Personalized Home-Networks

Identity-driven network behaviour and configuration

Soler, José; Gandy, Michael

Published in:
6th International Conference on Networking (ICN07)

Link to article, DOI:
10.1109/ICN.2007.73

Publication date:
2007

Document Version
Publisher's PDF, also known as Version of record

Personalized Home-networks

Identity-driven network behaviour and configuration

José Soler
Communication (COM) Department, Networks Area
Technical University of Denmark (DTU)
Lyngby, Denmark
jsoler@com.dtu.dk

Michael Gandy
TELETEL S.A.
Athens, Greece
M.Gandy@teletel.gr

***Abstract*—The paper provides details of a home-networking architecture based on an enhanced residential gateway. Initially the need for mechanisms allowing user-dependent network behavior is described and afterwards details of an initial implementation are provided in terms of architectural description, enabling technical components and interaction.**

Keywords- home network, residential gateway, personalization, identity management, policy, profiles, JAXB, XACML, OSGi.

I. INTRODUCTION

As users of multiple devices we are used to operate them according to our preferences and privileges. Take for example a personal computer with accounts for multiple users. Each user accesses to the computer with specific credentials, which determine its identity in the computer. Based on these credentials the user can access some of the applications and related peripherals in the computer, while other remain hidden to him or its functionality is reduced according to the rights granted by the computer's administrator and its administration policies. The user himself can also configure the applications he has access to, according to its preferences, i.e. application's behavior configuration, content formatting and display or preferred contents of media-reproduction.

This user-driven configuration and behavior can be extended to a home network environment where different inhabitants share the different home appliances (multimedia devices and white-goods). The access to them and its functionality can be "personalized" based on different policies as well as the users' own configuration. This is the idea behind the EU IST project ESTIA, Enhanced Networked Architecture for Personalized Provision of AV Content within the Home Environment (www.ist-estia.org): to define and demonstrate a home networking architecture in which the networked elements are accessed, and provide different functionalities, based on personalization and identity management.

The rest of the paper obeys to the following organization. Section II presents the proposed architecture based on an enhanced residential gateway. Section III, IV, V and VI describe each of the fundamental technologies the Residential Gateway is based on. Section VII closes the paper with a brief conclusion.

II. GENERAL HOME NETWORKING ARCHITECTURE

Figure 1 presents an oversimplified view of ESTIA's home network. Basically a central element, the so-called residential gateway (RG) allows communication between the home network and other external networks. Besides this basic gatewaying operation, the RG performs a central role in the home network, as connectivity hub and controller of the devices connected to the network. As displayed in the figure, not only traditional audiovisual devices such as TV or Video/DVD players and recorders are considered networked devices but also white-goods such as cook, laundry-machine or refrigerator , as well as other devices such as lighting switchers are under ESTIA's consideration of networked devices.

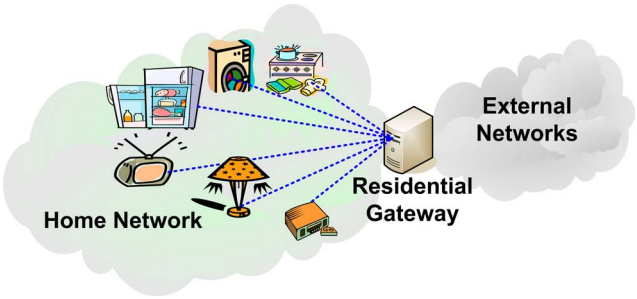

Figure 1. Simplified Home Network architecture

The connectivity between these heterogeneous devices and the residential gateway is one of the key elements of the ESTIA project and technologies such as Powerline, Konnex (KNX) and Ethernet are considered. Nevertheless the focus of this paper is not on the enabling technologies in the physical layer and their interoperation, but on the software mechanisms that allow user dependent use of the different elements, regardless of the connectivity mechanisms towards the RG.

In order to illustrate the "personalization" in the access and use of devices, consider the following example: A user categorized as "adult" could have granted access to a cooking device, but another user categorized as a "child" would f.ex. only be allowed to access the device to switching it off. This would allow that no children get hurt or cause a problem if f.ex. turning the fire on, without being noticed by an adult, but on the other hand he can prevent something to get burnt when the cooking device is on and no adult is aware of the problem.

Proceedings of the Sixth International Conference on Networking (ICN'07)
0-7695-2805-8/07 $20.00 © 2007 IEEE

Similar actions and restrictions could be defined in any device based on differentiation of user groups and definition of different policies for the different groups or individuals [1].

