2010 Sixth International Conference on Wireless and Mobile Communications

Design and Implementation of a RESTful IMS API

Heidi-Maria Rissanen, Tomas Mecklin, and Miljenko Opsenica
Ericsson Research
NomadicLab, Jorvas, Finland
email: {firstname.lastname}@ericsson.com

***Abstract*—Web 2.0 has been happening on the fixed Internet side for several years already. This phenomenon has forced traditional Telco operators to consider new business models. One of the possible models is to utilize the IP Multimedia Subsystem (IMS) network architecture. Using the IMS, Telco operators have more control over the data instead of becoming just bit pipes. In addition, operators interest in providing open Application Programming Interfaces (APIs) exposing IMS and network assets has grown. In this paper, a RESTful IMS API implementation, is represented. The design and implementation of the API exposing IMS functionality is shown. In addition, the implementation of a REST application using the API is described.**

Keywords-IMS; REST; Telco API

I. INTRODUCTION

During the last few years, mobile operators have been challenged by the fixed Internet service providers, in particular by the Web 2.0 phenomenon and mashup applications combining data from different sources into one end-user application. Mashups can be built very fast utilizing already existing components. Web 2.0 related technologies and architectural styles such as AJAX (Asynchronous JavaScript and XML), REST (Representational State Transfer) [1] and SOAP (Simple Object Access Protocol) [2] have enabled service providers to provide this kind of components and APIs (Application Programming Interfaces).

Mashups are typically niche applications, for which the need might be short-lived. API providers themselves are not usually interested in building such applications but they provide the opportunity to other developers. As third party developers build applications using the APIs, service providers get more traffic to their web sites. In addition, service providers also wish to get innovative ideas from application developers.

In the Telco world, walled-garden strategy has been the dominating business model for a long time. New services have traditionally gone through a long standardization process. Service design has not taken into account end-user interests. One possibility for Telco operators is to expose the unique assets from their networks and move towards open-garden strategy [3]. IP Multimedia Subsystem (IMS), which is a mobile network architecture specified by 3GPP (3rd Generation Partnership Project), is one of the possible platforms enabling the fixed and mobile Web convergence and move towards open-garden strategy.

Today, REST is the most popular architectural style for mashup APIs with over 70 % [4]. Recently, open APIs have gained a lot of attention also from the mobile operators. Operators have run projects to get into the open API business. Ribbit [5] provides APIs for messaging, calling and user management. Deutsche Telekom has a developer program called the Developer Garden [6]. Modeling Telco resources to RESTful resources has been studied in [7]. In particular, making IMS RESTful, has been studied in [8].

Recently, Telco standardization has began work on open APIs, too. To be able to use the same applications across different operators, common APIs are needed. To define open API specifications, GSMA One API initiative [9] was started. The first release of the specification consists of APIs for messaging, payments and location. APIs are available both in REST and SOAP.

In this paper, a project, which provides IMS functionality through RESTful interfaces, is represented. In Section II, IMS network architecture and functionality is described. In Section III, the REST architectural style and the design of the IMS REST API is discussed. The implementation of the RESTful API and a sample application is described in Section IV. Finally, in Section V, the paper is concluded.

II. INTRODUCTION TO IMS

IMS is a network architecture standardized by 3GPP. IMS has been coupled with LTE (Long Term Evolution) networks, which will be followers of the third generation (3G) mobile networks. Thus, IMS networks are an essential part of the fixed and mobile Internet convergence.

IMS uses several IP protocols defined by the Internet Engineering Task Force (IETF) and ITU-T. The Session Initiation Protocol (SIP) [10] is used as the session control protocol, to establish and control multimedia sessions. 3GPP does not define standardized nodes, but functionality. In practise, one node implements one function. In the following, the IMS functions used in the IMS testbed network of this project are described.

- Home Subscriber Server (HSS) stores user-related information. It has information about the services that the IMS user has subscribed to. The HSS is always located in the home network of the user.
- Call Session Control Functions (CSCFs) perform different type of SIP proxy functions. A Serving CSCF (S-

978-0-7695-4182-2/10 $26.00 © 2010 IEEE
DOI 10.1109/ICWMC.2010.30

CSCF) is a stateful SIP server providing session control functionality. In addition, it acts as a SIP registrar. It keeps a binding between the user location and the Public User Identity (PUI) of the user. The S-CSCF is on the signaling path of the SIP messages and it inspects all messages. If needed, it will forward SIP messages to an Application Server (AS).

