convention is hierarchical

```
org.ow2.aspire.<module>
org.ow2.aspire.<module>.<probe>
org.ow2.aspire.<module>.<probe>.<instance>
```

For instance,

```
org.ow2.aspire.ale
org.ow2.aspire.ale.llrp
org.ow2.aspire.ale.llrp.tagsysOnCOM1
```

5.4.2 MBeans Implementations

Probes and Instances are implemented by MBeans
Probes MBeans are created at starttime.
Instance MBeans are created and destroyed as the consequence of operations (on probe level or others)
Probes objectnames are used for the naming convention

```
org.ow2.aspire:type=<module>,name=<probe>
```

Instances objectnames are used for the naming convention

```
org.ow2.aspire:type=<module>,name=<probe>,instance=<instance>,…
```

For instance

```
org.ow2.aspire:type=ale,name=llrp
org.ow2.aspire:type=ale,name=llrp,instance=tagsysOnCOM1
```

Extra attributes could be defined and added into the objectnames naming schema.

5.4.3 VisualVM Plugins

The default plugins shipped with the VisualVM target JVM low level data such as memory usage, thread allocation and so forth. However, the plugin infrastructure allows the VisualVM to be extended. Therefore, by providing custom plugins it is possible to leverage that infrastructure and transform the VisualVM in a management console for specific types of Java applications. In the case of AspireRFID, the VisualVM acts as the converging point of different plugins for managing the different components of that RFID middleware.
A plugin instance typically monitors MBeans of the same Java process to which it is connected. Each Aspire component (ALE, EPCIS) would have its own management plugin. In the case of an OSGi platform running different Aspire components, all corresponding VisualVM plugins would be available. If a given Java process only provides on manageable AspireRFID component, only the corresponding management plugin would be enabled and displayed by the VisualVM.

Figure 13 depicts the ALE Server management plugin for the VisualVM. In this case, the VisualVM connects to the Java process that executes the ALE Server and inspects the

available MBeans whose object names match the pre-defined naming pattern. Based on that information the plugins inform the VisualVM if they are enabled or not. The figure provides management operations for configuring ECSpecs and LRSpecs, as well as informing which sensors are deployed in the platform. For each sensor the measurement information is displayed in a graph that shows the evolution of values over the time elapsed.

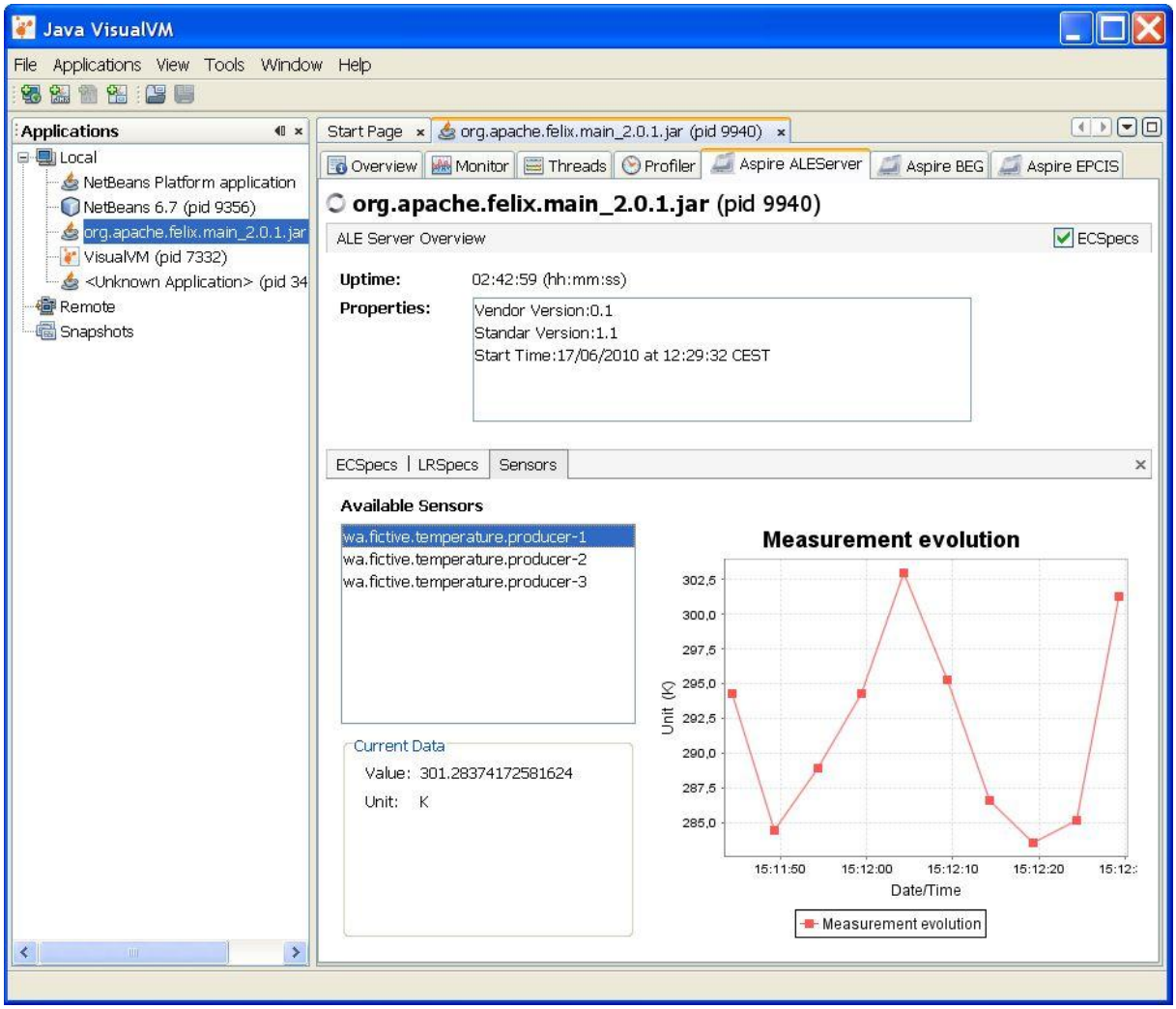

Figure 13 VisualVM plugins for Aspire E2E Management

5.4.4 Jasmine Autonomic Manager

A strong requirement is monitoring several components executed on a distributed platform. For instance, admin must follow the metrics associated to a transaction id across the different components (ALE, BEG, EPCIS, …).

Autonomic managers can help the system administrator to monitor a complex system such as the Aspire middleware. Most of autonomic managers are inspired from the MAPE[19] architectural pattern. OW2 Jasmine is an autonomic management for JavaEE clusters management. In Jasmine, probes and effectors are implemented by JMX MBean distributed over several servers. The metrics collected by probes can be logged for a posteriori auditing and for improving the knowledge. The decisions are implemented ECA rules written in the Drools syntax. Rules consume metrics and trig actions: alerts (SMS sent to the operator, architecture reconfiguration in case of server fail, ….

[19] Monitor-Analyze-Plan-Execute http://www.ibm.com/developerworks/autonomic/library/ac-edge6/

5.4.5 Automatic readers discovery

The installation of new readers and their configuration in the ALE Server is a burden for non-expert people (shop manager) in the context of SOHO.

We recommend to propose a “zero-configuration” feature for readers communicating with IP (Ethernet, Wifi, Bluetooth). The zero-configuration can be provided by popular service discovery protocols and frameworks such as UPnP and DNS-SD (aka mDNS or Apple’Bonjour).

5.4.6 Documentation

The full documentation of the E2E Management components is available in the OW2 Aspire public wiki : http://wiki.aspire.ow2.org/xwiki/bin/view/Main.Documentation/E2EMngmt

5.4 ASPIRE JMX End-to-End Implementation Design

5.4.7 Reader Core Proxy

5.4.7.1 Component analysis

The Reader Protocol bundle defines the API that is used by the implementation bundle. The reason this strategy is followed is to allow for other developers to implement their own implementations if such an alternative is required. The operations defined through the API are also the ones that are exported as management operations through JMX.

The reader protocol implementation bundle is based on the codebase of Fosstrak[1] reader project. This bundle is responsible for creating the JMX MBean and exporting it through MOSGI[20]. Moreover, it maintains the configuration which can be updated through JMX.

Then the implementation bundle should be deployed in the OSGI container and the implementation class should be identified in the RP configuration through JMX.

5.4.7.2 JMX MBean

Through the managed OSGI (MOSGI) we expose one managed bean (MBean) that provides several operations and attributes. This MBean is defined within the ReaderProtocol bundle through the Java interface *org.ow2.aspirerfid.reader.rp.RmRpMBean*. The MBean implementation exists in the ReaderProtocolImpl bundle through the *org.ow2.aspirerfid.reader.rp.impl.ReaderProtocol* class. The following is provides is the list of operations and attributes.

