# Handling User Profiles for the Secure and Convenient Configuration and Management of Mobile Terminals and Services

G. Bartolomeo[(1)], F. Berger[(2)], H.J. Eikerling[(2)], F. Martire[(1)], S. Salsano[(3)]
*(1) Radiolabs, Rome, Italy, (2) Siemens Business Services C-LAB, Paderborn, Germany,*
*(3) DIE, Univ. of Rome "Tor Vergata"*

## Abstract

*Profiled information is becoming of fundamental importance to configure and manage Mobile Terminal and Services for the personalized use according to the needs and preferences of a mobile user. This paper describes an architectural approach for the customization of terminals and services accessed via these terminals, focusing on the secure handling and applying of user profiles taking into consideration the distribution of the profile data.*

## 1. Introduction

Mobile users nowadays are surrounded by a plethora of different types of networks, services, and terminals. Though this is obviously favorable for the user, the complexity needs to be managed in a rather obtrusive and secure way. Architectural approaches for doing this are being studied by the FP6 projects Ubisec [3] and Simplicity [4]. Within Ubisec the security and configuration aspects of nomadic service discovery and access are examined for an environment being composed of separate elementary networks (EN) interlinked with each other through a global network (GN). The GN might be partly absent, thus centralized services like for instance trusted third parties for providing basic security services like for instance issuing and validating certificates might at least temporarily not be accessible. In contrast to this, Simplicity deals with the configuration management of mobile terminals and services based on a distributed brokerage framework in which (virtually) one agent is associated with the network / service being accessed by the user and another one is associated with the terminal and interacts with a special device (Simplicity Device) which enables the customization process. Since the entire configuration process in both approaches is driven by the user, the user profile is an important part in this picture. However, user profile and other user related data linked to the identity of a specific user must be secured and privacy issues should be taken into account.

In this work we describe an architectural approach which supports the configuration process on the application level especially taking into account the mobility of the users and the complementary use of different types and instances of terminals. This approach will be referred to a as ***Customization Framework (CF)*** and represents the combination and generalization of concepts and solutions developed in the context of the Ubisec and Simplicity projects. The CF concept is targeted to include several aspects like authentication, service discovery, management of services, user mobility, presence, location awareness and so on. A key element of the Customization Framework is the management of profiled user information, and this aspect will be specifically dealt with in this paper.
The paper is structured as follows: subsequently we will give a system overview containing target use cases and explaining the system principles, the proposed architecture and the definition of profiles. Afterwards, we will focus on the security and privacy aspects of profile handling.

## 2. System Overview

### 2.1 Targeted Application Scenarios

The envisaged Customization Framework is intended for the largest possible set of mobile and context-aware applications (similar to those described in [6]). Just to mention a specific set, we can have: mobile worker scenarios, where the CF assists the user in configuring its devices and applications for accessing different networks while being on the move, nomadic leisure scenarios, where a nomadic user requests to access media files (either consisting of personal records stored on his domestic digital recording device or being provided through a third party content provider) from his mobile terminal and so on. Depending on the user's context, his preferences (e.g., QoS), the capabilities of his access devices (in terms of supported wireless transport protocol, display capabilities etc.) and the communication mechanisms available on the site, the

local applications should be configured automatically. A comprehensive list of use cases is described in [11].

## 2.2 System Principles

In this sub section we provide a very high level view of the system principles, as shown in Fig. 1 which relates the major entities being involved in the customization process to each other.

**Access Device**: we refer to the user terminal as an access device which hosts the user applications. The applications are subject to configuration, as well as the networking interfaces of the terminal and also the access device itself.

