# FlexRFID: A Security and Service Control Policy-Based Middleware for Context-Aware Pervasive Computing

Healthcare Scenarios

*Mehdia Ajana El Khaddar[1], Mhammed Chraibi[2], Hamid Harroud[3], Mohammed Boulmalf[4], Mohammed Elkoutbi[1], Abdelilah Maach[2]

1: SIME Lab, ENSIAS, Rabat, Morocco
2: Ecole Mohammedia des Ingénieurs, Rabat, Morocco
3: Alakhawayn University in Ifrane (AUI), Ifrane, Morocco
4: International University of Rabat (UIR), Rabat, Morocco

***Abstract*— Ubiquitous computing targets the provision of seamless services and applications by providing an environment that involves a variety of devices having different capabilities. The design of applications in these environments needs to consider the heterogeneous devices, applications preferences, and rapidly changing contexts. RFID and WSN technologies are widely used in today's ubiquitous computing. In Wireless Sensor Networks, sensor nodes sense the physical environment and send the sensed data to the sink by multi-hops. WSN are used in many applications such as military and environment monitoring. In Radio Frequency Identification, a unique ID is assigned to a RFID tag which is associated with a real world object. RFID applications cover many areas such as Supply Chain Management (SCM), healthcare, library management, automatic toll collection, etc. The integration of both technologies will bring many advantages in the future of ubiquitous computing, through the provision of real-world tracking and context information about the objects. This will increase considerably the automation of an information system. In order to process the large volume of data captured by sensors and RFID readers in real time, a middleware solution is needed. This middleware should be designed in a way to allow the aggregation, filtering and grouping of the data captured by the hardware devices before sending them to the backend applications. In this paper we demonstrate how our middleware solution called FlexRFID handles large amount of RFID and sensor scan data, and executes applications' business rules in real time through its policy-based Business Rules layer. The FlexRFID middleware provides easy addition and removal of hardware devices that capture data, as well as uses the business rules of the applications to control all its services. We demonstrate how the middleware controls some defined healthcare scenarios, and deals with the access control security concern to sensitive healthcare data through the use of policies. We propose hereafter the design of FlexRFID middleware along with its evaluation results.**

***Keywords*— RFID; Middleware; WSN; Ubiquitous; Pervasive Computing; FlexRFID; Policy-Based; Security; Healthcare; access control***

## I. Introduction

Pervasive computing aims at providing intuitive and seamless support for the users through leveraging the distinct functionalities of a number of devices, and developing various backend applications that use data gathered from these devices. Through wireless communication, the automation devices can share data, and combine them for a more accurate inference of their surroundings. This inference enables applications to reason about the past, the present, and the future, and allows them to behave according to the expectations of the user. This is making pervasive applications very attractive to users on one hand and close to nightmare for developers on the other hand. This is due to the fact that pervasive applications need to deal with device heterogeneity, unreliable wireless communication, duplicate and continuous raw data readings, uncertainty in sensor readings, and changing user requirements and application domains. Therefore, the development of this kind of applications is considered error prone, non-trivial and time consuming, and needs definitely a rescue which is a middleware for pervasive computing [25].

Healthcare services are becoming increasingly pervasive where monitoring technologies are fast becoming integral to the care process and important to realize a proficient healthcare service. WSN and RFID can be considered two adjacent technologies that help tracking healthcare items and patients, and providing context information about them. Sensors measure physiological state (inpatient monitoring), and also allow remote care (outpatient monitoring, i.e. at the patient's home rather than in hospital) [1]. RFID not only offers tracking capability to locate patients in real time while they are moving in a hospital, but also monitors access control to the different medical departments, and provides efficient and accurate access to medical data for doctors and other health professionals [2]. Such technologies assist in the early identification of health issues, and provide alerts in case of emergencies.

The healthcare environment is becoming data driven, in the sense that care providers require information in order to deliver care services. However, health information is sensitive and must be protected [3]. Thus, it is necessary to consider the context in which it is shared. The development of a middleware, which hides the complexity of the underlying network and eases application development, is central to provide a ubiquitous secure healthcare.

The solution proposed in this paper is a middleware which supports simultaneous communication of multiple applications with the RFID and WSN hardware, and deals with the above challenges through the use of policies. The middleware provides all data processing capabilities like filtering, grouping and duplicate removal. The paper is structured as follows, Section II introduces related work. Section III introduces the middleware architecture and focuses more on the policy-based Business Rules Layer, presents the policies types and structure, and shows how all the services provided by FlexRFID can be managed by the use of application-defined policies. Section IV defines and models policies for some healthcare scenarios, followed by conclusions and future work in section V.

## II. Related Work

Most of the existing RFID middleware solutions are commercial. These include “BizTalk RFID” middleware from Microsoft, and “Java RFID System” from Sun, to name a few. Other middleware solutions were developed from research e.g. “WinRFID” by UCLA and “Accada” by ETH Zurich. Sun Java RFID System is a Java based commercial middleware that has a dynamic service provisioning architecture that enables scaling from small to large deployments with high data volume [4]. The Biztalk RFID middleware solution from Microsoft provides support for both standard and non-standard devices through the plug-and-play architecture [5]. It has an event processing engine that manages the RFID events by creating business rules, through which it provides real time visibility of the RFID data [5]. WinRFID [6] developed at the University of California Los Angeles (UCLA), uses web services and enables rapid RFID applications development. It has certain unique features like hiding of communication details from the end users, network management on a large scale, intelligent data processing and routing, support for hardware and software interoperability, provision for system integration and system extendibility, etc. WinRFID exploits the .Net framework‘s runtime plug-in feature to support the addition of new readers, protocols, and data transformation rules with minimum disruption of the existing infrastructure [6]. The Accada middleware [7] developed by ETH Zurich uses EPCglobal (Electronic Product Code) based specifications for the reader protocols, the application level event specifications and the EPCIS (EPC Information Services) capture and query interface to handle RFID data flow across enterprises. It has three main modules: the reader, the middleware, and the EPC information services module. The Accada reader implementation uses standard edition of SUN Java Virtual Machine [7].

