Device Discovery in Future Service Platforms through SIP

Yuan Chen
Department of Electronic Engineering
University of Surrey
Guildford, UK
chyu1988@gmail.com

Suparna De, Ralf Kernchen, Klaus Moessner
Center for Communication Systems Research
University of Surrey
Guildford, UK
{S.De, R.Kernchen, K.Moessner}@surrey.ac.uk

Abstract—**This paper proposes an extension to Session Initiation Protocol (SIP) for contextualized service delivery in a service delivery platform (SDP) that enables device specific multimedia delivery. SIP separates between session establishment and description and is thus, amenable to be extended for advanced implementations which make it an ideal platform for service creation. Device specific multimedia delivery needs rich and flexible device descriptions, and our approach proposes advanced device descriptions through semantic technologies. The proposed SIP extensions have been implemented on a SIP Application Server which functions as SDP in IP Multimedia Subsystem (IMS). The validation of the proposed extensions is shown through an Android SIP client application that acts as a device browser and recommender for different multimedia services to users. An example device user agent (UA) application has also been implemented on a laptop.**

Keywords-device description; Ericsson SDS; Google Android; IMS; ontology; SIP extension

I. Introduction

Convergence of multimedia services and IP communication is becoming a growing trend in the telecommunication industry. Simultaneously, increasing device diversity and multiplicity offers opportunities as well as challenges for service delivery personalization. In such a scenario, Service Delivery Platforms (SDP) can offer flexible creation and delivery of multimedia services [1]. A SDP provides service management across content-based and session-based services, with network layer abstractions and agility in service provisioning. The combination of SDP and IP Multimedia Subsystem (IMS) gives the operators both wider service choices and easier network management, with a unified control layer for users connecting to the network [2]. Personalization techniques need to be integrated into SDPs for service differentiation. Also, with a large variety of devices available to users in the current environment, personalization needs to be enabled for multi-device scenarios.

The key technology behind IMS is Session Initiation Protocol (SIP) [3]. SIP is used for establishing, modifying and terminating multimedia sessions that are based on IP network. With a wide range of devices supporting SIP, they may have different capabilities for different multimedia purposes such as streaming multimedia, file transfer, etc. Devices in the SIP network are known as user agents (UAs) and identified by a uniform resource identifier (URI). Hence, it is necessary for clients to discover the available devices and their capabilities from a SDP in order to establish personalized multimedia services to relevant devices.

SIP, as defined in RFC 3261, [3] is a control protocol on application layer. SIP itself does not provide any services, being only involved in the signaling portion of a communication session. Rather, it is used in conjunction with other IETF protocols to build a complete multimedia architecture, e.g. SIP can locate a user, and another protocol such as Real-Time Protocol (RTP) or HTTP can be used to deliver streaming media to that user. Since the original purpose of SIP is to control sessions over the Internet, it doesn't provide any methods to convey device capabilities. Hence, an extension of core SIP is necessary. Moreover, in order to describe the multimedia capability of devices, a device description approach is needed. The approach should be extensible and manufacture-independent to ensure unified interoperability. Moreover, it must be comprehensive to describe multimedia features about SIP devices, including supported multimedia services, so that varying multimedia service context can be matched to different available devices.

This publication proposes a SIP extension that can convey device capabilities to enable device specific multimedia delivery. We employ OWL ontology to provide a semantic framework for device descriptions. The proposed extensions have been implemented using Ericsson Service Development Studio (SDS) [4] on a SIP Application Server (AS) which functions as SDP in IMS. We validate our approach by developing an Android client that discovers SIP UAs (also known as device UAs) and their device descriptions from a SDP by employing the extended SIP messages so that the device specific multimedia delivery can then be successfully established.

The rest of the paper is structured as follows: we review related work on SIP extensions and device descriptions in section II. Section III discusses the design of the proposed SIP protocol extension. The device description approach is discussed in section IV. The system overview, implementation and demonstration are illustrated in sections V, VI and VII, respectively. Section VIII concludes this paper with a discussion of the contributions.