5.4.7.2.1 MBean Operations

Start,stop and check the functionality of the reader proxy based on the active configuration:

```
public boolean start();
public boolean stop();
public boolean isStarted();
```

Loads an updated configuration and makes it active:

```
public void loadConfig();
```

Loads the default configuration and makes it active

[20] Managed OSGi [51]

```
public void resetConfig();
```

Loads an XML serialized configuration file and makes it active:

```
public void loadConfigurationFile(String serializedFile);
```

Serializes the XML configuration file and returns it:

```
public String saveConfigurationFileAs();
```

Get or set functional parameters of the proxy bundle, without making them active. To activate and save the configuration you have to use the loadConfig operation:

```
public void setEPC(String epc);
public String getEPC();
public void setName(String name);
public String getName();
public void setManufacturer(String manufacturer);
public String getManufacturer();
public void setManufacturerDescription(String desc);
public String getManufacturerDescription();
public void setModel(String model);
public String getModel();
public void setHandle(int handle);
public int getHandle();
public void setRole(String role);
public String getRole();
public void setMaxSourceNumber(int number);
public int getMaxSourceNumber();
public void setMaxTagSelectorNumber(int number);
public int getMaxTagSelectorNumber();
public void setMaxTriggerNumber(int number);
public int getMaxTriggerNumber();
public void addReader(String name, String className, String
propertiesFile);
public void addReaderReadpoint(String readerName, String readpoint);
public String[] getReaders();
public String getReaderClassName(String readerName);
public String getReaderPropertiesFile(String readerName);
public String[] getReadPointNames(String readerName);
public void setCurrentSource(String source);
public String getCurrentSource();
public void addSource(String name, boolean fixed, String readpoint);
public String[] getSources();
public boolean getIsSourceFixed(String sourceName);
public String getSourceReadpoint(String sourceName);
public void addIOEdgeTriggerPortManager(String port);
public String[] getIOEdgeTriggerPortManager();
]public void addIOValueTriggerPortManager(String port);
public String[] getIOValueTriggerPortManager();
public void setTcpServerConnection(boolean isEnabled);
public boolean getTcpServerConnection();
public void setTcpPort(int port);
public int getTcpPort();
public void setHttpServerConnectionEnabled(boolean isEnabled);
public boolean getHttpServerConnectionEnabled();
public void setHttpPort(int port);
public int getHttpPort();
public void setNotificationListenTimeout(long timeout);
public long getNotificationListenTimeout();
```

```
public void setThreadPoolSize(int size);
public int getThreadPoolSize();
//For the active source
public void setIsSourceFized(boolean isFxed);
public void setGlimpsedTimeout(long timeout);
public long getGlimpsedTimeout();
public void setObservedThreshold(long threshold);
public long getObservedThreshold();
public void setObservedTimeout(long timeout);
public long getObservedTimeout();
public void setLostTimeout(long timeout);
public long getLostTimeout();
public void setReadCyclesPerTrigger(int cycles);
public int getReadCyclesPerTrigger();
public void setMaxReadDutyCycles(int cycles);
public int getMaxReadDutyCycles();
public void setReadTimeout(long timeout);
public long getReadTimeout();
```

5.4.8 ALE

The probes are Server, LLRP, ECSpec, Sensor
The naming is org.ow2.aspire:type=ale

5.4.8.1 Server

5.4.8.1.1 Probe level

The naming is org.ow2.aspire:type=ale,name=server
Operations

- none

Data

- uptime (read only)
- properties : Map<String, String>

Metrics

- none

5.4.8.1.2 Instance level

No instance

5.4.8.2 LLRP

5.4.8.2.1 Probe level

The naming is org.ow2.aspire:type=ale,name=lrspec

Operations

- addLLRP public void addLogicalReader(String logicalReaderName, String lrSpecDescription) throws IOException;
- removeLLRP public void removeLogicalReader(String logicalReaderName) throws IOException;

Data

- none

Metrics

- none

5.4.8.2.2 Instance level

The naming is org.ow2.aspire:type=ale,name=lrspec,instance=<logicalname>
Operations

- none

Data

- Map<String, String> properties

Metrics

- number of Tag reads in the last minute
- number of Tag reads in the last ten minutes
- number of Tag reads in the last hour
- number of Tag reads in the last ten hours
- total number of Tag reads since the start time

5.4.8.3 ECSpec

5.4.8.3.1 Probe level

The naming is org.ow2.aspire:type=ale,name=ecspec

Operations

- addECSpec
- removeECSpec

Data

- none

Metrics

- none

5.4.8.3.2 Instance level

The naming is org.ow2.aspire:type=ale,name=ecspec,instance=<name>
Operations

- subscribe
- unsubscribe

Data

- none

Metrics

- number of generated ECreports in the last minute
- number of generated ECreports in the last ten minutes
- number of generated ECreports in the last hour
- number of generated ECreports in the last ten hours
- total number of generated ECreports since start time

- number of read/event cycles in the last minute r
- number of read/event cycles in the last ten minutes r
- number of read/event cycles in the last hour r
- number of read/event cycles in the last ten hours r
- total number of read/event cycles since creation

5.4.8.4 Sensor

Monitor sensors attached to the host.
Sensors are OSGi WireAdmin producers wired to the ALE server with the OSGi WireAdmin.

5.4.8.4.1 Probe level

The naming is org.ow2.aspire:type=ale,name=sensor

Metrics

5.4.8.4.2 Instance level

Operations

- none

Data

- source r
- lastValue rn
- lastTimestamp rn
- lastValueType (SI unit) rn
- valuesTypeHistory (Position,Measurement,State,<Image,timestamp>,<Long,timestamp>) r
- valuesHistory Collection of (Position,Measurement,Image,Long) r
- historyMaxSize (0 to Integer.MAXVALUE) rw
- historyMaxTimeWindow (0 to Long.MAXVALUE) rw

Metrics

- number of pushed data in the last minute
- number of pushed data in the last ten minutes
- number of pushed data in the last hour
- number of pushed data in the last ten hours
- total number of pushed data since the wired time

5.4.9 EPCIS

5.4.9.1 Server

5.4.9.1.1 Probe level

The naming is org.ow2.aspire:type=epcis,name=server

Operations

- none

Data

- uptime (read only)
- properties : Map<String, String>

Metrics

- none

5.4.9.1.2 Instance level

No instance

5.4.9.2 MasterData

5.4.9.2.1 Probe level

The naming is org.ow2.aspire:type=epcis,name=masterdata

Operations

none

5.4.9.2.2 Instance level

The naming is org.ow2.aspire:type=epcis,name=masterdata,instance=<name>

Operations

- generateEventCapture

- subscribeQuery
- unsubscribeQuery

Data
- none

Metrics
- number of captured MasterData in the last minute
- number of captured MasterData in the last ten minutes
- number of captured MasterData in the last hour
- number of captured MasterData in the last ten hours
- total number of captured MasterData since start time

- number of queried MasterData in the last minute
- number of queried MasterData in the last ten minutes
- number of queried MasterData in the last hour
- number of queried MasterData in the last ten hours
- total number of queried MasterData since start time

5.4.9.3 EPCIS Event

5.4.9.3.1 Probe level

The naming is org.ow2.aspire:type=epcis,name=masterdata

Operations

none

5.4.9.3.2 Instance level

The naming is org.ow2.aspire:type=epcis,name=epcisevent,instance=<name>

Operations
- generateEventCapture
- subscribe
- unsubscribe

Data
- none

Metrics
- number of captured EPCISEvent in the last minute
- number of captured EPCISEvent in the last ten minutes
- number of captured EPCISEvent in the last hour
- number of captured EPCISEvent in the last ten hours
- total number of captured EPCISEvent since start time

- number of queried EPCISEvent in the last minute
- number of queried EPCISEvent in the last ten minutes
- number of queried EPCISEvent in the last hour
- number of queried EPCISEvent in the last ten hours
- total number of queried EPCISEvent since start time

5.4.10 UPnP Profile for RFID Readers (UJF)

UPnP is the defacto standard for plug-and-play appliances in the small shop and small office (SOHO) context. In order to alleviate the end-user (which is not a RFID integrator or expert),

the Aspire middleware will provide a way to taken in account plug-and-play RFID readers (such as shop portals, portable readers, ..) using UPnP
The preliminary version of the service description (DCP[21] of the RFID Readers is urn:schemas-ow2aspire-org:service:Reader:1 (see Annex 1). It can be combined with standardized SDCP[22] (TemperatureSensor[23], LowPower[24], ...) . It enables to control points (i.e. the ALE Server) to discover RFID readers and to automatically receiving read tag events by subscribing notifications on the ReportMembers state variable.