For WSN, there exists many middleware approaches. Among these approaches we find the virtual machine, database, application driven, message-oriented, and modular programming middleware [8]. For each of the WSN middleware approaches, some WSN middleware solutions have already been proposed. Hereafter we name some WSN middleware solutions: Impala [8, 9], Mate [10, 8], Middleware Linking Applications and Networks (MiLAN) [8], Sensor Information and Networking Architecture (SINA) [8], Mires [8], etc.

A publish/subscribe middleware providing event based data control mechanisms for healthcare has been proposed in [11]. The main objective of this event based healthcare middleware is to give caregivers fine-grained control over the circumstances for health data transmission. It provides two categories of interaction control rules; subscription rules and event transformation rules. These rules allow the administrative domain to set the circumstances in which it is appropriate to transmit particular information.

A middleware architecture using the MVC pattern is proposed in [12]. The proposed middleware architecture for pervasive computing supports the architectural quality attributes of adaptability, availability, security, and modifiability. All these requirements are ensured using the MVC design pattern as discussed in [12].

Fusion from ORACLE is a Service Oriented Architecture (SOA) middleware for healthcare integration, connecting administrative and clinical processes. Through the use of Fusion Middleware, Oracle helps organizations to reliably exchange information while adhering to important industry standards and initiatives. This enables organizations to lower operating costs and accelerate time-to-market by delivering a consistent user interface, security architecture, management console, and monitoring environment. The Fusion middleware is designed to correlate clinical data, link applications, and comply with the myriad challenges of this highly regulated, data-intensive industry. Smoothing data interchange helps streamline every phase of the healthcare lifecycle from initiation, eligibility, and enrollment to service delivery, program analysis, and reporting [13].

There exist many other pervasive computing middleware solutions e.g. *SeSCO*, *OneWorld*, and *AoC* to name a few [22]. Though the existing middleware solutions are useful, they themselves have varied features and contribute partially, to context, data, or service management related application developments. Most of them are oriented toward mobile applications and do not provide abstraction for many types of applications. There is no single middleware solution that can address a majority of pervasive computing application development issues, due to the diverse underlying challenges. Also, there is a huge scope for research in the area of RFID and WSN middleware and applications, and many solutions were developed to cope with the technology –related challenges.

As compared to the related work described herewith, FlexRFID middleware has many distinguishing aspects. It provides the applications with a device neutral interface to communicate simultaneously with many different hardware devices, creating an intelligent network of RFIDs, sensors, and any other type of automation devices. The policy-based Business Rules Layer allows FlexRFID middleware to enforce all data processing capabilities by applying the backend applications defined rules. This enforces security by restricting access to data only to the applications or users that satisfy certain conditions stated in the policies. Also the modular layer of the middleware allows seamless integration of different types of enterprise applications, which makes it a general middleware not related only to one application domain. We present hereafter the middleware architecture, and show how business rules are modeled and applied by the middleware using some healthcare scenarios.

## III. PROPOSED POLICY BASED MIDDLEWARE AND ITS SUBSYSTEMS

### A. *FlexRFID Middleware Overview*

The FlexRFID middleware as described in [14] is a multi-layered middleware consisting of Device Abstraction Layer (DAL) which abstracts the interaction with the physical network devices, Business Event and Data Processing Layer (BEDPL) which provides data services such as dissemination, aggregation, transformation, and duplicate removal, Business Rules Layer (BRL) which is a policy-based management engine that defines the rules that control resources and services of the FlexRFID middleware, and Application Abstraction Layer (AAL) which provides a high level of software abstraction that allows communication among the enterprise applications and the FlexRFID middleware. FlexRFID was integrated in many domains: library management [14], inventory control with Opentaps software [15, 16], and healthcare [17].

The BRL is a policy-based management engine that defines the rules that grant or deny access to resources and services of the FlexRFID middleware, and enforces different types of policies for filtering, aggregation, duplicate removal, privacy, and different other services. This is achieved by determining the policies to apply when an application requests the use of a service in the BEDPL. Hereafter, we give more details about the BRL, policies architecture, and examples of policies representation.

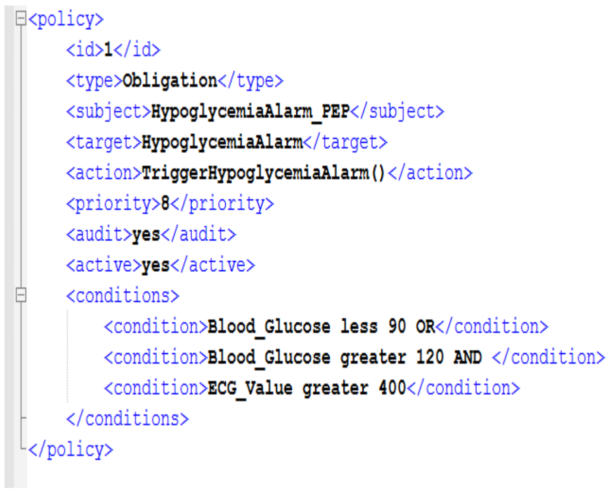
### B. *Policy-Based Business Rules Management Layer*

#### 1) *Policy Types and Structure*

Software policies have been widely used to provide security for WSN as in SecSNMP [18]. Policies are operating rules used to maintain order, security, consistency, or other ways of successfully achieving a task. In our middleware architecture we are using the types and structure of policies defined for the system in [19]. There are basically two types of software policies. *Authorization policies* are rules that are usually enforced in access control systems. In the case of the FlexRFID middleware, authorization policies would be rules defined by the application to enforce or deny access to certain data if a certain set of conditions is fulfilled. *Obligation policies* refer to actions to be enforced when a set of predefined conditions is fulfilled or a change in the context happens. For FlexRFID middleware, an obligation policy would be to trigger the duplicate removal service for an application only 5 minutes of data reads. The policy specification language, based on which we have decided to model our policies is Ponder, and we have chosen XML to represent policies due to its ease of editing and use.