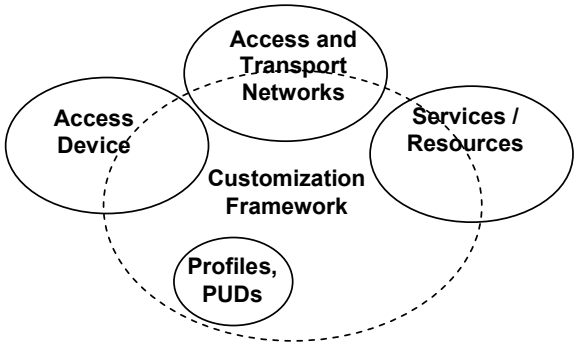


Fig. 1: Scope of the customization framework

**Service / Resource Host**: hosts the services to be called by the access device over a network. Similar to the application accessing the service, the service itself is also subject to a configuration.

**Access and Transport Network**: in principle, also Access and Transport networks could need to be configured according to the needs of the users.

**Personal User Device (PUD)**[1]: is the component that represents the user in the system. The PUD is something like a sophisticated key that a user can carry with him; it stores user profiles including preferences, user credentials for authentication and authorization and any other personalization information needed for the customization of services and applications. Ideally, the PUD should offer both storage and processing capabilities and should be capable of crypto features. The PUD can be implemented using different technologies like smart cards, flash memories, smart phones, SIM cards. However, due to memory limitations of these types of devices, it keeps only most sensitive information of the user profile as long as the large amount of the profile data are distributed in a network repository. This implementation leads to security problems and a general user's reluctance to give personal data to a third party.

[1] In the Simplicity Project the PUD is called Simplicity Device (SD).

**Customization Framework**: handles the configuration (i.e., profile data) information that is used to configure the application on the access device as well as for the services on the service host. The information can be either accessed through a network or, alternatively, through local profile manager. This is due to the fact that for certain profile data the retrieval of the profile over the network might be inefficient or might be indicated in order to support the access to certain profile information (user / identity related information) in off-line situations (which essentially is an aspect of nomadicity).

## 2.3 System Architecture

The main architectural entities that constitute the Customization Framework are shown in Fig. 2: Terminal Brokers (TB), Network Brokers (NB), Personal User Devices (PUD). The Brokers are the software entities that implement the Customization Framework in the Access Device and in the Network. Their functionalities may include: collection and combination of information from other entities, provision of the resulting information (context, preferences, policies) to other entities, management of network-related functionalities (e.g. advanced mobility management), provisioning of services and service discovery. The brokers communicate each other through a (XML based) inter-broker protocol.

**Terminal Broker**

The Terminal Broker is the entity that manages the interaction between the information stored in the PUD and the terminal in which the PUD is plugged in. On the other side it enables to perform service discovery and usage, adaptation of services to terminal capabilities. To realize these features it may request global information (e.g. the list of available services which the user has subscribed to) that TB presents to the user via a graphical interface.

The TB is logically decomposed into a set of components called subsystems which cooperate in order to perform the tasks described above. The most important subsystem for the customization feature is the Profile Access Manager (TB-PAM) that gives access to the required user profile information needed. This information may reside on a device being attached to or being part of the access device in order to perform service customization.

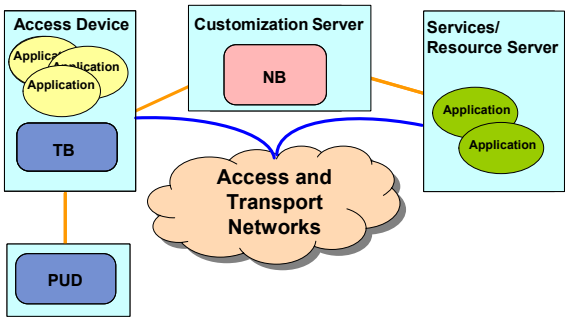


Fig. 2: Architectural entities of CF

**Network Broker**

The Network Broker System is the entity at network side that allows for a user friendly, easy to use and context-aware service provisioning. It does so by coordinating the selection, distribution of and subscription for information originating from different and potentially distributed resources like for instance terminal devices, location tracking systems, or network monitoring functions.