- SIP AS is a logical element, which hosts and executes end-user services. An AS can be located either in the home network or in a third party network. In the latter case, there needs to be a service agreement between the two parties.
  To be able to use the IMS services, a user needs to be provisioned to the network. Provisioning creates a default profile for the user into the IMS network. The profile defines the IMS services that the user is allowed to use.

A. Exposed IMS Functionality

In the scope of this project, limited IMS functionality is exposed to third party developers. In the following, the exposed functionality is described.

1) Registration: To be able to use the IMS services, the user has to be registered to the IMS network. This is done by sending a SIP REGISTER request to the IMS network.

2) Instant Messaging: Instant messaging in IMS is possible in two different modes. Stand-alone messages can be sent using SIP MESSAGE requests. This is called page messaging. If instant messaging happens as part of an existing SIP session, the MSRP (Message Session Relay Protocol) [11] is used. This is called session-mode messaging. MSRP runs on the media plane, which means that MSRP messages do not traverse SIP proxies in the IMS network. Thus, the MSRP is recommend to be used if there is a lot of content to send since the IMS nodes might have limitations regarding the size of the message.

3) IMS Presence Service: The IMS presence service is defined by 3GPP [12]. It is based on the SIP and SIMPLE specifications defined by IETF and OMA (Open Mobile Alliance). Using the IMS presence service a IMS presentity, for example a person or device, can publish information to the presence server in the IMS network. Presence information of an IMS user is represented with a PIDF (Presence Information Data Format) [13] document.

User of the presence service can define presentity lists containing the publishers whose presence information the user wants to follow. Lists are created and modified by using the XCAP (The Extensible Markup Language Configuration Access Protocol) [14]. After subscribing to a presentity list, the user will get notifications every time there are changes in the presence information of any of the presentities.

To control the visibility of one's own presence information, presence authorization rules [15] are used. Presence rules define, which users are allowed or denied to subscribe to the presence information. Rules are created and modified using XCAP. SIP also has a specific watcher information event for changes regarding watchers [16]. A presentity can subscribe to the watcher information events and get notifications.

4) Provisioning: Provisioning creates a default profile for the user into the IMS network. The profile defines the IMS services the user is allowed to use.

III. RESTful Design

REST is a non-standardized, architectural style introduced in [1]. It is based on the principles of the Web and RESTful applications are built using Web standards such as HTTP and URIs. In REST everything is modelled as resources and each resource has a unique URI. Standard HTTP methods, GET, POST, PUT and DELETE, are used for invoking each resource. GET method fetches information from the resource without changing anything on the server, POST method is used for creating or updating resources. PUT method is also used for resource updates. DELETE removes the resource. For example, the following query gets a SIP chat session with ID 12345: *GET /chat/12345*. Truly RESTful services are completely stateless, thus they are easily scalable.

RESTful services can be categorized into Hi-REST and Lo-REST based on the level of the "RESTfulness". Hi-REST APIs utilize main HTTP standard methods to communicate with the resource. In addition, for example XML documents are used as the data interchange format, metadata is put into HTTP headers and URIs are used meaningfully. In Lo-REST APIs, only HTTP GET method is used.

A. Data Interchange Format

For REST APIs the data interchange format, that is the payload of the messages, is typically either XML or JSON (JavaScript Object Notation) [17]. JSON is a text-based interchange format, which is commonly used in Ajax applications. The advantage of JSON is that the data is suitable for browsers.

The API has support for both XML and JSON interchange format in most of the methods. However, as some of the request and response payload is based on XML standards, there is support for XML only in those methods.

B. IMS and SIP Resources

The SIP and IMS concepts used in the API are based on the SIP Servlet API v1.1, JSR (Java Specification Request) 289 [18], and the IMS Services API, JSR 281 [19]. Mapping SIP concepts into REST resources is not straightforward. For example, the SIP Servlet API defines a class called *SipApplicationSession*, which stores information about both HTTP and SIP sessions. Even though HTTP is a stateless protocol, SIP has to have some kind of state in most cases. To be able to map a SIP session used in the *SipApplicationSession* class into a REST resource, the state of the session needs

to be stored internally in the REST application. First we planned to use cookies to store the state but gave up with the idea. Cookies store application state on the client, thus they violate REST principles [1].

There is a problem regarding state for SIP registration as well. In Web sites not using REST, a HTTP session can be used to identify a valid user. Using IMS, the user needs to have a valid SIP registration to be able to use IMS services. Thus, the IMS registration status of the user needs to be stored in the application. This means that our API is not stateless and not fully compliant with the requirements of REST.