This paper describes work undertaken in the context of the Internet of Things – Architecture (IoT-A) project (http://www.iot-a.eu/public) contract number: 257521.

II. Related Work

The core SIP protocol is designed to be used in every SIP implementation. Every SIP UA or server should understand the standard declared in RFC3261 [3]. When core SIP is no longer capable of resolving a specific problem, an extension of SIP is necessary with new methods, new header fields, new body types, or new parameters following RFC 4485 [5], which is defined to ensure that new extensions do not change the spirit of SIP. To describe device capabilities, RFC 3840 [6] defines a mechanism for SIP UA to convey its capabilities and characteristics to other UAs and to the registrar of its domain. The general idea of this specification is to convert the description information into parameters in the contact header field. However, a SIP UA implementation of this specification would require converting all the description information into the contact header. Instead, storing all the information about device description in the body of a SIP message and using a more generic syntax (XML) can offer a cleaner approach for SIP UAs. Therefore, we propose to extend the OPTIONS and MESSAGE methods with a new 'description' header for conveying device capabilities. The use of XML for exchanging information in a platform independent manner has also been proposed in [7], where XML was employed for communication over a proposed new interface in the IMS Policy and Charging Control (PCC) architecture [8]. The proposed interface was envisioned to take into account dynamic subscriber information to guide online charging policies. Subscriber information in IMS is held in the Subscription Profile Repository (SPR) [8]. The SPR provides static data such as allowed usage threshold as well as dynamic data on counter values and status of valid tariff options. In addition, for application sessions, the Application Function (AF) [8] may provide information on media type and format, bandwidth and flow description and status. A well-structured, machine-processible knowledge of device capabilities available to a user can supplement the SPR and AF information and enable contextualized delivery.

Other efforts for personalization using SIP extensions include that by Kuhnen et al. [1] who describe a method to manage user preferences and device capability profiles for a personalized multi-device environment. They use the IMS infrastructure and SIP PUBLISH messages for transmitting static and dynamic device capabilities described using the User Agent Profile (UAProf) [9] vocabulary. After registration, the device sends a PUBLISH message which contains a UAProf device description and the changed dynamic capabilities to a Personalized Communication Controller in IMS. By applying the personalization in the core network, changes in preferences or device capabilities can immediately affect running sessions, without having to renegotiate with the other involved parties. Based on the SIP event notification framework, an event package extension called information category is proposed in [10] to enable multimedia information sharing in IMS. The end user can use the SIP SUBSCRIBE method to subscribe to a specific information category, get information from the application server via NOTIFY, and post information with PUBLISH. The service also provides a method to support audio, video and large text sharing between end users. The UA can upload media through the PUBLISH method, with the media URI. Thereafter, the server can publish the information with NOTIFY. However, it uses Session Description Protocol (SDP) negotiation to establish multimedia sessions, which is not easy to be extended and has strict parsing rules. It does not provide any method for device description and multimedia content matching on SDP.

Several standardized vocabularies and techniques for device capability description have been designed, including Composite Capabilities/Preferences Profile (CC/PP) [11], UAProf and Universal Plug and Play (UPnP) [12]. However, as noted in [13], none of these approaches can be used as a mainstream solution to describe multimedia devices. Specifically, CC/PP defines profiles, profile components and attributes, but the actual vocabulary of which device features can be described is left open. Also, the CC/PP template is not intuitive for building comprehensive descriptions. UAProf extends CC/PP with device capability names and allowed values, but is tailored to Wireless Application Profile (WAP) devices and does not scale well to more complex devices. The UPnP XML-based description template provides an interesting concept of device composition, with each being capable of being discovered and used independent of the container device. However, with the service-based focus of the description template, hardware specification is not intuitively included. Moreover, both CC/PP and UAProf are vocabularies, while UPnP discovery is targeted more towards a LAN scope. These existing description methods are reused and extended in this paper, with an emphasis on multimedia capabilities to extend SIP discovery to cope with device specific characteristics.