Note that the Network Broker is not a single central entity in a given network, it is implemented in a distributed and scalable way, but these distribution aspects are out of the scope of this document. The design of the NB is based on subsystems that are quite a replication of the subsystems inside the TB. The rationale for this is that a sub-system will request a given service to the *local* sub-system which will contact the corresponding *remote* sub-system if it is not able to provide the service locally. In fact the Network Broker hosts a counterpart of the TB-PAM, named NB-PAM and is able to perform operations similar to those performed by the TB-PAM on the user profile.

**Personal User Device Profile Manager (PUD)**

As pointed out in the previous section, the PUD is something like a key which the end user can carry on with him allowing him accesses a number of personalized services customized using profile information kept inside it.

## 2.4 Profile access management

The profile access management is composed of two cooperating managers, one assigned to the Terminal Broker (TB-PAM) and the other to the Network Broker (NB-PAM). They deal with request issued by the application residing on the terminal, e.g. for retrieving or updating / editing profile information. For handling these requests, each PAM has a layered internal architecture. At the top level there is an element named Profile Request Handler (PRH). The PRH is the major component for handling security aspects in the system with respect to privacy and other security aspects of personalized data as will be described later on. The two PRHs on the Terminal Broker and on the Network Broker can communicate with each; furthermore, they communicate with their respective underlying elements, called Data Access Managers (DAMs). These latter in turn are able to communicate with the last layer element of the PAM architecture, the controllers. There may be many controllers (e.g., a PUD controller for accessing the data on the PUD or another one for accessing the data residing on a network repository) as each controller is associated with a given data storage repository. This design pattern has been chosen because it allows designing the DAM independently of the interfaces exposed by the each data storage repository. Inside the PAM, the TB-PRH is the core component responsible for processing requests coming from third party services or applications which might involve the retrieval or modification of profile information. Each request must be supplied with the needed credentials of the requester. The TB-PRH first checks these credentials, then checks the access rights associated to the requester and, depending on the result of the check, permits or denies accesses to the profile element.

In other words, the TB-PRH logic does answer the following questions: "*Who wants to access what? What does it wants to do with that?*" The details of handling privileges during the processing request will be described in the subsequent paragraph.

## 2.5 Handling of profile data distribution

The Data Access Manager (DAM), located inside the TB-PAM, is the entity in charge of supporting and managing the distribution of profile data over different storage resources complementing the user related data stored on the PUD. Inside the DAM, the user profile is arranged in a way similar to how files are arranged into a modern hierarchical file system. Each part of the user profile is can be an XML file, a text file or a binary file stored on the physical PUD or on the network repository; furthermore, these file may be encrypted if needed. The DAM offers an abstraction of the user profile on which it is possible to operate without knowing where and how data are physically stored. This also simplifies the definition of a request format which has to tag the methods for reading and writing the profile data in an abstract and rather generic way. From an external (application) point of view, there is just one user profile stored in XML format accessible through the DAM.

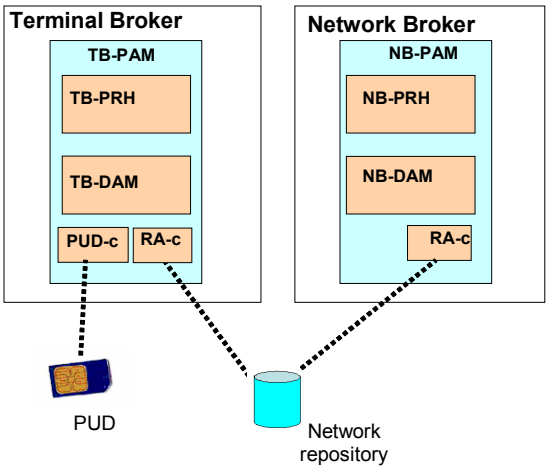


Fig. 3: Broker architecture for profile management

The DAM encapsulates an XQuery engine and exposes an interface based on the XQuery/XPath language specifications [10]. Inside the TB-PAM, the TB-PRH subsystem exploits the aforementioned interfaces for reading and/or modifying the user profile according to the requests it receives form other subsystems/applications.