To identify the IMS user, each REST request contains IMS authentication information of the user in standard HTTP headers. The user of the REST service also has to be authenticated. In this project, the REST service and IMS credentials of the user are the same. To be able to authenticate REST service users, REST service credentials are stored in a database on the Web server.

The GSMA One API does not provide specifications for functionality that is exposed in this project, thus modeling resources is not based on any standard. According to REST principles, nouns are used to identify resources. In the following, the resource model and the payload format used in the API is described.

1) Registration: For registration, the only resource defined is *registration*. POST method creates a registration to the IMS network, and DELETE method unregisters. GET method returns the status of the last registration request for a particular user.

2) Instant messaging: As instant messages are supported in two modes, session-mode and page messaging, different resources for different type of messaging were defined.

The resource URI to fetch all page messages for a user is */im*. The resource URI to send a message using HTTP POST is */im/{user}*, where *user* is the one to whom you are sending the message *alice@imsinnovation.com*. New messages can be fetched from the IMS network using GET method. The following response format for instant messaging requests was defined:

- messages: received messages for this user
  - message: description of a message
    - from: URI of the sender
    - content: describes the content
      - content type: MIME content type of the message
      - plainTxtContent: content

A request to get new messages would be:

```
POST /im
Host: https://provisioning.imsinnovation.com
Accept: application/xml
```

A successful response would be:

```
HTTP Status: 200 OK
Payload:
<?xml version="1.0"
encoding="UTF-8" standalone="yes"?>
<imsRestResponse
xmlns="com:ericsson:imsinnovation:rest:instantmsg">
  <imMessages>
    <imMessage>
      <from>sip:bob@imsinnovation.com</from>
      <content>
        <Content-Type>text/plain</Content-Type>
        <plainTxtContent>hello!</plainTxtContent>
      </content>
        </imMessage>
    </imMessages>
</imsRestResponse>
```

For session-mode messaging using SIP sessions, the resource URI is */chat*. Each user can have several sessions and each session has a unique identifier, which is returned in the response when a session is created. If a particular session is wanted to be used, the resource path is */chat/{chatId}*.

The payload structure for a session-mode chat is the following:

- session: describes a SIP instant messaging session
  - identifier: a unique identifier
  - media type: type of the media used in the session
  - session description: textual description of the session
  - session state: state of the session, possible states are *proceeding,established,negotiating,terminating* and *terminated*

Messages for a chat are accessible using the following resource URI: */chat/{chatId}/messages*. Besides to the fields used in page messaging, some additional information is needed:

- message
  - transaction: describes a message transaction
    - transaction ID: identifier of the transaction
    - status code: SIP status code for the message status
  - receivers: to whom this message is sent to

3) Presence information: For publishing presence information, the standard PIDF XML document defined in [13] is used. The resource to represent the PIDF document is *presence information*. It is possible to publish, unpublish and fetch the current PIDF document from the presence server using POST, DELETE and GET methods respectively.

4) Presentity Lists: To subscribe to the presence information of other users publishing information, the user needs to create presentity lists using HTTP POST on the presence server. The resource URI to presence lists is */friendlists*. To represent presentity lists, we use the following payload format:

- presentity list: describes a presentity list
  - owner: SIP URI of the list owner
  - name: unique name of the list
  - presentity: describes a friend on this list

A presentity element on the list is modelled in the following way:

- presentity: describes a friend in the list
  - name: SIP URI of the peer
  - status of the subscription: describes the status of the subscription, for example, if the presentity to be watched has accepted the request

To receive notifications about presence information of the other users, a user creates a SIP subscription to the list. A subscription is created using POST method and resource URI to the subscription resource: */friendlists/{friendlist}/subscription*. New notifications can be fetched from the server using GET method and URI */friendlists/{friendlist}/subscription/notifications*.

Presence information of another presentity will be available only if the other user has consented to be watched. The presence information of each presentity in PIDF format can be accessed from */friendlists/{friendlist}/peers/{peerusername}/pres*.