As presented in [19], the policy has nine main attributes. The *policy ID* is a unique identifier of the policy and helps in the search operation. The *type* of policies refers to whether we deal with an authorization or obligation policy. The *subject* of the policy is the entity that enforces the action of the policy, and the *target* is the entity on which the policy's action is enforced. Usually the action of the policy is a call for a method that belongs to the target. The *priority* attribute of the policy is used to solve the problem of conflicting policies. The *audit tag* of the policy allows the system to keep track of the triggered policies and their context, and the *active tag* specifies if a policy is active or not. One of the most important attributes of the policy is the *set of conditions* which refers to conditions under which the policy is triggered. These are expressed using first order logic and comparison operators. Context information that is included in the policies is part of the condition set. For example this context information could refer to time, location, type of application, and role of users.

Fig. 1 below shows an example policy for blood glucose management, with the attributes mentioned above. In this example, policy with ID 1 is an obligation policy that triggers an alarm for a specific group of physicians taking care of a diabetic patient facing a hypoglycemia state. When the blood glucose (Blood_Glucose) and heart rate (ECG_Value) values are communicated by the sensors, they are checked to see whether they satisfy the set of conditions mentioned in the policy.



```
<policy>
    <id>1</id>
    <type>Obligation</type>
    <subject>HypoglycemiaAlarm_PEP</subject>
    <target>HypoglycemiaAlarm</target>
    <action>TriggerHypoglycemiaAlarm()</action>
    <priority>8</priority>
    <audit>yes</audit>
    <active>yes</active>
    <conditions>
        <condition>Blood_Glucose less 90 OR</condition>
        <condition>Blood_Glucose greater 120 AND </condition>
        <condition>ECG_Value greater 400</condition>
    </conditions>
</policy>
```

Fig. 1. Sample policy structure for hypoglycemia management

The policy in Fig. 1 states that If the condition set is met, the Trigger_HypoGlycemiaAlarm() method of the doctors' application is called to inform them about this emergency case.

#### 2) *Policy-Based BRL Architecture*

In general a policy management system is composed of three main entities; the PDP (Policy Decision Point), the PEP (Policy Enforcement Point), and the PIB (Policy Information Base). The PDP is responsible for taking the decision whether to allow or deny an action based on the request details and the policies available in the PIB. The PIB refers to the database containing all the system policies. Once the action is processed and selected by the PDP, it sends a message to the PEP which is responsible for enforcing the action on the target. In our system we kept the same components and added some others that ease management of policies and adapt the policy management system to the middleware.

Fig. 2 shows the architecture of the FlexRFID middleware adapted to put more emphasis on the components of the policy-based BRL, and explain interaction between them.

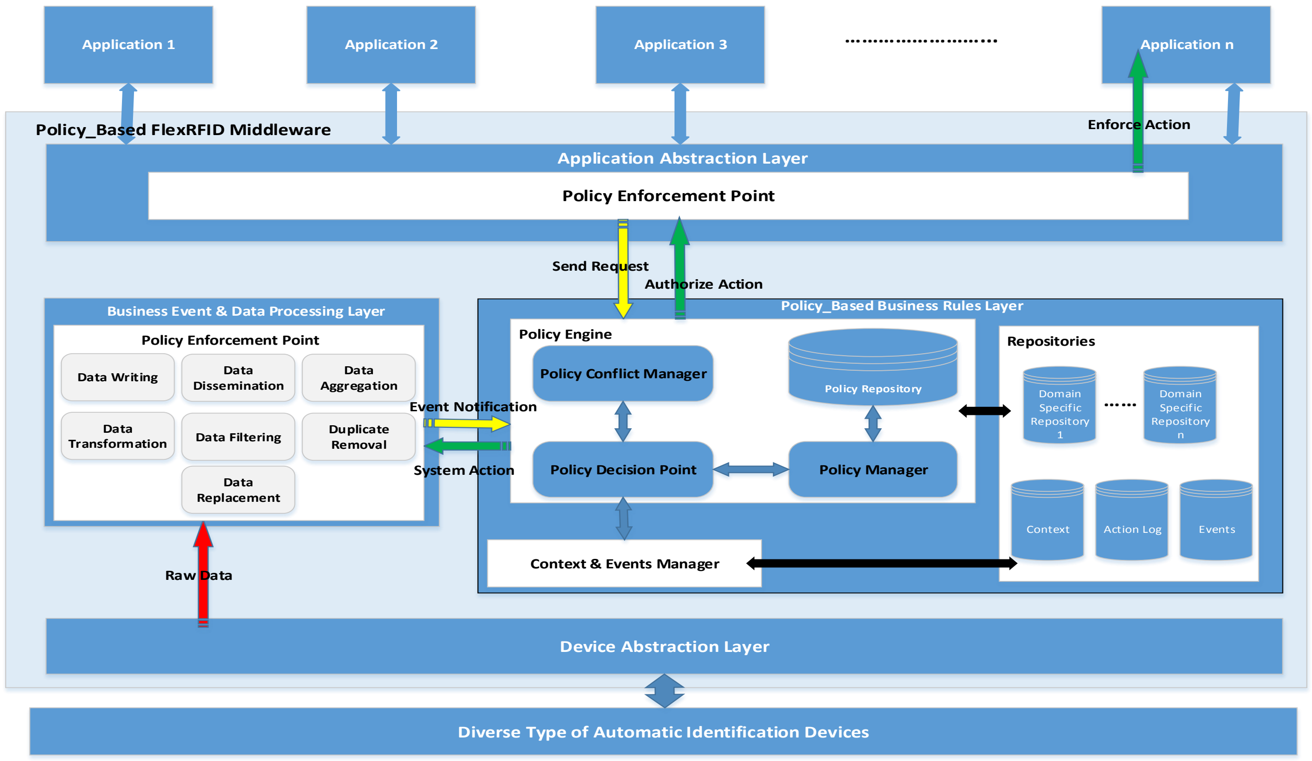


Fig. 2. Policy-based Business Rules Layer (BRL) architecture of the FlexRFID middleware