The following is an example (dealing with the negotiation of network characteristics) in which the DAM is asked about the user preferences in service QoS:

```
declare namespace s='http://www.ist-
simplicity.org/SUP';
for $b in ('Quality','Speed','Cost')
for $a in //s:*
where name($a)=$b
return string((name($a),'=',$a,'\n'));
```

When this query is executed, the DAM returns a list of preferences formatted as 'properties' (i.e. a set of couple of strings using the syntax 'key'='value'):

```
Quality=0.7
Speed=1.0
Cost=0.5
```

As mentioned before there might be situations in which it is impossible or impractical to contact the TB-PAM; therefore there is a certain necessity to have access to the user profile data from the Network Broker bypassing the Terminal Broker. The NB-PAM just allows this, working as a sort of proxy for the TB-PAM. Unlike the TB-DAM, however, in order to guarantee a certain degree of security to the user, the NB-DAM has an access to the physical profile data location limited to those pieces of data for which the NB-PRH has received a delegation and moreover it can't contact the PUD directly; anyway data stored into the PUD may be accessed through a specific request at PRH level from the NB-PRH to the TB-PRH. using an appropriate access control language. The delegation mechanism applies to: a given part of the user profile which can be given on element, sub-tree or whole profile set level; a given type of operation which is possible to perform on that part of the user profile (e.g. read/modify/delete/…).

## 2.6 Profile access management

In order to access profile information, the respective application or subsystem has to issue an according request. The request has to include the targeted repository type, the unique identification of the repository and further information for querying the profile data. This URIs alike meta-information is processed inside the DAM. As a prerequisite to applying this scheme, each resource (e.g., file) containing pieces of user profile has to be assigned a unique address.

Following, there are some examples of URIs used to address a given storage space:

```
PUD://default?start=0&stop=50&namespace=http:/
/www.ist-simplicity.org/SUP/&name=
UserPersonalProfile/Identity/FullName/text()
```

By using this URI, the DAM can address a sequence of bytes inside the PUD, starting from position 0 and ending at position 50; these bytes contains a plain text which represents the user's full name; in the UP, this data is located inside the element addressed by the following XPath expression which addresses a text element inside a given XML tag:

```
UserPersonalProfile/Identity/FullName/text()
```

Tags are referred to by giving a namespace http://www.ist-simplicity.org/SUP/. We use the entry "*default*" to indicate that on the PUD there is just one storage space available (in the form of a sequence of bytes). Anyway there may be PUDs which support a file system storage space. In this case the expression "*default*" is replaced with a suitable filename.

```
NR://filename?namespace=http://www.ist-
simplicity.org/SUP&name=UserSubscribedNetworks
```

In a similar way, this URI refers to a given file stored in the network repository storage space owned by the user. The content of the file is represented by an XML instance which forms part of the user profile and offers information about e.g. the user's subscribed networks.

## 2.7 User Profile Definition

Personalisation is an element of customisation and the User Profile is key to the personalisation process. The User Profile (UP) is further specified by means of the 3GPP Generic User Profile (GUP) [7] concept which does not specify the data content, but only the format, i.e. the data model and the schema (Data Description Method based on XML Schema).

Actually the UP contains five components, corresponding to User Personal Data, User Devices, Personal User Devices, Subscribed Networks and

Subscribed Services. The main idea is that the User Profile is a "*User Level*" representation of the user himself and his ambient "*information and communication*" context which might change dynamically due to movements. The figure below shows that the UP is divided into five components. The figure shows that there is another level of representation in addition to the "*User Level*" that contains the UP. The "*Universe*" Level contains the format descriptions (schemas) of all existing devices, access networks, services and personal user devices.