5) Presence Authorization Rules: Resources related to watcher information document and presence authorization rules are defined in the following way. A presentity will get notifications for new watchers and other changes on the list by subscribing to the watcherinfo resource, which is based on the SIP watcher information event defined in [16]. The watcher information subscription resource is defined as */winfo/subscription*, and notifications can be fetched from */winfo/subscription/notifications*. A response format to get notifications is the following:

- watcher-list: contains a list of changed watchers
  - watcher: describes a watcher on the watcherlist
    - display name: textual representation of the name of the watcher
    - identifier: a unique identifier for the watcher
    - status: status of the subscription made by the watcher
    - event: event that caused the transition to this status

Presence rules resource represents the presence authorization rules document on the IMS presence server based on the standard [15] . The URI to the presence rules document is at */winfo/rules*. When there is a notification about a new watcher, the presentity can decide to either allow or decline the watcher and the new rule will be added to the presence rules document.

IV. REST Implementation

Java was used as the programming language in this project, thus we chose to use Jersey [20] for implementing the RESTful API. Jersey is a reference implementation of the JSR 311, which is a Java API for RESTful services [21]. Jersey supports the annotations defined in the JSR and eases the RESTful service development.

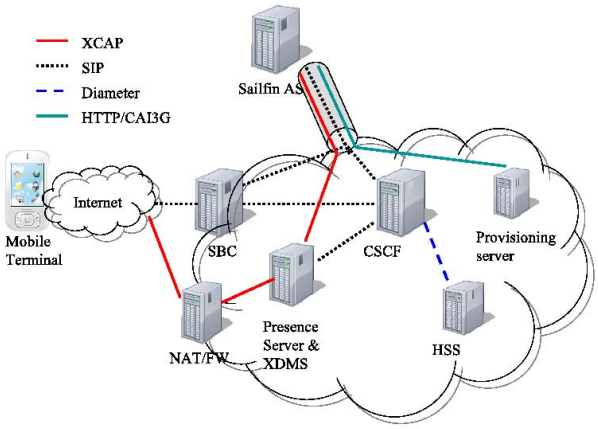

Figure 1. Ericsson Labs IMS Network

A. Testbed Network

The REST application is deployed to the Ericsson Labs IMS testbed network [22]. The IMS network is operator independent. The architecture of the testbed network is shown in Figure 1. The IMS network consists of a S-CSCF, HSS, SIP AS and a Presence Server. The Presence Server contains also a XML Document Management Server (XDMS). There is also a Session Border Controller (SBC) on the edge of the IMS network controlling the signaling and MSRP connections to and from the IMS terminals.

In the IMS testbed network there are two Sailfin ASs [23]. Sailfin AS is a Java EE AS with a SIP Servlets technology extension. On one AS we run a provisioning application and the REST implementation. On the other server we run REST sample applications using the IMS REST API. Each AS is connected to the IMS network with a VPN (Virtual Private Network) tunnel.

The REST service is deployed to one of the Sailfin ASs. Also, Jersey was installed on the AS. To deploy an RESTful IMS application to the AS is similar to normal web application deployment.

To provision new users to the IMS network, we use the Ericsson Multi Activation provisioning system [24] in the IMS testbed network. The SOAP-based provisioning interface is called the Customer Administration Interface Third Generation (CAI3G). To provision new users or to update existing user profiles, a provisioning application is running on the AS.

B. Asynchronous Communication

SIP is an asynchronous protocol, which means that the server will notify the client, that is the AS, if there is an active subscription to particular events. However, as HTTP is used between the REST client and AS, the REST client needs to pull for new notifications from the AS. An example of this is shown in Figure 2. The user creates a presentity list

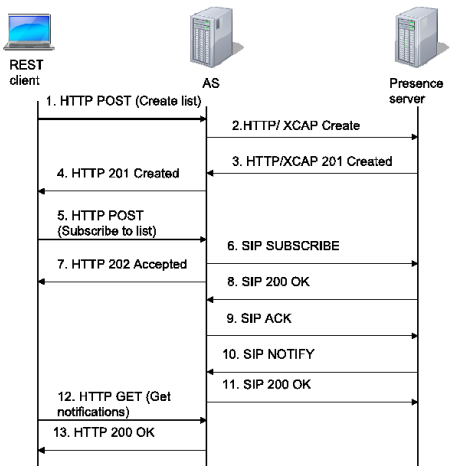

Figure 2. Synchronous sequence diagram

to the presence server and starts a subscription. Even though the AS will be notified about changes on the list, the REST client will not get notifications before it pulls the server. Thus, to know if there are notifications for the subscription, the user has to send a GET request to the AS.