Various implementations of the UPnP specification are available for Java. The Apache Felix UPnP Base Driver provides a bridge between OSGi registry and the UPnP network. It can directly integrated in the OSGi profile of the Aspire ALE Server to discover and to use the UPnP RFID readers as the Figure 14 shows

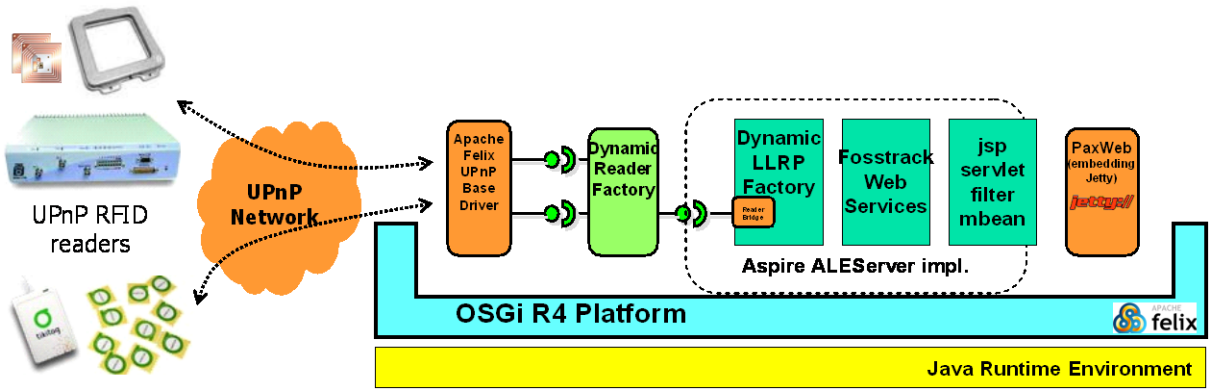

Figure 14 Integration of UPnP RFID Readers in the ALE Server' OSGi profile

5.5 Good Practices recommendations

The Aspire project' default builder is Maven2.
The Aspire project' default IDE is Eclipse IDE (+M2Eclipse plugin for Maven).

MBeans definition (ie *MBean interfaces) must be built and packaged as Maven artefacts.

MBeans implementations must be packaged separately of the associated AspireRFID components core projects. The assembly projects wrap together core classes and MBean implementations in order to provide the deployment artefacts (i.e. war or OSGi bundles).

NetBeans is the recommended IDE to development of the VisualVM plugins (which are packaged as NBM NetBeans Modules).

MBeans and VisualVM plugins must be documented in the OW2 Aspire Wiki
http://wiki.aspire.ow2.org/xwiki/bin/view/Main.Documentation/E2EMngmt

[21] Device Control Protocols
[22] http://upnp.org/sdcps-and-certification/standards/sdcps/
[23] http://upnp.org/specs/ha/hvac/
[24] http://upnp.org/specs/lp/lp1/

5.6 Security and Privacy standards and certification criteria

Privacy and security are application- and context-dependent. As explained in D2.5, the standardisation of practices leading to security and privacy get complicated by the existence of technical, social and cultural barriers; and the tension between business and social interests.

There are a number of European projects aiming to put such tension at rest. Of particular importance is the FP7 project EuroPriSe[25], which aims at standardising a privacy criteria for Information and Communication Technologies (ICT) in the European space, and create certification programmes and an European Privacy Seal to make use of branding mechanisms to award compliant organisations. Led by a prestigious German privacy authority, EuroPriSe concluded successfully in 2010 and is posed to become the main privacy seal for European ICT businesses.

The original DoW of ASPIRE proposed the creation of a certification programme and privacy seal for European RFID implementations, or the use of suitable existing ones as is the case with EuroPriSe. OSI made contact with EuroPriSe's management and the collaboration quickly cemented. As a first step, OSI's staff involved in ASPIRE attended a course to become EuroPriSe experts and became certified. Subsequently, OSI held meetings with the management and technicians of EuroPriSe to extend and adapt the certification criteria to RFID and other ubiquitous wireless devices. The main objective was the extension of EuroPriSe to the realm of RFID and the reutilisation of EuroPriSe advanced certification criteria and privacy seal in ASPIRE's endeavours. In doing so, ASPIRE adopters will have the choice of using EuroPriSe's privacy seal with all its benefits.

Results were very encouraging and an extended and adapted certification criteria was created by OSI. Such criteria, currently under review by and subject to the approval of the management of the EuroPriSe project, is listed and detailed in Annex 2.

[25] See https://www.european-privacy-seal.eu

Section 6 End to End Supply Chain Information Management (ONS)

6.1 Overview of the ONS

ONS is a name lookup/directory service for EPC tags. In the software engineering context a directory is mapping between names and values. The service thus allows to look up the values which are usually not easy to remember and may change over time, given the name which is unique, easy to remember and immutable over a long period of time. It functions similar to a dictionary, and likewise a given name can have more than one value attached to it – more than one interpretation. Early examples of directory services include the Yellow Pages and Telephone directories for landlines in a certain town. Modern versions of directory services are a core part of every distributed system. Examples include:

- DNS [31][32][33]: Matching domain names –www.example.com – to URI's – 192.0.32.10
- LDAP [34] and the various descendants of X.500 suite of protocols: Matching resource names – usual folder names in a network – with their actual location at the network.

ONS is based on DNS and follows the same distributed nature and hierarchy of servers. Provided with a request about a specific EPC tag, the ONS server shall return service end-points where additional information can be obtained about this tag. There can be multiple end- points associated with a single tag, each one providing potentially different information or the same information in different formats.

Each EPC translates to an alphanumeric string, which is treated like a Universal Resource Identifier (URI). The URI is separated in distinct parts by dots (.), with each dot indicating different domains and points of delegation of authority. Each entity in this hierarchy shall either provide the final answer to the query –if responsible to do so – or forward the query to the next point in the delegation chain.

For example an EPC might translate to "*urn:epc:id:sgtin:0614141.000024.400*" which in turn would produce an ONS request for "*000024.0614141.sgtin.id.onsepc.com*" – more on how the request is produced later. The first domain '.com' is a top level domain. 'onsepc' is owned and managed by EPCGlobal, Serialized Global Trade Item Number (SGTIN) by the organization responsible for SGTIN code namespace and 0614141 is owned and managed by the company that has purchased the particular namespace. Requests like that to an ONS server would yield a result like: *http://epc-is.example.com/epc-wsdl.xml*, which is the web service EPCIS interface for the company example.com which owns and has issued the specific tag.

As a standard, the ONS service defines only the core functionality needed for an application to discover information end-point about a specific EPC tag. Contrary to other directory services, it does not provide any information about what kind of additional information the end-points provide or how to actually consume the new-found services. Such information has to be obtained by contacting the provider of the service outside the context of the ONS request. Given that RFID implementation are based on still emerging and sometimes still open standards we believe this is an acceptable trade-off for the sake of simplicity and robustness. Also, in its current incarnation the ONS definition does not have any support of provision for security or user authentication. Future implementation can include this, as the use of DNS-SEC [35] (secure version of the DNS protocol upon which ONS is working) becomes more widespread. ASPIRE's proposal fully implements the ONS standard in its current version (1.0.1). The provided ONS server accessing API automatically does any

necessary processing to go from the EPC tag format to the ONS request format as defined in the EPCGlobal protocol (see following API description for details).

6.2 Directory Server Architecture and Implementation

According to the EPCGlobal ONS standard, ONS servers are standard DNS servers containing the appropriate zone files for their domain of responsibility. Each zone file contains Name Authority Pointer (NAPTR [36]) records which mach specific EPC classes with one or more service end-points. The endpoints can be EPCIS repository interfaces, SAOP or XML-RCP endpoints or anything else that can accept further queries about the specific tag.

Since the ONS server is in effect a DNS server, our implementation uses the BIND [37] open source software on a Windows platform. BIND is the de-facto reference and most widely accepted and used DNS implementation, thus the reason for our selection. We set up the server as an authoritative only server for all EPC tags belonging to one company – all tags share the same company identifier – according to the BIND installation manual [38]. There were no special set-tings used, and the server is accessible via requests at ports TCP/UDP 53 as per the default set-tings for DNS requests. A sample zone file, for a company with identifier 1 and product classes 3 and 4 is depicted in Table 5 below.

```
$TTL 6h
$ORIGIN      1.sgtin.id.onsepc.com.
@ IN SOA ns1.1.gtin.id.onsepc.com. info.1.gtin.id.onsepc.com. (
                    2010112107
                    10800
                    3600
                    604800
                    86400)

@ NS ns1.1.sgtin.id.onsepc.com.
ns1 IN A     127.0.0.1

3      IN      NAPTR      0      0      "u"      "EPC+ws"      "!^.*$!      http://epc-
is.example.com//aspireRfidTracking/tracking.wsdl!" .
4      IN      NAPTR      0      0      "u"      "EPC+ws"      "!^.*$!      http://epc-
is.example.com//aspireRfidTracking/tracking.wsdl!" .
3 IN NAPTR 0 0 "u" "EPC+epcis"                "!^.*$!                    http://epc-
is.example.com//aspireRfidEpcisRepository/query.wsdl!" .
4 IN NAPTR 0 0 "u" "EPC+epcis"                "!^.*$!                    http://epc-
is.example.com//aspireRfidEpcisRepository/query.wsdl!" .
```

Table 5 Sample zone file.

Company has identifier one (1), is using the sgtin encoding scheme, and is currently offering two (2) classes of products, with identifiers three (3) and four (4). For each class, both an EPC-IS and a /tracking endpoint are defined.

This particular zone file defines 2 service endpoints for every product of classes 3 and 4 belonging to company 1. One is the EPCIS repository – of type “EPC+epcis” – and the other one is the tracking application – of type “EPC+ws”. The company has is using the Serialized Global Trade Item Number (SGTIN) coding scheme for the issued tags.