This so-called "ESTIA personalization" is based on a set of processes that can be summarized as:

- Profiling devices: defining its operations, access status and categorization in predefined groups (i.e. white goods, audiovisual, other).
- Profiling users: defining specific information and categorization depending on predefined roles and categories (i.e. administrator vs. inhabitant vs. guest or adult vs child)
- Definition of policies: defining permissions per device, functionalities and groups.
- Identity Management: maintenance and verification and of user's identity attributes granting access to the network and its devices.
- Policy Management and Enforcement: mechanisms to validate requests for device access according to the defined policies and the attributes of the users requesting them.
- Profile Management: storage and modification of the defined profiles for users and devices.
- Integration of the different process in a single homogeneous system.

The following sections provide a description of the mechanisms used within ESTIA for the implementation of the previous processes.

III. PROFILE DEFINITION AND STORAGE

At the beginning of the ESTIA design process, it was decided that the user and device profiles would not be stored as entries in a Data Base, within a Database Management System (DBMS). This would only complicate the design of the RG and increase its memory / storage requirements and at the same time, it would make its implementation dependent on the chosen DBMS.

To overcome these facts it was decided that the different user and device profiles would be XML documents compliant with generic User and Device profiles specifically defined for ESTIA.

Therefore ESTIA specific User and Device profiles were defined as XML Schemas [2].

An initial basic User Profile was defined for the initial demonstration of the project (scheduled by January 2007) as an XML schema with a single element of type UserProfile. Figure 2 shows a graphical representation of the design of this type.

As appear in Figure 2, the User profile is formed by five simple elements, each of them belonging to four different types.

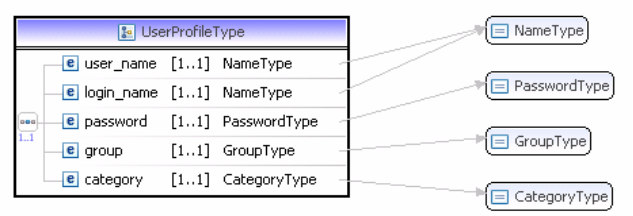

Figure 2. Demonstrative basic ESTIA's User profile

These types are defined as extensions to the basic String type within XML, as shown for the case of ESTIA's User Group Type in the following excerpt from the resulting XML Schema:

```
<simpleType name="GroupType">
<restriction base="string">
  <enumeration value="inhabitant ">
  </enumeration>
  <enumeration
     value="guest"></enumeration>
   <enumeration value="administrator">
   </enumeration>
</restriction>
</simpleType>
```

Figure 3. Excerpt for the XML definition for ESTIA's User Profile

As defined in the previous excerpt the user can belong to 3 different groups: *administrators*, *inhabitants* or *guests*.

In the same way, the Category Type is defined as an extension over the String type and as possible values: *child* and *adult*.

Different XML documents can be then defined as User profiles compliant to the previous XML schema, as shown in the example below.

```
<?xml version="1.0" encoding="UTF-8"?>
<estia_basic:user_profile … >
  <user_name>NEW_USER</user_name>
  <login_name>ali_baba</login_name>
  <password>12345678</password>
  <group>inhabitant</group>
  <category>child</category>
</estia_basic:user_profile>
```

Figure 4. Example demonstrative basic ESTIA's User Profile

In a similar way, a generic basic Device profile was defined as an XML schema with a single element of type DeviceProfile, designed as appear graphically in Figure 5.

Again the types are extensions over the basic String type, defining length and value constraints.

The GroupType would allow applying generic policies to devices belonging to a common group and is defined in this case with the following possible values: *whitegood*, *audiovisual*, *communication* or *other*.

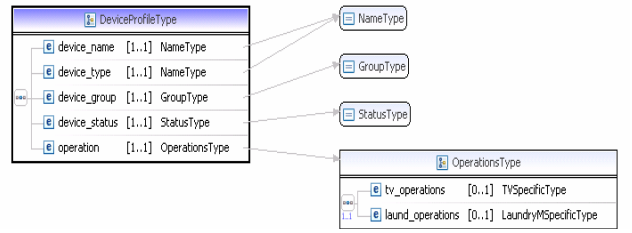

Figure 5. Example demonstrative basic ESTIA's Device Profile

For the StatusType, two generic values were defined: *ON* or *OFF*.