The BRL contains three main components: the Policy Engine, the Context and Events Manager, and the Repositories. The *policy engine* is responsible for policy management of the middleware and has four main components namely the PDP, the policy manager, the policy conflict manager, and the policy repository. The *policy manager* accesses the policy repository to read the policies and organize them for use by the two other components; the PDP and the policy conflict manager. The policy manager is also responsible for updating the policy repository by adding new policies when needed by the applications or when new applications subscribe to the middleware. The *policy conflict manager* sorts the list of policies given by the PDP in increasing order of priority, and triggers the policy with the highest priority for a specific event. The PDP is responsible for evaluating the policies and deciding whether a policy action is to be triggered or not. The PDP is either triggered by an incoming request or event that is external to the system for example new sensor data, or by an internal event that is a notification from the context manager of a change in the environment's context. The *policy repository* contains all the obligation and authorization policies of the system.

There are four main repositories in the BRL. The *context repository* contains all context information that is of use to our system such as time, and location and is used by the context manager. The *actions log repository* contains a log for every policy that has been triggered. The log helps providing with accountability such as the requester identification, information about the type of the policy; whether it is an obligation or an authorization policy, the subject and the target. The *events repository* contains all the events encountered by the middleware for example the read of a new RFID tag, or new sensor data detection, a change in the object location, etc. The *domain specific repositories* contain data related to a certain application domain. In the case of healthcare the domain specific repository would be the *Electronic Health Record* (EHR).

The PEP stands both at the AAL and BEDPL of the middleware because it is the one responsible for performing actions that are specified in the system and applications policies. At registration phase with the middleware, the application loads its set of policies to the middleware so that both PEPs can enforce actions specified in the policies when the condition set is met.

The sequence diagram in Fig. 3 shows how the different components of the policy based BRL interact with each other and with the remaining middleware components when new data is detected. Once the automatic identification device detects the data, they are sent to the service PEP at the level of the BEDPL. The PEP in its turn sends the event to the PDP.