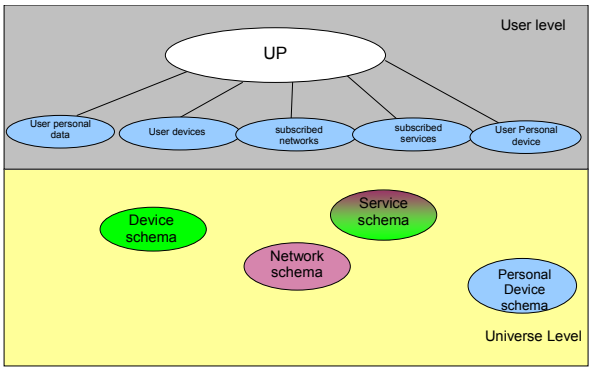


Fig. 4: User Profile – abstract view

In order to analyze an instance of the UP in the "*User Level*" for a given user, consider the *user devices* component (the same can be applied to the network, services and Personal device components). This component includes the representation of all owned devices, simply "copied" from the device representation existing in the "*Universe level*". For each device, some user configuration, preferences and personalization data is added.

In order to define the five components of the "User Level", the existing proposals and standards have been investigated and wherever possible integrated in the User Profile (e. g. Liberty Alliance Project Personal Profile (PP) **Error! Reference source not found.**, UAProf Schema provided by the WAP Forum [9]). Actual XML schemas and example instance files can be found in [12].

## 3. Security Issues

The appropriate handling of security issues with regard to ubiquitous environment is a rather contemporary concern [1], [2]. Security features inside the CF deal with encryption in order to ensure privacy, hashing of profile payload data in order to track unauthorized changes of profile information, and a mechanisms featuring tamper-resistance of the utilized storage spaces, authentication of users for accessing data and authorization for dealing with access privileges.

With respect to the architecture described above, the following security aspects have to be considered (numbering refers to Fig. 5)

Mutual authentication of the PAM (either TB-PAM or NB-PAM) and the application or service has to be done, preferably only once in the beginning (5).

The transmission of data between the PAM and the profile storages has to be protected against eaves-dropping (1).

The delegation of requests from the TB to the NB as well the access to certain parts of the profile data has to be authorized (2, 3).

Tracking of fraudulent and unauthorized changes of request meta-data and data payloads has to be enabled (4). According to Fig. 5 this is approached as follows:

Encryption is used for the data exchange between the storage devices and the DAM. The DAM keeps the encryption keys at a secret place, different from the location where the profile data is stored. Third-party applications can only access protected profile data if they contact the PRH. In the latter case, the applications have to provide credentials to proof that they are in fact allowed to access the requested profile data. Unprotected profile data can be accessed without providing credentials.

(2) XACML for exchanging delegation. This is a specific aspect of authorization and two complementary schemes for this can be supported. Firstly, a PMI (Privilege Management Infrastructure) is applied. To establish a PMI in the context of the CF, a (potentially remote) Attribute Authority and a (local) Privilege Verifier have to be employed. The Privilege Verifier communicates with the Attribute Authority to check for valid authorization. Secondly, authorization can be performed by providing other forms of credentials (PINs, passwords, pass phrases) which come along with the requests.

(3) ACSL for defining policies based on privileges for access control inside each PRH with respect to the requesting principal. Similar to (2), two authorization schemes can be supported.

(4) The use of hash values (e.g., SHA1) permits uniquely characterize the profile data, so that tampering of profile data can be observed. A hash value has to be recomputed by the PRH each time the profile data is changed. Hashing can be applied to different parts of the profile, i.e., at the element level, at the sub-tree level, or on the entire profile data level.

(5) The mutual authentication of the PRH (either TB-PAM or NB-PAM) and the application / service is performed by using an enhanced public key infrastructure (ePKI) solution which supports connected as well as disconnected modes of devices

(w.r.t. a trusted third party managing a certification authority). Since applications interact with profile storages on behalf of the user through the PAM, the user may have to authenticate with the application in the beginning. From the users point of view this has the advantage of having authentication only be done once, right in the beginning.

