



ITEA
INFORMATION TECHNOLOGY
FOR EUROPEAN ADVANCEMENT

ITEA Project 04012

LOMS – Local Mobile Services

Task 6.6: Common end-user service components

Editor:

Stefan Goeman

Devoteam Telecom & Media

Authors:

Jürgen Tacke (Orga Systems)

Florian Klompmaker (C-Lab)

Security:	public
Version:	1.0
Date:	12/08/2008
Number of pages:	10

Alterations

Version	Date	Person	Comment
0.1		Stefan Goeman & Jürgen Tacken	Initial Draft Version
0.2		Stefan Goeman, Jürgen Tacken, Florian Klompmaker	
1.0		Stefan Goeman, Jürgen Tacken, Florian Klompmaker	Final Version

Abstract

Common end-user services are “horizontally” applicable services or components that can be used in many other end-user services. Common end-user services are not enabling services but rather building blocks interacting with the system end-user. In this document we describe two such common end-user services that have been developed in the LOMS project, i.e. the Threshold management service, which can be used by the end-user to define service specific thresholds, and the LOMS Instant Messaging Service, which can be integrated into other service and used by end-users to send messages to each other.

Conventions

Throughout this document, the keywords “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” are to be interpreted as described in [RFC2119].

Table of Contents

1	Executive Summary	4
2	Threshold Management Service	5
3	The LOMS Instant Messaging Service.....	9

1 Executive Summary

The goal of Task 6.6 on Common end-user service components was to define components that are “horizontally” applicable in many end-user services. These services are not enabling services exposed by an operator or the LOMS platform operator but are rather common building blocks interacting with the system user. Examples of such services might include shared calendars, personal digital purse, etc.

This document describes two such services that were developed during the LOMS project, i.e. the Threshold Management Service and the LOMS Instant Messaging Service. The Threshold Management Service implements a service that an end-user can use for different other end-user services. This service allows a user to define service specific thresholds for arbitrary end-user services. In case this threshold is reached, the user will be notified by a notification system using a preferred channel.

The LOMS Instant Messaging Service (LOMSIMS) is a Web Service that provides registered users with the ability to exchange short messages. Other end-user services can make use of this LOMS Instant Messaging Service. With the LOMS Instant Messaging service integrated into another service, end-user can send messages to each other and in case users are not online the messages will be stored by the service and delivered to the recipient when he comes back online.

2 Threshold Management Service

The Threshold Management Service is a component horizontally applicable to many end-user services. Threshold Management refers to the feature of notifying a user about reaching a threshold for a specific service. A user will be enabled to define service specific thresholds for arbitrary end user services. In case this threshold is reached, e.g., when the user has spent 3.50 euro on a latest news service, the user will be notified by a Notification System using a preferred channel (i.e., either SMS, email or a direct notification through the end user service itself).