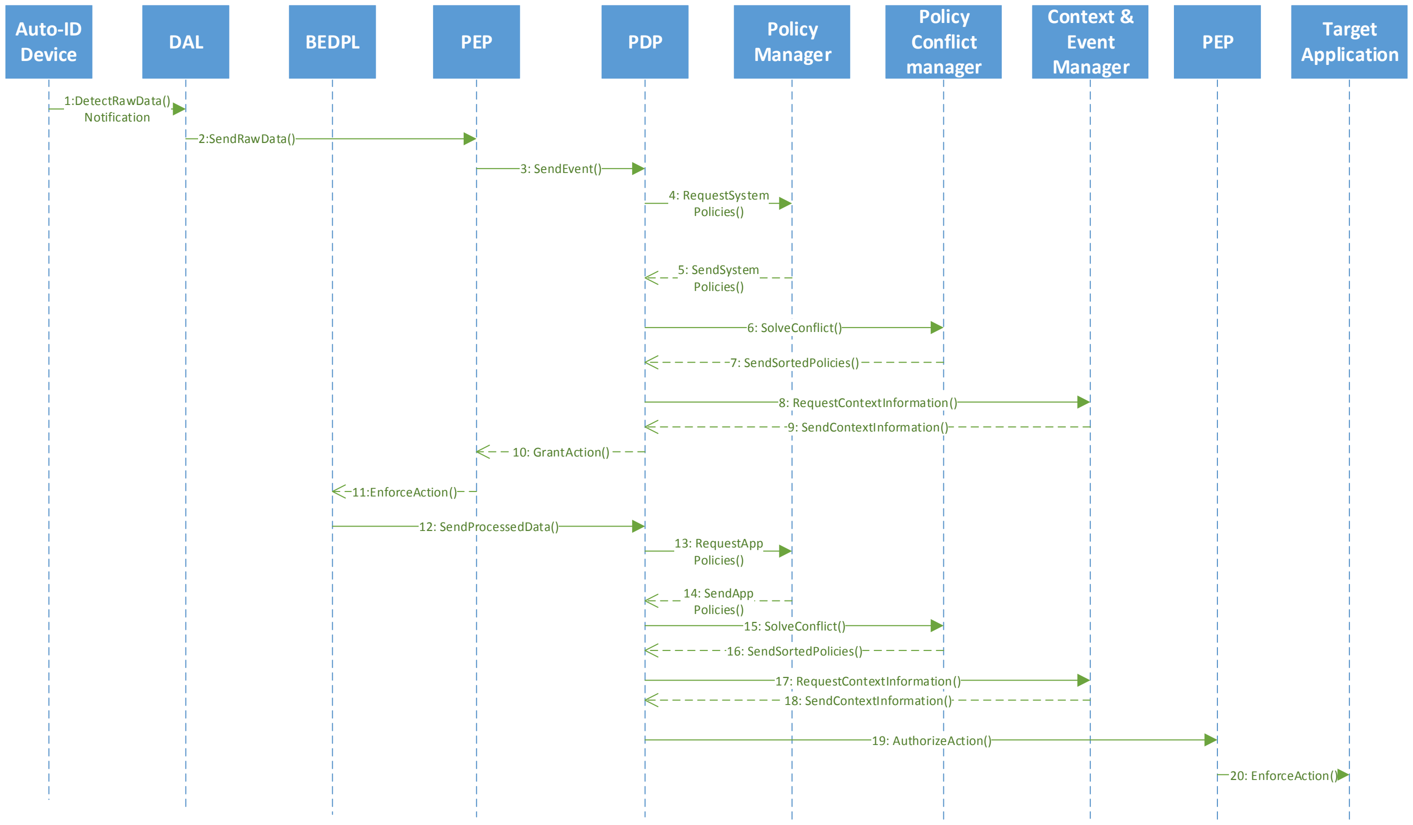


Fig. 3. Policy-based data processing flow in FlexRFID middleware

The PDP requests system policies in charge of controlling the BEDPL services from the policy manager, and requests solving conflict from the policy conflict manager in case of conflicting policies. PDP also requests the context information from the context manager if available. After resolving conflict and acquiring context information the PDP grants the action to the service PEP which in turn enforces the target service at the level of BEDPL. The BEDPL sends these processed data to the PDP for application of further policies related to the backend application business rules. The PDP checks again the application policies, solves conflict, and requests context information if needed. Once the policies are applied it authorizes the action for the PEP at the application level, and the PEP enforces the action on the target application. As can be seen from the sequence diagram in Fig. 3 policies are divided logically into two types: system policies related to the middleware services, and application related policies which refer to any additional rules that need to be applied to data after the first processing by the BEDPL.

## IV. Healthcare Scenarios

Healthcare is a highly collaborative environment which requires information sharing to provide care. Different services take part of the care process; a patient gets admitted to a hospital, doctors prescribe treatment for their assigned patients, prescriptions are sent to pharmacy, and specific information is sent to accountants for the billing purpose.

With the provision of sensor and automatic identification technologies, we are more talking about *homecare* and *remote care* outside of traditional care institutions. This reduces the need for human intervention, alerts of particular incidents, and provides detailed representations of patient state in real-time. The patient becomes empowered and feels more independent while receiving more information to assist him/her in self-care [20].

Remote care environments are dynamic and each instance of them is created on demand to cater for specific aspects of patient care. The instance is customized to a particular situation in terms of management policies and entities involved. Remote care requires entities that deliver services depending on their role in the care process, and requiring notification of events as they occur. Events include either actions performed such as a patient taking a drug, data notification such as a sensor monitoring a vital sign, and state changes such as a detection of an emergency case.

In a healthcare application, the FlexRFID middleware should support the real-time dissemination of events to the interested entities, while providing support for heterogeneous devices capturing data and means to control information processing and disclosure. This involves loading the policies into the middleware by the different healthcare entities or applications in order to define the situations for data release.

Example of policies in healthcare may include the following:

**Subscription for particular events**: this policy is used in case a user may request particular information for delivery as it occurs. For example a policy might allow a doctor to receive treatment events only for patients that he treats.

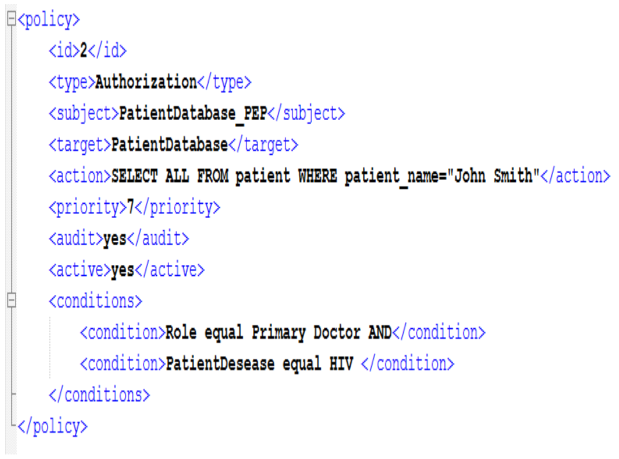
**Data access control and event restriction**: this policy defines the conditions under which certain data or events are not delivered for a subscribed application. For example a policy might prevent a nurse from receiving a patient's HIV treatment data, while allowing this for his caring doctor. Another policy might allow a physician to modify any medical record for which he or she is designated as primary physician. Also a policy can state that access to a medical record is allowed for five times only and each access expires after one minute.

**Data and event transformation**: this policy involves modifying the event type or attributes to better satisfy the requirements of the subscribed entity or its current context. It is a context-aware policy that accounts for emergency cases. For example if a doctor has no relationship with a patient, their subscription can be deactivated. Another example is to transform glucose reading to an alert if it is too high for two successive days.

Hereafter we describe the application of our policy-based data control middleware to two healthcare scenarios.

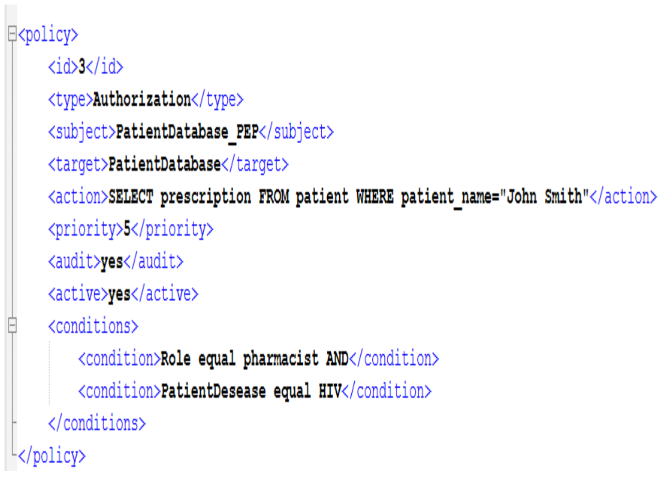
## A. *Drug prescription control scenario*