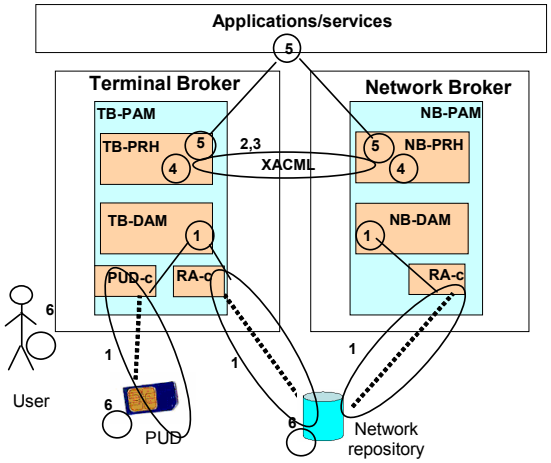


Fig. 5: Security issues w.r.t. to Customization Framework

To control access to profile data, the PRH checks credentials that are provided together with incoming requests based on the policies defined in the ACSL. Just to briefly explain, an access control policy is a set of rules of the form (`cred, profileId, accType, ctrlKind, scope`). Herein `cred` refers to a credential intended to proof the right of the requesting entity to access the desired profile data. The credential can, e.g., be a PIN uniquely representing a person, an application, or a group of people/applications.

`profileId` refers to the desired profile instance to access, `accType` refers to the type of access and can be one of *create*, *delete*, *modify*, *read*, *write*, etc., `ctrlKind` can be one of *grant* or *deny*, i.e., there are rules to explicitly grant or deny access, `scope` refers to the scope of the rule, e.g., a single element of the profile or an element together with its direct sub-elements or its complete sub-tree.

The policies are used to evaluate whether an access request is granted or denied. Several access control languages such as XACML and XACL have already been defined and used in different domains. Though the concepts presented here are fairly independent of this, we use XACML for the purpose.

## 4. Conclusions

Next-generation wireless system (3G and beyond) will likely increase the already overwhelming number of services, access devices, network interfaces that the user will have to manage. A Customization Framework able to support the users in the configuration and use of services and devices is envisaged in this paper. In particular, the issues related to the management of user profiles which is an important aspect of the Customization Framework have been discussed. For this an architectural framework has been devised, integrating the results of the two IST projects Simplicity and UBISEC. Utilizing this framework it is possible to access the profile data stored in "Personal User Devices" as well as distributed in network repositories in a seamless though "controlled" way, i.e. addressing the security concerns of the users.

We believe that there is a strong need to standardize the mechanisms related to the definition and handling of user profiles and this work represents a contribution in this area.

## References


[1] Workshop on Security in Ubiquitous Computing, 2002: http://www.teco.edu/~philip/ubicomp2002ws

[2] Second Workshop on Security in Ubiquitous Computing, 2003: http://www.vs.inf.ethz.ch/events/ubicomp2003sec/

[3] UBISEC website: http://www.ubisec.org

[4] IST Simplicity Project: http://www.ist-simplicity.org

[5] B. Rao, L. Minakakis, Evolution of mobile location-based services, Communications of the ACM, pp. 61-65, Vol. 46, Dec. 2003

[6] Y.-B. Lin and I. Chlamtac, "Wireless and Mobile Network Architecture", John Wiley & Sons, Inc., 2001.

[7] 3rd Generation Partnership Project. Data Description Method (DDM) - 3GPP Generic User Profile (GUP). Technical specification of Technical Specification Group Terminals, Version 6.1.0. 2004.

[8] Project Liberty Alliance: http://projectliberty.org

[9] Wireless Application Forum: http://www.wapforum.org/

[10] XQuery 1.0: An XML Query Language, W3C Working Draft 04 April 2005: http://www.w3.org/TR/2005/WD-xquery-20050404/

[11] IST Simplicity Project, D2101 - Use cases, requirements and business models

[12] XML UP Schemas: http://www.ist-simplicity.org