This ONS server has the domain name: ns1.1.sgtin.id.onsepc.com and is responsible for all URI’s that fall under the 1.sgtin.id.onsepc.com domain. It can and shall give an authoritative answer for any URI in that domain – if the necessary records exist. For any query regarding an URI that is outside the authoritative namespace of the server, the server is responsible for forwarding the query to the ONS server directly above it in the ONS/DNS hierarchy for resolving. The process will continue recursively until the query reached the final responsible ONS server. A sample flow is illustrated in Figure 15 below.

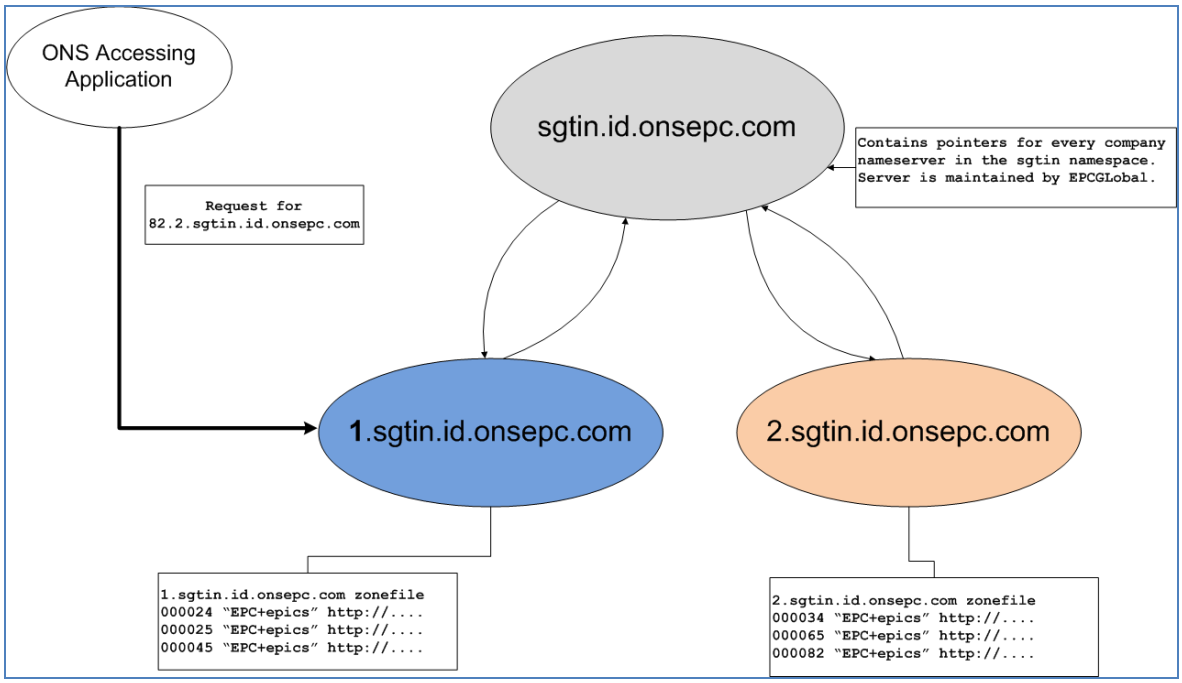

Figure 15 Hierarchy and message flow of ONS servers

To allow applications to access and send requests to the ONS server, a small API was built. The API consists of several Java ™ classes that accept as input the IP of the ONS server and an EPC tag, and return the ONS server answer if available. The API is based on the open source dnsjava [39] library. In order to form a valid ONS request from a specific EPC tag certain processing has to oc-cur. Quoting from the ONS standard document we have that in order to query the DNS for the EPC, the URI form specified above must be converted to domain name form in Step 2. The procedure for this conversion is as follows:

1. Begin with an EPC represented in the pure identity URI form as defined in Section 4.3.3 of the EPCglobal Tag Data Standards [EPC]. For example, urn:epc:id:sgtin:0614141.000024.400.
2. Remove the urn:epc: prefix (in the example, leaving id:sgtin:0614141.000024.400).
3. Remove the serial number field. In all tag formats currently defined in [EPC] (SGTIN, SSCC, SGLN, GRAI, GIAI, and GID), the serial number field is the rightmost period (.) character and all characters to the right of it. (In the example, this leaves id:sgtin:0614141.000024)
4. Replace each colon (:) character with a period (.) character (in the example, leaving id.sgtin.0614141.000024)
5. Invert the order of the remaining period-delimited fields (in the example, leaving 000024.0614141.sgtin.id)
6. Append .onsepc.com. In the example, the result is 000024.0614141.sgtin.id.onsepc.com.

The API does all the processing required so as to provide a clean and data agnostic interface to any application. It supports all EPC coding schemes currently defined by EPCGlobal, namely:

- SGTIN: Serialized Global Trade Item Number
- SSCC: Serial Shipping Container Code
- SGLN: Serialized Global Location Number
- GRAI: Global Returnable Asset Identifier
- GIAI: Global Individual Asset Identifier
- GID: General Identifier

To cater for future revisions and expansions, the API provides the capability for issuing DNS re-quests without the processing needed for EPC tags. As a final note maintenance of the ONS server zone files and additional administration should be done following the recommendations for DNS servers.

6.3 Tracking Application and Interfaces

6.3.1 System Overview.

In this section we describe a novel application that uses the functionality of the ONS server infrastructure. The application becomes part of already existing RFID deployment within a company – as deployments we refer to EPCIS repositories as defined by EPCGlobal. It facilitates data exchange between deployments on different enterprises for the purpose of tracking EPC tagged products as they move from one administrative domain to the next, as part of supply chain. It is based on the 3-tiered model paradigm. Core functionality, which spans the data and the application tier, is the following:

- A new /tracking interface is added to the EPCIS repository is the application is attached to, along with the /query and /capture interfaces. The tracking interface can be used by an external accessing application to obtain data about any EPC tag belonging to this par-ticular company/repository. Data includes both events from the local repository and data reported from foreign domains as products move to them. The tracking interface is a SOAP interface.
- The /tracking interface can also be used by foreign applications to report data about an EPC tag. Data includes at least time, date, geographical coordinates and information about the company which received the item and issued the data. Data can also include details about the particular item (name, features, price, date of production e.t.c) if available. All data of this kind is stored in a local database, separate from the EPCIS repository database.
- Finally the application polls the attached EPCIS repository periodically for any events regarding EPC tags having a company identifier different than the one designated as ‘parent company’. Events are grouped together by company, a query is send to the ONS server for an EPCIS/tracking service endpoint, and an attempt is made to report the events.

A typical deployment scenario would involve several business partners – a factory that makes the products, a wholesale business with a warehouse and a retailer. Each partner has EPCIS repository installations along with the proposed traceability application. The applications on both the wholesale business and the retailer report data about incoming products to the factory, by making use of the ONS server to discover appropriate service endpoints. An application message interaction cycle is depicted in Figure 16 for the first participant in a supply chain after the producer. Namely:

1. The warehouse receives cases of products from the factory. An EPC event is generated and stored on the local EPCIS repository
2. The tracking application on the warehouse polls the attached EPCIS repository for any new EPC events. The newly generated event is fetched.
3. Having received new events about EPC tags not belonging to the company, the ONS is polled to find an appropriate /tracking interface to report the events. The interface be-longing to the factory installation is discovered.
4. The events are reported back to /tracking interface of the factory EPCIS-repository

6.3.2 Data and application tier overview

The data tier of the application consists of 2 parts (Figure 16). One is a connection to the attached EPCIS repository via the /query web service interface. When a request about a specific EPC tag arrives, the local EPCIS repository is queried about all events containing that tag. While events from the EPCIS repository might contain additional data, in the current incarnation of the system only tag, date, time and geographical coordinates of the reader capturing the event are kept. The second part of the systems data tier is a local database. It stores the bulk of the application data, namely all events that are reported from foreign tracking application running on business partners.