A nurse may prescribe some controlled drugs in certain circumstances to a patient. This prescription must be validated by the primary treating doctor, who must access to all details about the patient care and history. The prescription must flow to the pharmacy without any details of the patient and the reason for prescription. An auditor must monitor the supply of the controlled drug and must not receive patient specifics. This emphasizes role based access control to patient data. Fig. 4 and Fig. 5 below show policies used for the above scenario to control drug prescription.



```
<policy>
    <id>2</id>
    <type>Authorization</type>
    <subject>PatientDatabase_PEP</subject>
    <target>PatientDatabase</target>
    <action>SELECT ALL FROM patient WHERE patient_name="John Smith"</action>
    <priority>7</priority>
    <audit>yes</audit>
    <active>yes</active>
    <conditions>
        <condition>Role equal Primary Doctor AND</condition>
        <condition>PatientDesease equal HIV </condition>
    </conditions>
</policy>
```

Fig. 4. Data access control policy to patient's data by the primary doctor



```
<policy>
    <id>3</id>
    <type>Authorization</type>
    <subject>PatientDatabase_PEP</subject>
    <target>PatientDatabase</target>
    <action>SELECT prescription FROM patient WHERE patient_name="John Smith"</action>
    <priority>5</priority>
    <audit>yes</audit>
    <active>yes</active>
    <conditions>
        <condition>Role equal pharmacist AND</condition>
        <condition>PatientDesease equal HIV</condition>
    </conditions>
</policy>
```

Fig. 5. Data access control policy to patient's data by the pharmacist

## B. *Location tracking and emergency management scenario*

Location sensors or RFID tags are commonly used in remote care to detect the location of patient. Tracking location is important in detecting emergencies especially for elderly care [21] (are patients in bed? Did they fall on the floor?), and in quickly dispatching the ambulance to the target location. Patients generally care about privacy and want their exact location to be obscured. A policy might specify that location information should not be transmitted unless it is an emergency situation, and for some defined entities in the care process for example for doctors and close relatives only.

Referring to the hypoglycemia policy modeled in Fig. 1, diabetes self-management can be easily deployed using policies. Sensors can be used to take the diabetic patient measurements like blood glucose, blood pressure, amount of meals, amount of exercise, and location. The measurements taken by the sensors can be sent to the middleware, aggregated, checked for the specified thresholds set by the doctors in the diabetes management application policies, and sent to the specialized doctor in real-time so that he/she can send advice to the patient. If the blood glucose and pressure are noticed to be too high over a certain period of time (a day for example), the doctor may advice the patient to go for a workout, or to take an additional insulin injection to lower the blood glucose. In case of hypoglycemia the doctor should advice the patient to stay in bed and eat more in order to increase his/her blood glucose. In this case if the blood glucose readings are too low and the RFID tag attached to the patient sends location change information, the middleware should trigger an alarm to the doctor so that he sends a dedicated medical team to take care of the situation, because the patient may fall down while moving in a hypoglycemic state.

The examples above highlight key features of policy based management in the middleware. Information is released depending on the context and situation only to the interested and eligible entities. An event can be interesting to many entities but its visibility is controlled by the policies defined beforehand. This ensures security and privacy concerns.

## V. FLEXRFID MIDDLEWARE EVALUATION

### A. *Device Evaluation*

Device evaluation corresponds to the portability metric of ISO/IEC 9126 standard [23] in terms of multiple devices support. This metric evaluates the heterogeneity, and scalability of the middleware as the number of devices increase. We mean by scalability here the durability of stable status of the middleware when certain conditions are met, in this case when the number of devices increases.

FlexRFID middleware provides the RFID applications an interface to RFID hardware and other sensors and automatic identification devices, called "*Device Abstraction Layer (DAL)*". Generally the hardware devices are accessed by a set of APIs provided by the device manufacturer, which are specific to each device. In our implementation of the DAL we used wrappers for the manufacturer provided reader APIs, in order to make the reader accessible through FlexRFID. These wrappers call the reader specific API to implement the desired functionality.

The FlexRFID middleware was tested with *Intermec Fixed IF61 reader* which was available in our lab. This reader's DLL was loaded to the FlexRFID. After set up of each connection to the IF61 reader, a handle on it is used to support all further communication with this reader. Further devices' DLLs must be identified and added to the FlexRFID, in order for the middleware to support communication with them.

### B. *Application Evaluation*

Application evaluation corresponds also to the portability metric of ISO/IEC 9126 standard in terms of heterogeneous system and application support. It evaluates the level of abstraction of the middleware in terms of providing standard APIs to communicate with multiple backend applications, and also scalability of the middleware as the number of connected applications, the number of policies loaded by the applications, and the number of requests from the applications increases.

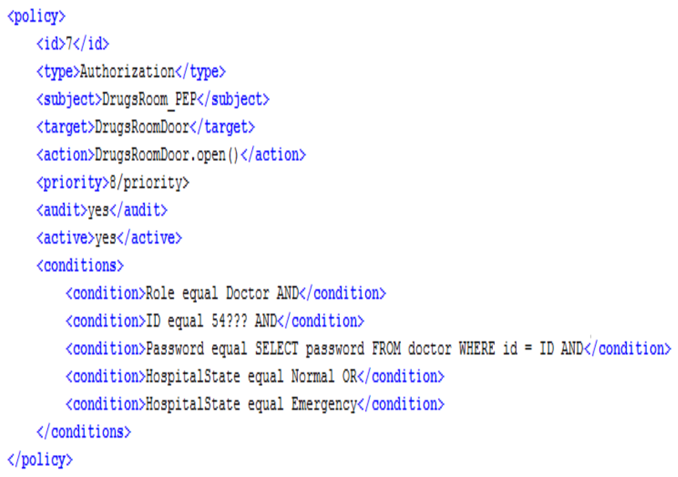
FlexRFID provides through the *Application Abstraction Layer (AAL)* a generic class that should be implemented by all applications that want to connect to the middleware. This class provides functions to access the general operations done by all RFID applications such as reading / writing data, duplicate removal, first level filtering, etc. Domain specific data treatment such as data transformation to some complex business events is either expressed through policies so that the middleware applies the application rules on data before dissemination, or done at the level of the application itself.