While these parameters are generic for any device, it was necessary to represent the possible operations over each of the devices. This set of operations is not general for every device in the home network, as the previous parameters, but on the other hand, it is device-type specific: TV sets allow different operations than refrigerators, and so on. Therefore a generic OperationsType was defined in order to cope with the specific operations of different devices.

As shown in Figure 5, the initial demonstration was planned with two devices in the home network, a TV set and a laundry machine. Consequently two new types were defined, TVSpecific and LaundryMSpecific and the generic OperationsType defined as a container of any of them.

As an example the profile for a laundry machine compliant with the previous schema definition is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<estia_device:device_profile … >
  <estia_device:device_name>
   lm_bathroom_upstairs
  </estia_device:device_name>
  <estia_device:device_type>
    laundry machine
  </estia_device:device_type>
  <estia_device:device_group>
    white_good
  </estia_device:device_group>
  <estia_device:device_status>
  ON</estia_device:device_status>
  <estia_device:operation>
  <estia_device:laund_operations>
    <estia_device:operation_status>
      Idle
    </estia_device:operation_status>
  </estia_device:laund_operations>
  </estia_device:operation>
</estia device:device profile>
```

Figure 6. Example demonstrative basic ESTIA's Device Profile

IV. Accesing Profiles: JAXB

With the presented XML schemas and extending them conveniently (by defining new specific OperationsType for each new device type) it was possible to represent the necessary information for user and device profiles as XML files compliant to those schemas.

The next step was to extract the necessary information from them. In order to do so an API was designed so that it was possible to query for any user or any device, elements of their profile. Since there was no plan to include a DBMS within ESTIA's Residential Gateway (RG) the information should be extracted from the XML-formatted profiles. In order to facilitate the creation of the mentioned API and avoid heavy involvement with the XML processing, XML to java binding techniques were used: the Java Architecture for XML Binding (JAXB) [3] was the base for these binding mechanisms and the resulting API.

As displayed in Figure 7, JAXB allows different operations over XML schemas and files that suited ESTIA's needs. The initial step with JAXB is the representation of an XML Schema as a set of Java Classes (step JAXB (a) in Figure 7). These Java clases, representation of the different elements and types of the schema, can be instantiated by a Java based application in a tree of Java objects that represent the elements of a XML document, compliant with the initial XML schema.

Therefore it is very easy to represent an XML document compliant with the initial XML schema as a set of objects with its corresponding attributes in what is termed as Unmarsahlling operation. This representation of the XML document can then be directly manipulated by an application in order to modify the different attribute values of the objects in the tree. In order to persist these modifications JAXB allows the reverse operation, termed Marshalling, to achieve an XML representation of the final tree of objects that is compliant with the initial XML Schema.

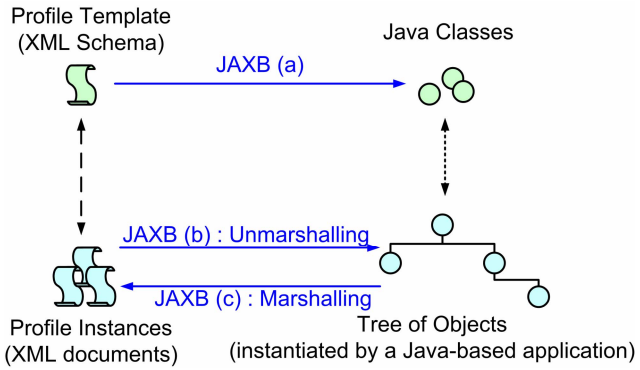

Figure 7. JAXB Operations

Based on the mentioned JAXB operations, an application was constructed for demonstrative purposes, allowing creation and modification of user and device profiles. Initially developed as a command line application, at the time of writing (early November 2006) this application is being ported to a graphical user interface based one.

V. Defining and Enforcing Policies: XACML

In order to allow a modular design and distribution of the work among the different partners of the ESTIA project it was decided that policy related information such as permissions to

access devices or its functionality should not be integrated in the User and Device profiles.

Instead, separate different policy definitions should be created and associated to the different users and devices based on their previous classification in groups and categories.

While the definition of policies did not represent any challenge (could be represented by different XML documents also), finding a technology and architectural model to enforce them was not that easy.

After investigation, the chosen candidate was Extensible Access Control MarkUp Language (XACML) [4], a language and architectural model for security policies definition and related mechanisms.