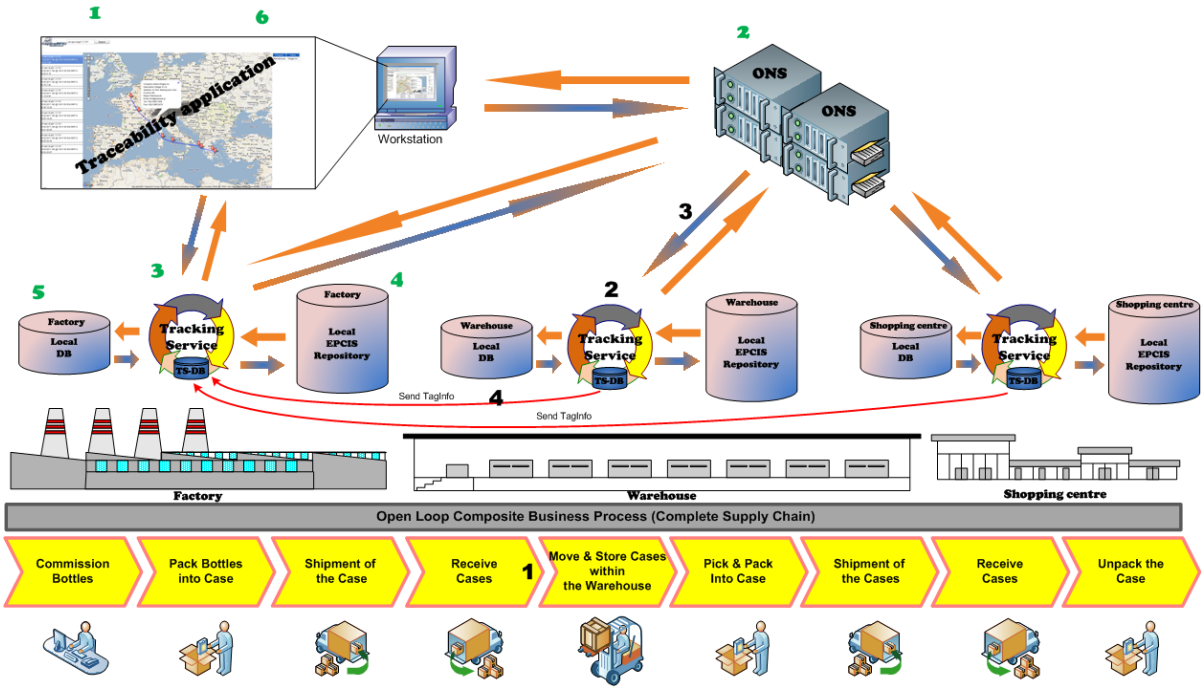

Figure 16 Overview for a typical deployment of an inter-enterprise RFID system

A final note regarding the data tier of the proposed system:
Since all of the data is generated by events retrieved from EPCIS repositories, it would be possible to not use a separate database and store everything on the attached EPCIS repository itself. Considering this solution we faced various challenges. While more or less EPCIS repositories follow the same structure for their data, a lot of parameterization can occur. This alone can cause severe non compatibilities from system to system which cannot be solved with simple settings tweaking. Apart from that, even for exactly identical EPCIS repositories there are systemic issues when trying to capture data directly as events in the repository. One company should have records possibly indicating 'business locations' or 'business step' from another company which makes little sense – and raises a lot of data privacy issues. Lastly, even in the simplest of cases some changes would have to be done to EPCIS repository structure, and the repository would be polluted and burdened with additional out of scope data. The goal for the application is to be a 'value adding add-on', which can be enabled when needed without affecting the operation of the EPCIS repository. Keeping everything on a different database allows users to enable or disable the tracking infrastructure without affecting existing systems. It also enables them to use the system even when only a fraction of their business partners choose to do so, since those who do choose to use it will enjoy the added functionality without negative results for those who don't.

Technology wise the /tracking interface for the attached EPCIS repository is a SOAP web service, which exchanges structured data in XML format. The implementation for the font-end to the out-side world is done on Java using the JAX-WS technology for creating xml web services. The open source Apache CXF [43] framework was used for the final development.

As described in the system overview section the /tracking interface supports 2 main functions. First is the query function. It accepts as input a fully qualified URI representing a tag - urn:epc:id:gid:1.3.128 – and returns a list of all data objects available for this EPC tag. The purpose of the query function is to serve as the main information end-point for 3rd party consumers and service providers, which can include entities both inside and outside the boundaries of the company.

Second is the report function. This is used by the tracking application running on remote domains to report data about company products as they move into their domain – and vice versa. The application polls the attached EPCIS repository periodically for new EPC event data, discards everything regarding epc-tags that have are 'mother' company property, contacts the ONS server to locate an appropriate endpoint and proceeds. The input to this function is a structured XML doc-ument. An example document containing 2 data objects is shown in Table 6 below.

```
<?xml version="1.0" encoding="UTF-8"?>
<trackerdocument xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 creationDate="2006-05-04T18:13:51.0Z" uri="http://www.example.com/epcis/query.wsdl">
    <companyInfo>
        <name>Example Company</name>
        <description>Widgets</description>
        <address>19.7 Markopoulou Ave</address>
        <country>GR</country>
        <region>Attica</region>
        <email>info@example.com</email>
        <tel>210-555-555</tel>
        <fax>210-555-556</fax>
    </companyInfo>
    <tagIdList idCount="2">
        <tag Id="urn:epc:id:gid:1.3.128">
            <geoCoords>35.25:14.28</geoCoords>
            <time>01:01:01.001</time>
            <userData JSONEncUserData=""/>
            <epcClassProperty JSONEncPropList=
"[{\"Manufacturer\":\"example.ltd\"},{\"Color\":\"Blue\"}]"/>
        </tag>
        <tag Id="urn:epc:id:gid:1.3.129">
            <geoCoords>35.25:14.28</geoCoords>
            <time>01:01:01.001</time>
            <userData JSONEncUserData=""/>
            <epcClassProperty
JSONEncProList="[{\"Manufacturer\":\"example.ltd\"},{\"Color\":\"Red\"}]"/>
        </tag>
    </tagIdList>
</trackerdocument>
```

Table 6 XML payload to report a pair of EPC tags to the producer.

The implementation offers a simple client for consumers of the web-service. The client uses the CXF framework which is used for the rest of the application. Naturally prospective consumers can develop their own clients using the wsdl (web service description language [44]) interface definition.

6.3.3 Presentation Tier and sample applications.

The system is designed to provide a service that is consumed by end users and so it does not have a per-se presentation tier. Final consumers are expected to define their own presentation applications to match their needs. Such a possible use-case is presented here to showcase the added value that can be extracted from our proposal.

The scenario involves a traceability application with 'Google of things' like interface. The user can enter a tag in the form or a URI, and search for any data available. Data is presented on

a map for visual reference. The user can click any point on the map to reveal additional information about the product that carries the tag and also the company that generated each data point (Figure 17). The demo application was created using the Google Web Toolkit [45] open source framework for web application. [A small note here: According to the Google Terms of Use [46], the Google maps framework for GWT is open source and free of charge for non-commercial applications. Using the framework for the purposes of tracking company assets or for a service that includes a fee requires purchasing a license from Google. However the use for purely demonstrative purposes is legal].

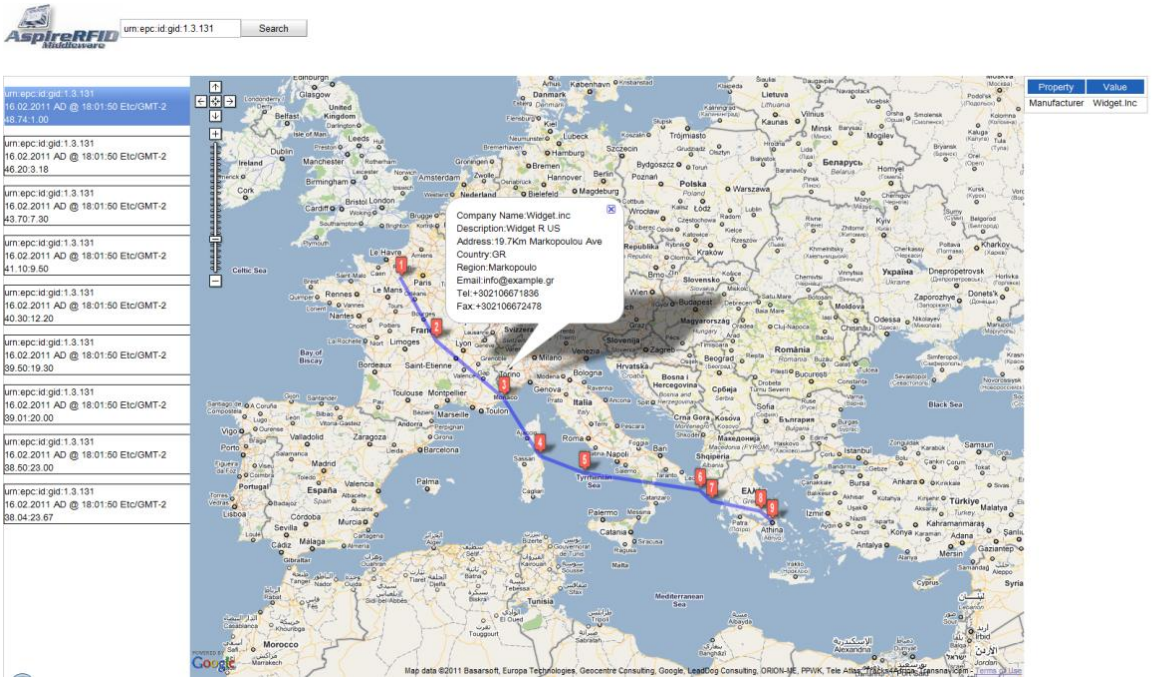

Figure 17 Traceability application GUI based on the GWT.

Component interaction for the proposed Traceability application follows the same flow as the rest of the system as shown in Figure 16. After the user initiates a request for a specific EPC tag, the server backend contacts the ONS server to locate the appropriate /tracking interface and invokes the query function. The tracking application retrieves all relevant data from the EPCIS-repository and the local database and returns them to the user agent for presentation.