FlexRFID was tested with a smart library application prototype [24], integration with OpenTaps for inventory control [16], and we have identified scenarios for healthcare domain integration that we need to simulate in a healthcare application prototype [17].

### C. *Security and Privacy Evaluation*

The security and privacy evaluation is meant to assess the security and privacy of the pervasive middleware, and how it protects the applications' sensitive data when needed through the use of policies. This is achieved by generating a scenario in which access to data is restricted to specific parties, and testing how the middleware deals with this access control policy.

The policy example that we have used in our scenario is shown in Fig. 6 below. This policy maps a business rule from a hospital that says that only doctors who have ID starting with "54" and who have correctly authenticated (ID + Password authentication) can access the "DrugsRoom". The policy restricts access to that specific room to a certain category of hospital employees. The policy enforces the restriction by allowing use of context information such as the role of the employee (Role Based Access Control), and the state of the hospital (Normal state or emergency state). In our test we have generated tags which lead to the creation of requests. The requests contained different specificities which did not match the conditions of the policy responsible for the management of the door leading to the "DrugsRoom". Therefore, none of the requests led to the opening of the door. It is to specify that we have conducted our tests on a local host where no external threats exist.



```
<policy>
    <id>7</id>
    <type>Authorization</type>
    <subject>DrugsRoom_PEP</subject>
    <target>DrugsRoomDoor</target>
    <action>DrugsRoomDoor.open()</action>
    <priority>8/priority>
    <audit>yes</audit>
    <active>yes</active>
    <conditions>
        <condition>Role equal Doctor AND</condition>
        <condition>ID equal 54??? AND</condition>
        <condition>Password equal SELECT password FROM doctor WHERE id = ID AND</condition>
        <condition>HospitalState equal Normal OR</condition>
        <condition>HospitalState equal Emergency</condition>
    </conditions>
</policy>
```

Fig. 6. Healthcare scenario for access control policy.

### D. *Context Evaluation*

The context evaluation assigns metrics to the contexts available in the environment in which the FlexRFID middleware is tested. The context can be time, location, user activities, objects movements, etc. The context evaluation can take place by generating different scenarios by the user and checking the application for context-awareness, for example whether it adapts to changes in context seamlessly. Location tracking and emergency management in healthcare can be a great scenario for showing the middleware's adaptability to context.

Location sensors or RFID tags are commonly used in remote care to detect the location of patient and measure vital signs like blood pressure, temperature, blood glucose, etc. Using policies, in an emergency case, Information is released depending on the context and situation, and only to the interested and eligible entities. An event can be interesting to many entities but its visibility is controlled by the policies defined beforehand. This ensures as well the security and privacy concerns.

### *E. Performance Evaluation*

Response time is among the most important performance parameters. It is the amount of time beginning upon sending some request to the middleware that performs required operations over the massive data gathered from the devices, to receive response from the middleware and disseminate the processed data to the interested applications. Response time measures the delay of query results.

We are interested in two different performance testing cases. First, we have the case where the system is dedicated to one client application. This means that all the policies in the system belong to the same application. Our measure is the time necessary for a request to receive a response depending on the number of policies specified by the client. In the second test, we have decided to have a fixed total number of policies. However, the policies would belong to different client applications. Since we are using one request to test for performance, we wanted to see how the total number of policies affects the system performance. We are in the process of testing these two test cases and getting the performance results.

## VI. Conclusion and Future Work

Policy based FlexRFID middleware was developed to cater to the need of many applications that need to take advantage of the ubiquitous computing technologies in order to automate some data processing. We have outlined the need for using this middleware solution for remote healthcare and the flexibility it offers for handling different scenarios thanks to its policy based Business Rules Layer.

We have implemented a working prototype of the middleware including the policy engine and context management system. The next step is to choose one of the defined scenarios, generate the corresponding events, load the defined policies in the middleware, and see how it handles the data related services, access control to data, and ensures patients' privacy. A further testing of the middleware will include performance and response time testing for the test cases defined above. For example we can see how the middleware behaves when the number of policies loaded by a specific application increases, or when the services need to gather data from many devices for processing, etc.

The current version of the FlexRFID middleware offers only basic support for security through the use of access control policies because the project intentionally focused on the described concepts and policies definition. More work on the security features is needed in the future for e.g. security at the level of tags or sensor nodes, application authentication, reader/tag authentication implementation, etc. Ongoing work will also consider the autonomy evaluation which deals with the following parameters: self-configuration, resource management, failure tolerance, high availability, and decision making. The FlexRFID will also be integrated in the cloud, provide a foundation for enabling applications to flexibly use services provided in the cloud, and automatically adapt the usage of cloud-based services depending on application policies and context considerations. As the technology matures in the future, FlexRFID may integrate other types of devices, and handle new services and applications.

## References

[1] M. Amimian, and H. R. Naji, "A hospital health care monitoring system using wireless sensor networks," J Health Med Inform, Vol. 4, No. 2, 2013. Available: http://www.omicsonline.org/a-hospital-healthcare-monitoring-system-using-wireless-sensor-networks-2157-7420.1000121.pdf

[2] W. Yao, C. H. Chu, and Z. Li, "The use of RFID in healthcare: Benefits and barriers," IEEE International Conference on RFID-Technology and Applications (RFID-TA), Guangzhou, China, June 17-19, 2010.

[3] M. Alam, M. Hafner, M. Memon, and P. Hung, "Modeling and enforcing advanced access control policies in healthcare systems with sectet," Workshop on Model-Based Trustworthy Health Information System (MOTHIS), 2007.