XACML provides a policy language allowing administrators to define the access control requirements for their application resources. XACML assumes an architecture according to the terminology defined by the IETF Policy Framework Working Group and the Distributed Management Task Force/ Common Information Model in [5].

As shown in Figure 8, a Policy Enforcement Point (PEP) receives request from users and resources. After checking a Policy Information Point (PIP) for the corresponding information regarding the user and the requested resources, submits the overall information to a Policy Decision Point (PDP) where the action over the corresponding resources is granted or not, depending on the policy information. The decision is sent back to the PEP who transmits it to the user in the appropriate format / protocol. As shown in Figure 8 XACML is used for the transactions between the different elements of the policy-access control architecture [6].

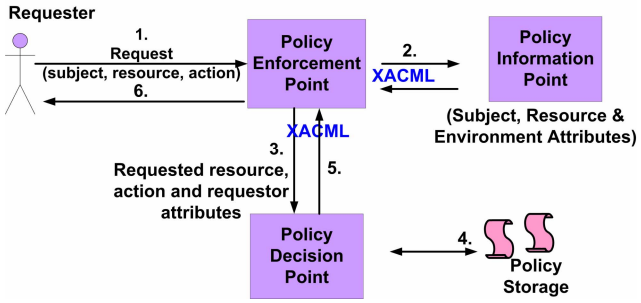

Figure 8. Policy Enforcement Architecture based on XACML flows

ESTIA's RG should integrate these elements of the policy enforcement architecture as independent software modules, which are accessed by the RG control logic. The corresponding PIP should access the different user and device profiles in order to provide the necessary information for the PEP to send the appropriate XACML request to the PDP. The reference XACML Java-based implementation by Sun [7] was identified as the base for implementation of ESTIA's policy definition and enforcement mechanisms. At the time of writing (early November 2006) initial policies have been defined and tested from a basic PEP and against a basic PDP.

VI. INTEGRATING THE DIFFERENT ELEMENTS: OSGI

A basic java-based application has been developed as initial identity management module for user's access and options configuration. A web interface has been provided to make the access easier from a remote terminal (laptop, PDA…). This application should coordinate with the ones allowing management of profiles and policies. In order to allow integration of these heterogeneous software modules, an Open Service Gateway initiative (OSGi) platform was selected as technological reference for ESTIA's Residential Gateway. The Open Service Gateway initiative (OSGi) [8] framework provides a Java technology-based lightweight container for software implementation. It handles the interactions between components and allows developers to remotely manage the entire application life cycle, including over-the-network deployment and updating. The OSGi architecture is depicted in Figure 9.

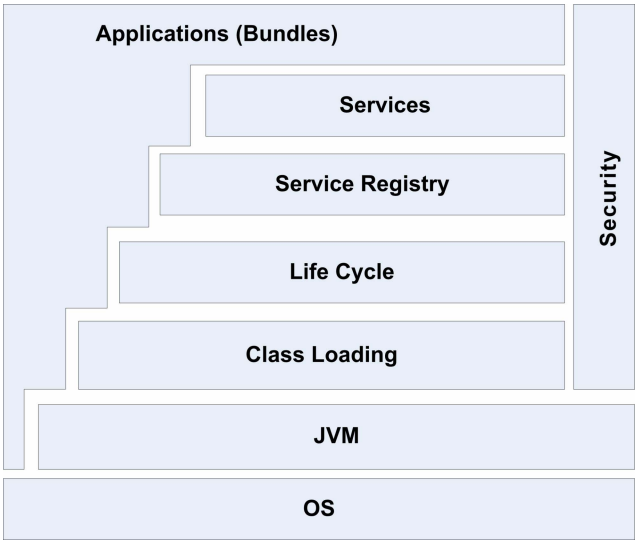

Figure 9. OSGi architecture

The software components that are maintained and managed by the OSGi framework are called bundles. They are standard Java Archive (JAR) files containing one or more classes and resources that implement one or more services. An OSGi application during its total life cycle might invoke an arbitrary number of bundles.

Besides reusability, the design of the bundle also takes the limited memory and storage capabilities of the device into consideration. All the participating bundles are not required to be available when the application is started. Instead, when a bundle is found to be missing, the OSGi service platform delegates the application to fetch the needed bundle from a management server and install it in run time. After invoking, those bundles that are never used can be uninstalled by the invoker application to free the memory or storage. The OSGi bundle is completely self-contained with all the necessary metadata in a specific file called Manifest. OSGi bundles can be installed, updated, or removed on the fly without ever having to disrupt the operation of the device. This characteristic has been the key for choosing OSGi as a

solution for ESTIA's RG: In a conventional, main-system architecture, like the one of the RG, an application should be stopped to be updated. This approach is not suitable for critical systems that have to be non-stop and highly available at any time, such as net-bank, Internet or telecommunication services. In this kind of systems the update must take place at run-time and the application should not be entirely stopped [9]. The OSGi framework provides the efficient container required for dynamic software components to correct, adapt, extend or perfect an application on the fly. OSGi is an efficient development environment because components can be added and updated at runtime [10]. It supports powerful event mechanisms: when an update is instigated, the framework deactivates the bundle to be updated, loads the new classes and calls the start method. Concerning the application