It is possible to overcome this limitation if the Web server has support for Comet, that is HTTP server push. Comet utilizes persistent or long-lasting HTTP connections and offers real-time interaction. We made the REST API asynchronous using Atmosphere [25], which is a framework that makes it easier to build application using both Comet and REST. Atmosphere is built on top of Jersey and Grizzly Comet [26]. Using the asynchronous API for example for registration, the AS will not send a response to the client before it has received a SIP response from the IMS network.

When Comet is not needed anymore, the connection needs to be closed down and resources released using a method, which is part of the API. A sequence using Comet and asynchronous communication is shown in Figure 3. In this case, the user will get the Comet event notifications whenever there are changes regarding the subscription.

C. RESTful samples

To demonstrate the usage of the REST API, a JavaScript library was implemented on top of the REST API. It supports both asynchronous and synchronous variants of the REST API. As the sample applications are running on a different AS than the REST implementation, the issue with cross-site scripting had to be solved. flXHR [27] Javascript lib was used for that.

A synchronous instant messaging sample JavaScript application is shown in Figure 4. New messages need to be pulled from the server using the ”Get messages” button.

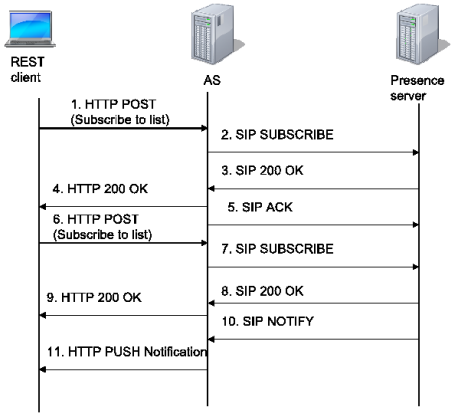

Figure 3. Asynchronous sequence diagram

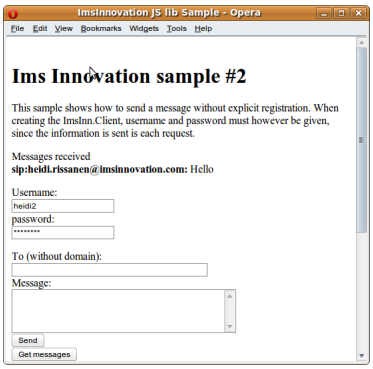

Figure 4. Instant messaging sample application

V. Conclusion and Future Work

The success of the Web 2.0 phenomenon has made the Telco operators to consider changing their business model. In particular, Telco operators have to provide open APIs and expose the unique assets from their networks. Providing open APIs and exposing mobile network assets might be one possibility for the mobile operators to change their business model.

On the Web 2.0 side, REST has been the dominating type of open APIs. Still today, there are not that many open APIs available to use the Telco assets in Web 2.0 mashups. This kind of APIs would be needed to converge these two worlds. In addition, Telco operators need to investigate what kind of assets to expose and how. APIs provided by operators need to be uniform, so that as little integration work as possible is needed from the application developers connecting to several operators' systems.

In this paper, a proof-of-concept project providing RESTful IMS API was represented. The design and resource structure of the provided API was shown. Design is based on REST principles and the existing IMS and SIP specifications. However, as SIP registration and SIP sessions require

state, the API is not totally stateless. The implementation of the API is based on existing Web technologies, such as the Sailfin AS and different Java-based APIs. To overcome the limitation of asynchronous SIP notifications, Comet is used. The RESTful API hides most of the SIP specific details, and using the provided JavaScript library it is easy to integrate the API with other Web 2.0 components.

In the future, there is still work to be done on regarding real-time media. Also, the RESTful API may be revisited as the specifications by GSMA are updated for example with presence service.

REFERENCES

[1] R. Fielding, "REST: architectural styles and the design of network-based software architectures," Doctoral dissertation, University of California, Irvine, 2000, Last accessed: 09.06.2010. [Online]. Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[2] N. Mitra and Y. Lafon, "Soap version 1.2 part 0: Primer (second edition)," World Wide Web Consortium Recommendation, April 2007, Last accessed: 09.06.2010. [Online]. Available: http://www.w3.org/TR/soap12-part0/

[3] Y. Raivio, S. Luukkainen, and A. Juntunen, "Open Telco: A New Business Potential," in *Proceedings of the 5th ACM Mobility Conference 2009*. ACM, Sep. 2009.