[4] Sun Microsystems, "Sun Java™ system RFID software 3.0 developer's guide," February 2006, [Online], Available: http://download.java.net/general/sun-rfid/Release30/Docs/Developers_Guide_819-4686.pdf

[5] Microsoft, "BizTalk server 2006 developer productivity study," January 2007, [Online], Available: ww.microsoft.com/biztalk/en/us/rd.aspx

[6] B. S. Prabhu, X. Su, H. Ramamurthy, C. Chu, and R. Gadh, "WinRFID – A middleware for the enablement of Radio Frequency Identification (RFID) based applications," Wireless Internet for the Mobile Enterprise Consortium (WINMEC), Los Angeles, December 2005. Available: http://citeseerx.ist.psu.edu/viewdoc/download?rep=rep1&type=pdf&doi=10.1.1.91.8928

[7] C. Floerkemeier, C. Roduner, and M. Lampe, "RFID application development with the Accada middleware platform", IEEE Systems Journal, Special Issue on RFID Technology, Vol. 1, No. 2, December 2007. Available: http://www.vs.inf.ethz.ch/publ/papers/floerkem-rfidap-2007.pdf

[8] J. Radhika, and S. Malarvizhi, "Middleware approaches for wireless sensor networks: an overview," International Journal of Computer Science Issues (IJCSI), 2012, Vol. 9, N° 3, pp: 224-229.

[9] K. Sohraby, D. Minoli, and T. Znati, "Middleware for wireless sensor networks", Wireless Sensor Networks Technology, Protocols, and Applications, John Wiley & Sons, 2007. Available : http://www.knovel.com/web/portal/browse/display?_EXT_KNOVEL_DISPLAY_bookid=4513&VerticalID=0

[10] S. Hadim, and N. Mohamed, "Middleware for wireless sensor networks: a survey," IEEE International Conference on Communication System Software and Middleware (COMSWARE), New Delhi, India, January 8-12, 2006.

[11] J. Singh, and J. Bacon, "Event-based data dissemination control in healthcare," Electronic Healthcare, 2009, vol. 0001, pp: 167—174.

[12] J. E. Bardram, and H. B. Christensen, "Middleware for pervasive healthcare - a white paper," Aarhus Denmark, 2001. Available: http://www.pervasivecomputing.dk/publications/files/wmmc2001.PDF

[13] ORACLE, "Oracle SOA Suite for Healthcare Integration," October 2013, [Online], Available: http://www.oracle.com/us/products/middleware/soa/soa-suite-for-healthcare-wp-2046692.pdf

[14] M. E. Ajana, M. Boulmalf, H. Harroud, and M. Elkoutbi (2011). RFID Middleware Design and Architecture, Designing and Deploying RFID Applications, Dr. Cristina Turcu (Ed.), ISBN: 978-953-307-265-4, InTech, DOI: 10.5772/16917. Available: http://www.intechopen.com/books/designing-and-deploying-rfid-applications/rfid-middleware-design-and-architecture

[15] M. E. Ajana, H. Harroud, M. Boulmalf, and H. Hamam, "A policy based event management middleware for implementing RFID applications," in Proceedings of the fifth IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Marrakech, Morocco, October 2009.

[16] M. E. Ajana, H. Harroud, M. Boulmalf, and M. El Koutbi, "FlexRFID middleware in the supply chain: Strategic values and challenges,"

Contemporary Challenges and Solutions for Mobile and Multimedia Technologies. IGI Global, 2013. doi:10.4018/978-1-4666-2163-3.ch010.

[17] M. E. Ajana, H. Harroud, M. Boulmalf, M. Elkoutbi, A. Habbani, “Emerging wireless technologies in e-health trends, challenges, and framework design issues,” Proceedings of International Conference on Multimedia Computing and Systems, International Conference on Multimedia Computing and Systems (ICMCS), Tangiers, Morocco, October 10-12, 2012.

[18] Q. Wang and T. Zhang, “Sec-SNMP: Policy-based security management for sensor networks”, in Proceedings of the International Conference on Security and Cryptography (SECRYPT), 2008, pp. 222-226.

[19] M. Chraibi, H. Harroud, and A. Karmouch, “Personalized security in mobile environments using software policies,” Proceedings of the 9th International Conference on Advances in Mobile Computing and Multimedia (MoMM), Hue City, Vietnam, December 5-7, 2011.

[20] M. Ahlsen, S. Asanin, P. Kool, P. Rosengren, and J. Thestrup, “Service-oriented middleware architecture for mobile personal health monitoring,” Proceedings of the 2nd International ICST Conference on Wireless Mobile Communication and Healthcare (MOBIHEALTH 2011), Kos, Greece, October 5-7, 2011.

[21] Y. Y. Ou, P. Y. Shih, Y. H. Chin, T. W. Kuan, J. F. Wang, and S. H. Shih, “Framework of ubiquitous healthcare system based on cloud computing for elderly living,” Proceedings of International Conference on Signal and Information Processing Association Annual Summit and Conference (APSIPA), Kaohsiung, Asia-Pacific, October 29 - November 1, 2013.

[22] V. Raychoudhurya, et al., “Middleware for pervasive computing: a survey,” Pervasive and Mobile Computing, 2013, Vol. 9, No. 2, pp. 177–200.

[23] C. Park, et al., “RFID middleware evaluation toolkit based on a virtual reader emulator”, in proceedings of the 1st International Conference on Emerging Databases, Busan, Korea, August 2009.

[24] M. E. Ajana, et al., “FlexRFID: A flexible middleware for RFID applications development,” in proceedings of the 6th International Wireless and Optical Networks Communications (WOCN) Conference, Cairo, Egypt, April 2009.

[25] G. Schiele, et al., “Pervasive computing middleware”, *Handbook of Ambient Intelligence and Smart Environments (AISE)*, Springer, US, 2010, pp. 201-227.