architecture, it is possible to dynamically add new components, however, in case of component removal, no new instances can be created, but existing instances continue to be running. The OSGi framework enables to deploy dynamically services that for example can be found in a third-party repository and thus constantly proposed to the user the most up-to-date and adequate service without at any time requiring the stop of the system. This is particularly important in the case of a home network as a large number of devices can be connected and updates can be performed randomly at any time [11]. Therefore, the processes related to identity, profiles and policy management and enforcement will be integrated as separated OSGi bundles running within the RG, allowing dynamic deployment, and management of the different software modules: profile management, policy management and enforcement and identity management.

VII. SHOWCASE EXAMPLE

One of the project's demonstrative showcases is called "Combined Audio/Video & White-goods Administration". The basic idea behind this showcase is that a user, while using the normal functionalities of any of the audiovisual devices in the home network, for example watching an specific channel in a TV-set, is able to access the control interface for any other device in the home network, via an overlay image on that TV-set. In such a way a user can, for example, switch on or off devices such as oven, laundry machine or any other without physically accessing them and without leaving the location where the TV-set is located. The control interface and the corresponding actions over the different devices will vary depending on the privileges granted to the user, according to the existing policies in the RG and as it has been explained.

VIII. CONCLUSION

The paper has presented the basic architectural elements of a residential gateway to be integrated in ESTIA's home-network architecture. The main aim of this architecture is to allow personalized use of the devices connected to the residential gateway (RG). Details of the different architectural elements of the RG have been provided.

Study of the scalability of the architecture, considering its applicability in a home environment, is not within the purposes of the project. Therefore issues such as performance simulation of the architecture are out of the scope of this paper. Nevertheless, it can be stated that scalability of the architecture can be limited by that of the communication busses and interfaces used for interconnection, while its global performance by the hardware platform used for the RG, its memory and storage capabilities to store and simultaneously handle different user and device profiles and its object representation in memory.

The current (by November 2006) status of development after the initial design stages is promising and according to schedule: individual demonstrative basic implementations, of each of the separated software components, have been reached and the integration within an OSGi architecture is scheduled by December 2006. A first demonstration of the presented ESTIA concept and architecture is scheduled by February 2007.

ACKNOWLEDGMENT

This paper is part of the dissemination activities of the EU IST Project 27191 *Enhanced Networked Architecture for Personalized Provision of AV Content within the Home Environment (ESTIA)*. Information regarding the project, partners' contacts and public deliverables can be found at http://www.ist-estia.org/

REFERENCES

[1] ESTIA Consortium, Deliverable D2.2. "ESTIA Functional Specifications". August 2006.

[2] W3C Recomentndation , "XML Schema Part 0". 28th October 2004. http://www.w3.org/XML/Schema

[3] http://java.sun.com/webservices/jaxb/

[4] OASIS Consortium, "eXtensible Access Control Markup Language. Version 2.0". OAIS Standard. 1st February 2005. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml#technical

[5] A. Westerinen et alt., "Terminology for Policy-based Management". IETF RFC 3198. November 2001.

[6] P.Griffin, "Introduction to XACML" February 2004. http://dev2dev.bea.com/pub/a/2004/02/xacml.html

[7] Sun's XACML Implementation at http://sunxacml.sourceforge.net/

[8] Open Services Gateway Initiative (OSGi) http://www.osgi.org

[9] A.Ketfi, N.Belkhatir, P.Cunin, "Adapting Applications on the Fly" , Proceedings of the 17th IEEE International Conference on Automated Software Engineering. p. 313. September 2002.

[10] R. Hall, "OSCAR, Open Service Container Architecture". http://oscar-osgi.sourceforge.net

[11] H. Jormakka, J. Koivisto, T. Kyntaja, "Open service architecture for heterogeneous home environment", Proceedings of the 5th IEEE International Workshop on Networked Appliances. p. 7-11. October 2002.