By extending or modifying the proposed traceability application, several new applications can be implemented including:

- Fraud prevention: Companies can offer their customers globally an easy way to check if the product they bought is indeed genuine.
- Quality control: Companies handling food or other sensitive products can visually track where items from a potential faulty production lot have shipped and easily recall them.
- Product profiles: Companies or state organizations can provide product profiles based on origin of materials used, or country of production and assembly. Customers can then for example choose products that support eco-friendly production or avoid foods that contain certain genetically mutated ingredients.
- Supply chain management: Companies can involve in statistical analysis about the effectiveness of their supply chains by tracking and comparing processing times versus product or business partner.
- Market analysis: Companies can track and compare product dispersion versus location, product type and time of year.

Section 7 Conclusions

In this deliverable we have analyzed the management (configuration and monitoring) requirements of the AspireRFID middleware. We have provided a state of the art about major management standards and recommendations for Java middleware as well as Web Services technology.

As result of this analysis, this deliverable proposes an extensible management architecture for the AspireRFID middleware. The management architecture is based on the JMX technology for probes, the GUI-based (user-friendly) administration consoles and utilization of an autonomic manager to reduce fine-grain interactions between the middleware and its administrator. In addition to the management reference architecture, the deliverable provides and set of naming recommendation for elements in the architecture and also and group of good practices for the development of their components.

Moreover we have introduced the AspireRFID open source implementation of an object directory service, which is based on EPCGlobal's ONS standards for managing items in an open loop supply chain management scenarios. The implementation has been carried out in a way that respects the ONS standard, while at the same time minimizing interference with existing enterprise systems. Overall, the open source implementation facilitates business partners to achieving automated and rapid tracing of assets and products, as the latter flow through the supply chain. This is a foundation for a number of applications, including high-impact applications such as "google-of-things" and green eco-friendly supply chains. Finally we have reported several added-value applications that could be based on the introduced traceability infrastructure.

Section 8 List of Figures

Section 9 List of Tables

Section 10 Acknowledgements

Part of this work has been carried out in the scope of Master Students Thesis that has been contributed to the AspireRFID open source software. More specifically the following students have partially worked for the Implementations:

Georgios Katsis	Autonomic configuration of LLRP Readers (Section 4.1.1)
Vassiliki Koletti	ECSpec & LRSpec Configurator tools (Section 4.2)
Konstantinos Mourtzoukos	End to End Supply Chain Information Management (ONS) (Section 6)

Section 11 References and bibliography

[1] FossTrak Project, http://www.fosstrak.org/index.html
[2] EPCglobal, "The Application Level Events (ALE) Specification, Version 1.1", February. 2008, available at: http://www.epcglobalinc.org/standards/ale
[3] EPCglobal, "Low Level Reader Protocol (LLRP), Version 1.0.1, August 13", 2007, available at: http://www.epcglobalinc.org/standards/llrp
[4] EPCglobal, "Reader Protocol Standard, Version 1.1, June 21", 2006 available at: http://www.epcglobalinc.org/standards/rp
[5] EPCglobal, "Reader Management 1.0.1, May 31", 2007 available at: http://www.epcglobalinc.org/standards/rm
[6] EPCglobal, "EPCglobal Tag Data Standards, Version 1.4", June 11, 2008, available at: http://www.epcglobalinc.org/standards/tds/
[7] EPCglobal, "EPCglobal Tag Data Translation (TDT) 1.0", January 21, 2006 available at: http://www.epcglobalinc.org/standards/tdt/
[8] EPC Information Services (EPCIS) Specification, Version 1.0.1, September 21, 2007 available at: http://www.epcglobalinc.org/standards/epcis/
[9] LLRP Toolkit, http://www.llrp.org/
[10] Matthias Lampe, Christian Floerkemeier, "High-Level System Support for Automatic-Identification Applications", In: Wolfgang Maass, Detlef Schoder, Florian Stahl, Kai Fischbach (Eds.): Proceedings of Workshop on Design of Smart Products, pp. 55-64, Furtwangen, Germany, March 2007.
[11] C.Floerkemeier, C. Roduner, and M. Lampe, RFID Application Development With the Accada Middleware Platform, IEEE Systems Journal, Vol. 1, No. 2, December 2007.
[12] C. Floerkemeier and S. Sarma, "An Overview of RFID System Interfaces and Reader Protocols", 2008 IEEE International Conference on RFID, The Venetian, Las Vegas, Nevada, USA, April 16-17, 2008.
[13] Russell Scherwin and Jake Freivald, Reusable Adapters: The Foundation of Service-Oriented Architecture, 2005.
[14] The XMOJO Project Product Documentation, available at: http://www.jmxguru.com/products/xmojo/docs/index.html
[15] Java Management Extensions (JMX) Technology Overview, available at: http://java.sun.com/j2se/1.5.0/docs/guide/jmx/overview/architecture.html
[16] Panos Dimitropoulos and John Soldatos, 'RFID-enabled Fully Automated Warehouse Management: Adding the Business Context', submitted to the International Journal of Manufacturing Technology and Management (IJMTM), Special Issue on: "AIT-driven Manufacturing and Management".
[17] Architecture Review Committee, "The EPCglobal Architecture Framework," EPCglobal, July 2005, available at: http://www.epcglobalinc.org .
[18] Achilleas Anagnostopoulos, John Soldatos and Sotiris G. Michalakos, 'REFiLL: A Lightweight Programmable Middleware Platform for Cost Effective RFID Application Development', accepted for publication to the Journal of Pervasive and Mobile Computing (Elsevier).
[19] Application Level Events 1.1(ALE 1.1) Overview, Filtering & Collection WG, EPCglobal, March 5, 2008 , available at: http://www.epcglobalinc.org/standards/ale
[20] WS-I, Basic Profile v1.0, available at: http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html.
[21] Benita M. Beamon, "Supply chain design and analysis: Models and methods", International Journal of Production Economics, Vol. 55 pp. 281-294, 1998
[22] Zhekun Li, Rajit Gadh, and B. S. Prabhu, "Applications of RFID Technology and Smart Parts in Manufacturing", Proceedings of DETC04: ASME 2004 Design Engineering

Technical Conferences and Computers and Information in Engineering Conference September 28-October 2, 2004, Salt Lake City, Utah USA.
[23] JSR 256, “Mobile Sensor API”, available at: http://jcp.org/en/jsr/detail?id=256
[24] JSR 275, “Units Specification”, available at: http://jcp.org/en/jsr/detail?id=275
[25] JSR 179, “Location API for J2ME”, available at: http://jcp.org/en/jsr/detail?id=179
[26] JSR 257, “Contactless Communication API”, available at: http://jcp.org/en/jsr/detail?id=257
[27] Java Management Extensions (JMX) Technology Overview, available at: http://java.sun.com/j2se/1.5.0/docs/guide/jmx/overview/architecture.html
[28] VisualVM https://visualvm.dev.java.net/
[29] UPnP Forum, http://www.upnp.org/
[30] LLRP Toolkit, http://www.llrp.org/
[31] RFC1035 “Domain Names, Implementation and Specification”, Internet Engineering Task-force - Network Working Group, November 1987, available online at : http://tools.ietf.org/html/rfc1035
[32] RFC1123 “Requirements for Internet Hosts -- Application and Support”, Internet Engineer-ing Taskforce -Network Working Group, October 1989, available online at: http://tools.ietf.org/html/rfc1123
[33] RFC2181 “Clarifications to the DNS specification”, Internet Engineering Taskforce - Network Working Group, July 1997, available online at: http://tools.ietf.org/html/rfc2181
[34] RFC4510 “Lightweight Directory Access Protocol (LDAP): Technical Specification Road-map”, Internet Engineering Taskforce – Network Working Group, June 2006, available on-line at: http://tools.ietf.org/html/rfc4510
[35] RFC4033 “DNS Security Introduction and Requirements”, Internet Engineering Taskforce – Network Working Group, March 2005, available online at: http://tools.ietf.org/html/rfc4033
[36] RFC2915 “The Naming Authority Pointer (NAPTR) DNS Resource Record”, Internet Engi-neering Taskforce – Network Working Group, September 2000, available online at: http://tools.ietf.org/html/rfc2915
[37] Internet Systems Consortium. The BIND software. Available at : http://www.isc.org/software/bind
[38] BIND 9 Administration reference manual. Available online at: http://www.isc.org/files/arm97.pdf
[39] Hibernate Org, http://www.hibernate.org/
[40] Apache CXF, http://cxf.apache.org/
[41] World Wide Web Consortium (W3C) “Web service description language (WSDL) 1.1”, March 2001, available online at: http://www.w3.org/TR/wsdl
[42] Google Web Toolkit. Available online at: http://code.google.com/webtoolkit/
[43] Google Maps/Google Earth APIs Terms of service, May 2009, available online at: http://code.google.com/apis/maps/terms.html
[44] Nikos Kefalakis, Nektarios Leontiadis, John Soldatos, Didier Donsez, “Middleware Building Blocks for Architecting RFID Systems”, MOBILIGHT 2009, pp. 325-336.
[45] Panos Dimitropoulos and John Soldatos, ‘RFID-enabled Fully Automated Warehouse Man-agement: Adding the Business Context’, International Journal of Manufacturing Technology and Management (IJMTM), Special Issue on: "AIT-driven Manufacturing and Management", IJMTM Vol 21, No.3/4, 2010.
[46] Nektarios Leontiadis, Nikos Kefalakis, John Soldatos, “Bridging RFID Systems and Enterprise Applications through Virtualized Connectors", International Journal of Automated Identification Technology (IJAIT), Vol. 1, No.2, 2009.
[47] Kiev Gama, Gabriel Pedraza, Thomas Lévêque and Didier Donsez, "Application Management Plug-ins through Dynamically Pluggable Probes", TOPI 2011 1st Workshop on Developing Tools as Plug-ins, ICSE Workshop, May 28, 2011, Honolulu, Hawaii, USA