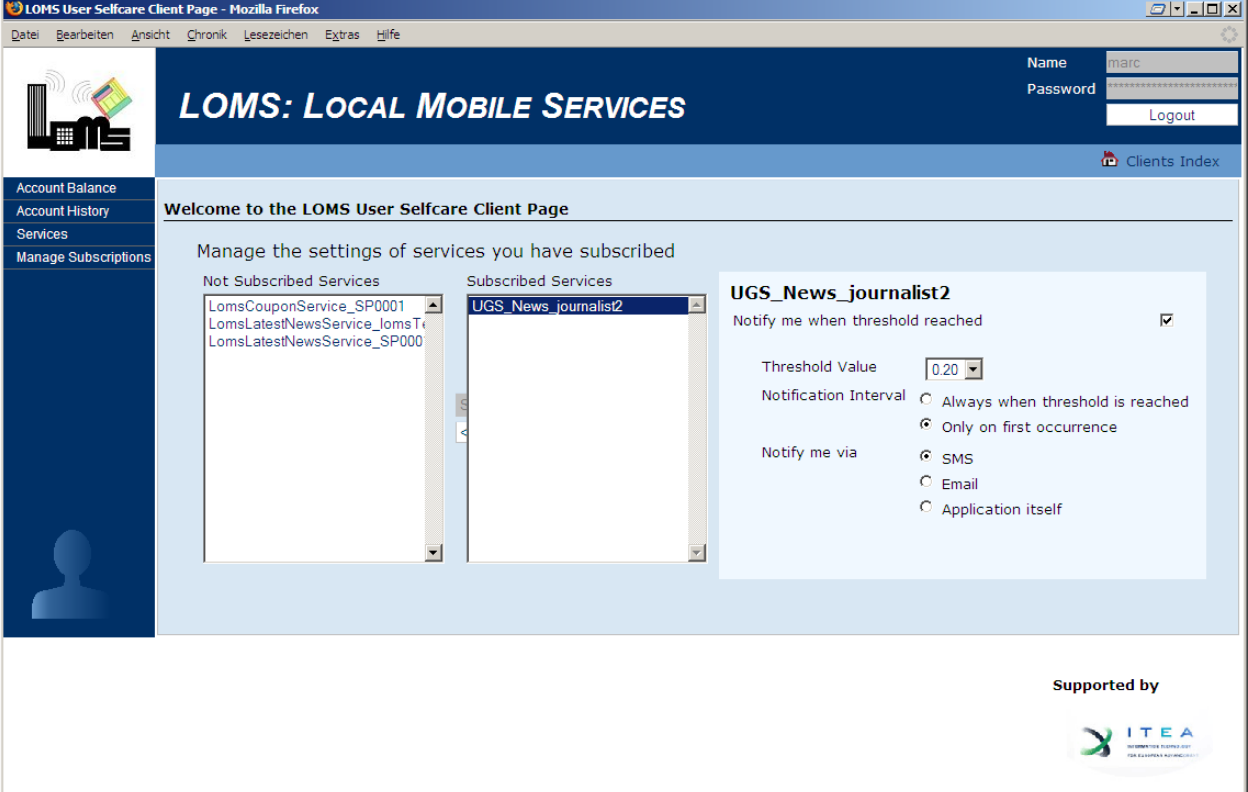


Figure 1: Integration of Threshold Management Service in LOMS Demonstrator.

Figure 1 shows the integration of the Threshold Management Service in the LOMS Demonstrator. Within the 'Manage Subscriptions' page of the LOMS User Selfcare Client, the user is able to activate/deactivate threshold notifications and change threshold settings for each subscribed service. The user is able to specify a threshold value, a notification interval and the notification channel.

The Threshold Management Service is tightly connected with the Service Management and the Charging Service. Figure 2 shows the overall architecture.

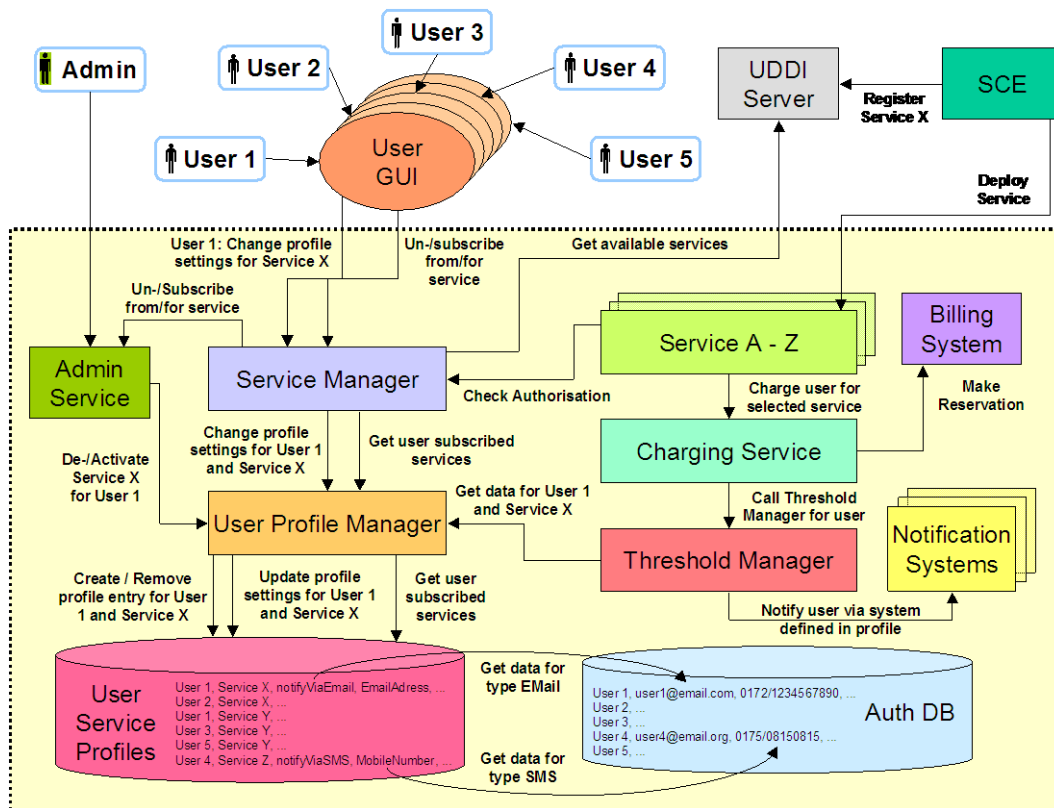


Figure 2: Architecture for Service and Threshold Management.

A central component is the User Service Profiles Database. In this database, the service-specific settings of the subscribed services for the end users are stored. A profile entry for a particular user and a particular end user service will be created at the time of subscription of the user for the corresponding service. The entry will be removed when the user unsubscribes from the service.