[4] Programmable Web. Last accessed: 09.06.2010. [Online]. Available: http://www.programmableweb.com

[5] Ribbit Corporation. Last accessed: 09.06.2010. [Online]. Available: http://www.ribbit.com

[6] Developer Garden. Last accessed: 09.06.2010. [Online]. Available: http://www.developergarden.com

[7] S. Mäkeläinen and T. Alakoski, "Fixed-Mobile Hybrid Mashups: Applying the REST Principles to Mobile-Specific Resources," in *WISE Workshops*, 2008, pp. 172–182.

[8] D. Lozano, L. A. Galindo, and L. Garcia, "WIMS 2.0: Converging IMS and Web 2.0. Designing REST APIs for the Exposure of Session-Based IMS Capabilities," *Next Generation Mobile Applications, Services and Technologies, International Conference on*, vol. 0, pp. 18–24, 2008.

[9] GSMA (GSM Association) One API. Last accessed: 09.06.2010. [Online]. Available: https://gsma.securespsite.com/access/default.aspx

[10] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," RFC 3261 (Proposed Standard), Jun. 2002, Last accessed: 09.06.2010. [Online]. Available: http://www.ietf.org/rfc/rfc3261.txt

[11] B. Campbell, R. Mahy, and C. Jennings, "The Message Session Relay Protocol (MSRP)," RFC 4975 (Proposed Standard), Sep. 2007, Last accessed: 09.06.2010. [Online]. Available: http://www.ietf.org/rfc/rfc4975.txt

[12] 3GPP, "Presence service; Architecture and functional description; Stage 2," 3rd Generation Partnership Project (3GPP), TR 23.141, Last accessed: 09.06.2010. [Online]. Available: http://www.3gpp.org/ftp/Specs/archive/23_series/23.141/

[13] H. Sugano, S. Fujimoto, G. Klyne, A. Bateman, W. Carr, and J. Peterson, "Presence Information Data Format (PIDF)," RFC 3863 (Proposed Standard), Aug. 2004, Last accessed: 09.06.2010. [Online]. Available: http://www.ietf.org/rfc/rfc3863.txt

[14] J. Rosenberg, "The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)," RFC 4825 (Proposed Standard), May 2007, Last accessed: 09.06.2010. [Online]. Available: http://www.ietf.org/rfc/rfc4825.txt

[15] ——, "Presence Authorization Rules," RFC 5025 (Proposed Standard), Internet Engineering Task Force, Dec. 2007, Last accessed: 09.06.2010. [Online]. Available: http://www.ietf.org/rfc/rfc5025.txt

[16] ——, "A Watcher Information Event Template-Package for the Session Initiation Protocol (SIP)," RFC 3857 (Proposed Standard), Internet Engineering Task Force, Aug. 2004, Last accessed: 09.06.2010. [Online]. Available: http://www.ietf.org/rfc/rfc3857.txt

[17] D. Crockford, "The application/json Media Type for JavaScript Object Notation (JSON)," RFC 4627 (Informational), Internet Engineering Task Force, Jul. 2006, Last accessed: 09.06.2010. [Online]. Available: http://www.ietf.org/rfc/rfc4627.txt

[18] "Sip Servlet v1.1," Java Community Process, 2008, Last accessed: 09.06.2010. [Online]. Available: http://jcp.org/en/jsr/detail?id=289

[19] "IMS Services API," Java Community Process, 2008, Last accessed: 09.06.2010. [Online]. Available: http://jcp.org/en/jsr/detail?id=281

[20] Jersey Project. Last accessed: 09.06.2010. [Online]. Available: https://jersey.dev.java.net

[21] "JAX-RS: The Java API for RESTful Web Services," Java Community Process, 2008, Last accessed: 09.06.2010. [Online]. Available: http://jcp.org/en/jsr/detail?id=311

[22] The ImsInnovation Developer Portal. Last accessed: 22.04.2010. [Online]. Available: http://developer.labs.ericsson.net/apis/mjcf

[23] The Sailfin Project website. Last accessed: 09.06.2010. [Online]. Available: https://sailfin.dev.java.net

[24] Ericsson Multi Activation. Last accessed: 09.06.2010. [Online]. Available: http://www.ericsson.com/ourportfolio/network/multi-activation-ema?nav=networkareacategory000

[25] Project Atmosphere. Last accessed: 09.06.2010. [Online]. Available: https://atmosphere.dev.java.net

[26] Project Grizzly. Last accessed: 09.06.2010. [Online]. Available: https://grizzly.dev.java.net

[27] flensed. Last accessed: 09.06.2010. [Online]. Available: http://www.flensed.com