[48] Fen Zhu; Mutka, M.W.; Ni, L.M., Service Discovery in Pervasive Computing Environments, IEEE Pervasive Computing, Volume 4, Issue 4, Oct.-Dec. 2005 Page(s): 81 - 90 Digital Object Identifier 10.1109/MPRV.2005.87
[49] EPCglobal Object Name Service (ONS), version 1.0.1, EPCglobal, May 2008, available at: http://www.gs1.org/gsmp/kc/epcglobal/ons
[50] Jeffrey O. Kephart, David M. Chess: The Vision of Autonomic Computing. IEEE Computer 36(1): 41-50 (2003)
[51] Yvan Royon, Stéphane Frénot, Frederic Le Mouel: Virtualization of Service Gateways in Multi-provider Environments. CBSE 2006: 385-392

Annex 1 : UPnP SCPD for UPnP RFID Reader

```
<!--
   Copyright 2005-2011, Aspire

   This library is free software; you can redistribute it and/or modify it
   under the terms of the GNU Lesser General Public License as published by
   the Free Software Foundation (the "LGPL"); either version 2.1 of the
   License, or (at your option) any later version. If you do not alter this
   notice, a recipient may use your version of this file under either the
   LGPL version 2.1, or (at his option) any later version.

   You should have received a copy of the GNU Lesser General Public License
   along with this library; if not, write to the Free Software Foundation,
   Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

   This software is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY
   KIND, either express or implied. See the GNU Lesser General Public
   License for the specific language governing rights and limitations.
-->

<!—version 0.1 -->
<scpd>
  <serviceStateTable>
    <stateVariable> <!-- Reader properties  -->
      <name>Properties</name>
      <sendEventsAttribute>no</sendEventsAttribute>
      <dataType>string</dataType> <!--pairs name-value x-www-form-urlencoded  -->
    </stateVariable>
    <stateVariable> <!--  duration between 2 reports -->
      <name>Duration</name>
      <sendEventsAttribute>yes</sendEventsAttribute>
      <dataType>ui4</dataType> <!--  in milliseconds -->
      <defaultValue>1000</defaultValue>
    </stateVariable>
    <stateVariable> <!--  pre filtering expression for scanned tags in order to decrease the
network traffic due to tag event notification -->
      <name>Filter</name>
      <sendEventsAttribute>yes</sendEventsAttribute>
      <dataType>string</dataType> <!--  regular expression -->
    </stateVariable>
    <!-- more parameters for the reader configuration  -->

    <stateVariable>  <!-- Report members  -->
      <name>ReportMembers</name>
      <sendEventsAttribute>yes</sendEventsAttribute>
      <dataType>string</dataType> <!-- CSV of the hexadecimal representation of EPC
identifiers  -->
    </stateVariable>
  </serviceStateTable>
  <actionList>
    <action>
      <name>SetDuration</name>
      <argumentList>
        <argument>
          <name>newDuration</name>
          <direction>in</direction>
          <relatedStateVariable>Duration</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>GetDuration</name>
      <argumentList>
        <argument>
          <name>currentDuration</name>
          <direction>out</direction>
          <relatedStateVariable>Duration</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>GetFilter</name>
      <argumentList>
        <argument>
```

```
          <name>currentFilter</name>
          <direction>out</direction>
          <relatedStateVariable>Filter</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>SetFilter</name>
      <argumentList>
        <argument>
          <name>newFilter</name>
          <direction>in</direction>
          <relatedStateVariable>Filter</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>GetProperties</name>
      <argumentList>
        <argument>
          <name>currentProperties</name>
          <direction>out</direction>
          <relatedStateVariable>Properties</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
  </actionList>
</scpd>
```

Annex 2 – Privacy Certification Criteria

- **Part 1: Preliminary Issues**

- **A. Scope of the European Privacy Seal**

A special case of IT products creating important privacy issues is that of Portable Wireless Devices, for example Radio Frequency Identification (RFID) devices and mobile phones. These are special because they may be used to determine the location and identification of carriers, or the nature or identity of carried objects, even in the absence of personal data. For example, a drug identified through an RFID tag may be detected by snoopers who may associate such drug with a medical condition of the carrier. A similar logic applies to political, religious, sexual or ideological objects. Tags on clothing or other portable objects may be abused to indirectly identify a person and his or her whereabouts, more so if such devices can also provide location data. In the latter case, object data becomes personal data as defined by the applicable Directives. In the former case the traits of the carrier (data subject) are the personal data as such, being the privacy-sensitive data the identification or location information of the carried object. As the former case is an exception to the current Directives, in May 2009 the EC issued specific recommendations on RFID and privacy and security.

- **B. Definitions**

- **B.3 Portable Wireless Devices Definitions**

Carrier
The Carrier of a PWD (defined below) is an identified or identifiable natural person as defined in B.2 who temporarily or permanently carries a PDW.
Eavesdropper
The Eavesdropper of a PWD (defined below) is a natural or legal person who registers and analyses the communication emitted by a PWD for purposes other than their original ones; likely in an unauthorised and unlawful way.
Location Data
Location Data are those that allow establishing the approximate or exact location of the PWD (defined below); and thereby that of the Carrier. Location data can be generated through specialised devices, for example Global Positioning Systems, or short-distant wireless identification, for example the response of an RFID device to a fixed reader the position of which is known or can be deducted.
Object Data
Object Data are those identifying an object either uniquely or by type or nature; or expressing characteristics of an object. In some cases object data may also include personal data of the carrier, owner or custodian.
Portable Wireless Device (PWD)
A PWD is a hardware device (IT Product) that incorporates a wireless interface to communicate with another compatible device in the sharing of Object Data. The wireless interface is usually electromagnetic, yet audio and visual signals may also be used. Data include a simple code identifying the device or its type, or may include more sophisticated data such as personal data on the owner or user, etc. Examples of PWD include mobile phones, laptops, RFID tags, etc.
Radio Frequency Identification (RFID)
A particular type of low-cost PWD used to identify objects; as defined in SEC(2009) 585 and SEC(2009) 586, 12.5.2009.
PWD Reader

A PWD Reader is a fixed or portable hardware device (IT Product) that incorporates a wireless interface to communicate with PWD.

Tagged Object

Tagged Objects are those which have a PWD temporarily or permanently attached to them to automatically provide Object Data through the wireless interface.

Uncontrolled Wireless Communication

An unprotected wireless communication takes place when a PWD transmits data in an uncontrolled manner, for example upon interrogation by unauthorised third parties.

Vulnerable Wireless Communication

A vulnerable wireless communication takes place when a PWD transmits sensitive data, for example Personal Data, in an unencrypted manner or using a single encryption key or secretive algorithm the leaking of which may compromise a number of similar PWD.

- **C. Target of Evaluation (ToE)**

In the case of PWD, the ToE must take into account all contexts where such devices will be carried or operated. For example, an RFID tag used to track a product in the supply chain may be left attached to such product after the point of sale and create privacy issues for consumers. In this context, the ToE must include all pre- and post-application contexts in which such device could be used or abused by eavesdroppers. Similarly, the ToE must account for any incidental data that the PWD is generating, for example location data or data revealing the nature of the objects being carried by the Carrier. Similarly, the ToE should be extended to account for Object Data potentially disclosing sensitive data about Carriers or personal data through proxy identifiers.

- **SET 1: Overview on Fundamental Issues**
- **1.1 Fundamental Aspects of Processing**
- **1.1.1 Processing Operations; Purpose(s)**

Relevant Questions:

- If the product is a PWD, could the product be used in a context (place, time, position, communication range etc) where eavesdropping is possible?
- If the product is a PWD, is it used to identify a portable object or an object the nature of which is sensitive from a privacy or security perspective?
- **1.1.2 Processed Personal Data**
- ***1.1.2.1 Personal Data***

Relevant Questions:

- If the product is a PWD, does the device contain a unique identifier that can be used as proxy to identify the carrier?
- If the product is a PWD, does it transmit Personal Data to or from other compatible devices?
- ***1.1.2.2 Special Categories of Data***

Relevant Questions:

- If the product is a PWD, is such device used to identify or track an object the nature of which may reveal sensitive data about the Carrier, specifically objects associated with racial or ethnic origin, political opinions, religious or philosophical beliefs, trade-union membership, health or sex life?
- **1.1.3 Controller**

Relevant Questions:

- Does the Carrier have a means to control access to the PWD?
- If not, who has such means and how is such control exerted?
- **1.1.4 Transnational Operations**

Relevant Questions:

- Are Tagged Objects transferred to countries that are neither a member of the EU nor of the European Economic Area?
- **1.2 Fundamental Technical Construction**
- **1.2.1 Data Avoidance and Minimisation**

Relevant Questions:

- Is it possible to carry out the processing without the use of the PWD?
- Do the Object Data associated with the PWD contain unique identification or object nature?
- If yes, is it possible to carry out the processing without such data?
- Does the PWD provide Location Data?
- If yes, is it possible to carry out the processing without Location Data?
- Does the PWD have a mechanism preventing the transmission of Object Data or Location Data outside the original application context?