While being subscribed to a service, the user always has the possibility to update the service-specific profile entry. A User Service Profile Entry consists of the following fields:

1. **userID:** Unique identifier of the user
2. **servicelD:** Unique identifier of the service
3. **isActive:** Flag indicating whether the threshold notification is activated or not
4. **threshold:** Value referring to the threshold for this service
5. **notificationType:** Type of user notification (SMS, Email, directReply, etc.)
6. **notificationAddress:** Target for notification type, like e.g. Mobile Number, Email Address, etc. The Mobile Number or Email Address is retrieved from another database, the Authentication Database (Auth DB) containing user specific information.
7. **alreadyNotified:** Flag indicating whether the user has already been notified or not
8. **notificationPeriod:** A flag indicating when the user is to be notified – *once* after reaching the threshold, *once daily* after reaching the

threshold, or *periodically, i.e., each time when the threshold value has been reached again*

9. **accumulateAmounts**: A flag that indicates whether the user wants the charged amounts for the particular service to be accumulated for the threshold notification or not. If “true”, all previously charged amounts for using the service are summed up in **accuAmount** (see below) and the threshold is continuously checked against this value. When the threshold is reached, the user is notified according to the desired **notificationPeriod** parameter (i.e., once when the threshold has been reached for the first time or periodically). If **accumulateAmounts** is “false”, the threshold is checked only against the ongoing usage session (i.e., already finished sessions are not considered).

Note that for the integration into the LOMS demonstrator as shown in Figure 1 this entry is set to “true” by default, such that the dialogue shown does not offer end users to set this entry.

10. **accuAmount**: When the **accumulateAmounts** entry is true, this entry is used to sum up the charged amounts across charging sessions.
11. **lastNotificationDate**: The date when the last threshold notification was sent
12. **subscriptionDate**: Date when the user subscribed for the service

The Threshold Management Service provides the following methods via its Web Service interface:

1. **boolean checkServiceThreshold (authInfo, userID, serviceID, currentAmount)**

Check whether the user activated the Threshold Manager for the service identified by **serviceID** and check the threshold for notification necessity accordingly.

Parameters:

- **authInfo**: Authorization Information of the requesting application
- **userID**: Unique identifier for the user
- **serviceID**: Unique identifier for the service
- **currentAmount**: Amount to check against the threshold

Response:

True in case of success, false otherwise.

2. **UserServiceProfileEntry readProfile(authInfo, userID, serviceID)**

Read the current profile of the user identified by **userID** for the service specified by **serviceID**.

Parameters:

- authInfo: Authorization Information of the requesting application
- userID: Unique identifier for the user
- serviceID: Unique identifier for the service

Response:

The User Service Profile Entry

3. boolean updateProfile(authInfo, userID, serviceID, entry)

Updates the profile of the user identified by userID for the service specified by serviceID.

Parameters:

- authInfo: Authorization Information of the requesting application
- userID: Unique identifier for the user
- serviceID: Unique identifier for the service
- entry The User Service Profile Entry

Response:

True in case of success, false otherwise.

3 The LOMS Instant Messaging Service

The LOMS Instant Messaging Service (LOMSIMS) is a Web Service that provides registered users with the ability to exchange short messages. It is intended to be used with the LOMS Thin-Client, LOMS Medium-Rich Client and the LOMS Rich-Client applications.

With these applications users can use the LOMSIMS Web Service to send messages to the service, which stores them at the server until they are requested by the intended receiver. Messages are sent with respect to a certain context (i.e. a location) and may contain textual content and binary attachments. Additionally, if users provide valid mobile phone numbers upon registration, the service generates SMSs to inform them about any messages that the service received for them.

The LOMSIMS Web Service provides methods for sending messages (SendMessage), retrieving a list of messages (GetMessages), retrieving the textual content of a single message (GetMessageContent), and retrieving the attachment of a message (GetMessageAttachment).

As already said, with respect to client side implementations, several small standalone client applications have been developed (mainly for testing and debugging purposes). In addition, the LOMSIMS Web Service has been integrated into the LOMS Thin-Client. The LOMS Thin-Client is a PHP web site (accessed with a normal web browser) that uses several Web Services including the LOMSIMS Web Service.

The following pictures show a few screenshots of how the LOMSIMS PHP web application looks like.



Figure 3

When the service has been subscribed to (not shown in any picture) you can proceed by clicking **Go!** (see Figure 3). This will invoke the LOMSIMS's *GetMessage* method and present the result in a new window (see Figure 4).

For more information, we would like to refer to the complete description of this service.



Figure 4

Figure 4 presents the results of the invocation of *GetMessage* for the current user, the current location, and a value of true for the *onlyNew* parameter, i.e. it only shows the new messages for the current user/location combination. If you want to see all messages for the current location that are stored at the server you can choose **Show all Messages**.