- **1.2.2 Transparency**

- ***1.2.2.1 Transparency and Description of the Product or Service***

(SEC(2009) 585 and SEC(2009) 586, 12.5.2009)
Relevant Questions:

- In the case of RFID, is it there a summary of the privacy and data protection impact assessment published?
- Are the potential privacy risks relating to the use of PWD in the application identified and disclosed? Have measures for individuals to mitigate such risks been properly documented and disclosed?
- Are there clear indications of the presence of PWD and their readers?
- Do PWD or their readers provide visual or audio indications of activity to alert Carriers of any potentially unauthorised accesses?
- **Set 2: Legitimacy of Data Processing**

- **2.1 Legal Basis for the Processing of Personal Data**
- **2.1.1 Legal Basis for the Processing of Personal Data in General**
- **2.1.1.1 Processing on the Basis of Consent**

Relevant Questions:

- Can the PWD be accessed without the knowledge or authorisation of the Carrier?
- Can the Carrier exert control and so grant or revoke consent at will, for example removing or deactivating the PWD?
- Is the removal or deactivation of the PWD temporary or permanent?
- Does the removal or deactivation of the PWD pose any threat of disadvantage?
- **2.1.4 Special Restrictions on Certain Data Processing under Directive 2002/58/EC**
- ***2.1.4.1 Special Restrictions on the Use of Cookies and other Information stored in the Terminal Equipment of a Subscriber or User***

Relevant Questions:

- Does the product or service involve accessing and storing information on to PWD being carried by an individual?
- **2.1.4.3 Special Restrictions on the Processing of Location Data**

Relevant Questions:

- Does the PWD generate Location Data?

- Can Object Data by used indirectly to identify a Carrier and therefore produce Location Data of an individual?
- **2.2 Special Requirements to the Various Phases of the Processing**
- **2.2.1 Data Collection (Information Duties)**

Relevant Questions:
- Are data collected from the PWD?
- **2.2.3 Disclosure of Data to Third Parties**

Relevant Questions:
- Are Object Data passed on to any third parties so they can detect and communicate with PWD?
- **2.3 Compliance with General Data Protection Principles and –duties**
- **2.3.3 Quality of Data**

Relevant Question:
- Are reasonable steps taken to prevent the faking of the Object Data, for example the cloning of an RFID device, and therefore compromise the Quality of Data?
- **Set 3: Technical-Organisational Measures: Accompanying Measures for Protection of the Data Subject**

- **3.1 General Duties**
- **3.1.1 Preventing Unauthorised Access to Data, Programs, Premises and Devices**
- ***3.1.1.2 Access to Media and Mobile Devices***

Relevant Questions:
PWD:
- If used, which measures prevent unauthorised access to data in the PWD?
- Are these measures adequate?
- Are measures used to track accesses? If yes: Apply Section to the generated data
- ***3.1.1.3 Access to Data, Programs and Devices***

Relevant Questions:
Product:
- Hardware/device: Does the PWD offer access control facilities (e.g. mechanical locks, PIN code, password protection)?
- Can access rights be granted with sufficient granularity per device and application?
- Can such access rights be easily changed by a new Carrier?
- ***3.1.1.4 Identification and Authentication***

Relevant Questions:
Product and service:
- Does the PWD provide adequate identification and authentication measures? See also Section 3.1.1.5 for authentication based on passwords.
- Does the PWD prevent repeating attempts to identify and authenticate after a certain number of unsuccessful attempts?
- Is the method of prevention (e.g. slowing down identification processes, temporarily deactivation of user accounts, permanent deactivation of user accounts) appropriate?
- **3.1.3 Network and Transport Security**

Relevant questions for product or service:
- Is the communication with the PWD an Uncontrolled Wireless Communication?
- If so, is this communication a Vulnerable Wireless Communication?

- If so, may this communication take place in public places or at range long enough for Eavesdroppers to intercept it?

- **3.1.5 Data Protection and Security Management**
- ***3.1.5.1 Security Policy***

Relevant Questions:
- In case of RFID: has the Privacy Impact Assessment been undertaken?
  - Does it assess the implications of the application implementation on the processing of personal data?
  - Does it assess whether the application could be used to monitor people or could lead to monitoring of individuals?
  - Is the level of detail of the assessment appropriate to the privacy risks possibly associated with the application?
  - Are appropriate technical and organisational measures taken to mitigate/eliminate such risks?
  - Are personnel appointed to review and check the appropriateness of measures taken on regular basis?
  - Is the assessment made available to competent authority 6 weeks before implementation?
- ***3.1.5.2 Risk Analysis***

Relevant Questions:
Product:
- Does the risk analysis extend beyond the organisational boundaries in the case of PWD, for example when customers take RFID tags with them?
- ***3.1.5.3 Documentation of Technical and Organisational Data Protection Measures***

Relevant Questions:
Product:
- In the case of RFID products or services, does the Privacy Impact Assessment include a documentation of the implemented security and data protection measures?
- ***3.1.5.4 Documentation of Individual Obligations***

Relevant Questions:
Service:
- Are obligations and duties of employees documented, for example the disabling of RFID tags at the point of sale?
- ***3.1.5.5 Inventory of Hardware, Software, Data and Media***

Relevant Questions:
Product:
- Does the product documentation provide information about PWD containing personal data?

Service:
- Is an up-to-date inventory of PWD used for the service?
- ***3.1.5.6 Media Management***

Relevant Questions:
Product/Service:
- Do PWD used allow the storage of personal data?
- ***3.1.5.7 Appointment and Duties of Security Officers***

Relevant Questions:
Service:

- Has a privacy officer been appointed to deal with privacy implications from RFID in line with the EC recommendations on RFID privacy and security?
- ***3.1.5.9 Data Protection and Security Audit***

Relevant Questions:
Service:
- Are the security and privacy measures been audited by independent experts according to the EC recommendations on RFID privacy and security?
- **3.1.6 Disposal and Erasure of Data**

Relevant questions:
Product:
- Can the Carrier of a PWD safely delete its contents?
- Is it necessary for such Carrier to take the PWD to a specialised centre for removal?
- **3.1.8 Documentation of Products and Services from a Customer´s Perspective**

Relevant Questions:
- If RFID is used, has the documentation been prepared according to the specifications in the EC recommendations on RFID privacy and security?

- **3.2 Technology-specific and Service-specific Requirements**
- **3.2.1 Encryption**

Relevant Questions:
Product or service:
- Does the use of the PWD involve the transmission of personal data through the wireless interface?
- In such case, is this personal data encrypted?
- **3.2.2 Pseudonymisation and Anonymisation**

Product or service:
- Can the PWD use temporary changeable identities to fend off Eavesdroppers?

- **3.2.4 Ensuring Transparency of Automated Individual Decisions**

Relevant Questions:
Product or service:
- Is the PWD capable of making automatic decisions on behalf of their Carriers?
- **Set 4: Data Subjects' Rights**
- **4.1 Rights under the Directive 95/46/EC**
- **4.1.1 Right to Be Informed**

Relevant Questions (for both 4.1.1.1 and 4.1.1.2):
- In the case of RFID, are data subjects and Carriers informed by the controller as required by the EC recommendations on RFID privacy and security?
    - If yes, does the information include:
        - Clear notice of the presence of RFID tags on products/packaging and presence of readers?
        - Consequences of such presence in terms of tag broadcasting information?
        - How to discard, disable or remove tags to prevent disclosure of data?
- **4.1.2 Right of Access**

Relevant Questions:
- data in the PWD be accessed?

- Can data automatically generated by the PWD be accessed?
- Can the owner or Carrier of Tagged Objects access the data of his or her objects?
- If the PWD only contains ID is there provision for data subject to view the semantics of the data on the network?
- How is the owner or Carrier of the PWD authenticated?

- **4.1.3 Right of Correction**

Relevant Questions:
- Can data in the PWD be corrected?
- Can data automatically generated by the PWD be corrected?
- Can the owner or Carrier of Tagged Objects correct the data of his or her objects?
- If the PWD only contains ID is there provision for data subject to correct the semantics of the data on the network?
- How is the owner or Carrier of the PWD authenticated?

- **4.1.4 Right of Erasure**

Relevant Questions:
- Can data in the PWD be erased?
- Can data automatically generated by the PWD be erased?
- Can the owner or Carrier of Tagged Objects erase the data of his or her objects?
- If the PWD only contains ID is there provision for data subject to remove the semantics of the data on the network?
- How is the owner or Carrier of the PWD authenticated?

- **4.1.5 Right of Blocking**

Relevant Questions:
- Can wireless data sent by PWD be blocked, for example by means of devices such as the RFID blocker?
- Can the owner or Carrier of Tagged Objects block the data of his or her objects?