III. SIP Protocol Extension Design

We extend the SIP methods and headers for device and multimedia content discovery. The SIP OPTIONS method is used to query a server about its capabilities. This has been extended to query the capabilities of registered UAs on the server. The SIP MESSAGE request is defined to carry the sender's message in its body. Normally, it is used to deliver instant messages. This method is extended to carry the device and media descriptions. Besides these extensions of the SIP methods, the SIP headers are also extended to distinguish between the normal SIP methods and extended ones. The newly added 'description' header is used for the following three purposes:

A. Complete Device Description Upload

The server gets to know the capabilities of registered devices via the device description uploading process (Fig. 1).

After the standard SIP registration, a device UA uploads its full device description information to the SDP server. The information is carried inside a MESSAGE request with the proposed 'description' header with value equal to the UA's URI. Thus, for a device UA with SIP URI sip:example@mydomain.com, the description header is 'description:sip:example@mydomain.com'. URIs help the server to differentiate between different registered SIP UAs.

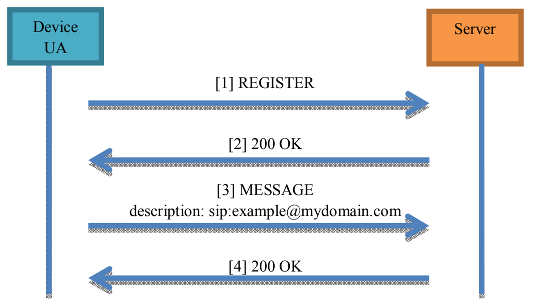


Figure 1. SIP extension: Device UA registration with its device description

B. *Request/Response for List of Registered Device UAs*

This extension allows a user to see a short summarized list of available devices on the client GUI (device browser). The device list, which is customized to show devices available in the requesting client UA's location, shows the registered devices with their SIP URI, model name and a sub-text containing device type and media capabilities. The list of available devices is presented as a vertically scrolling list of clickable items. This available device list request and response is achieved through the proposed header 'description:list' (Fig. 2). The extended OPTIONS method with the 'description: list' header is sent by the client to request the list of registered devices. After sending a '200 OK' confirmation response, the server sends the device list via a MESSAGE method which also has header 'description:list', which is followed by a client confirmation message.

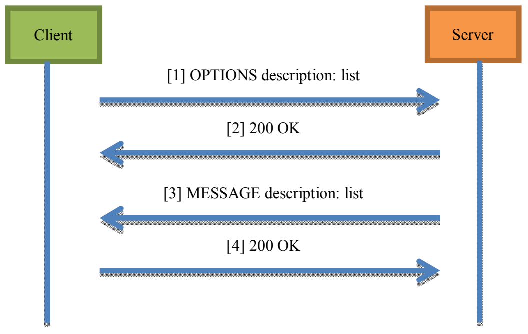
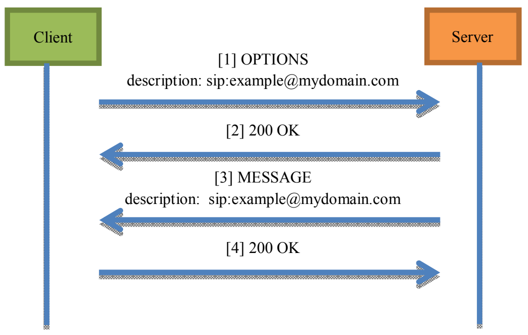


Figure 2. SIP extension: list information retrieval

Thereafter, if the user is interested in a device's complete description, he can click on the corresponding item, which results in a request being sent to the server to retrieve the complete device description and display it in another GUI view. For this, an OPTIONS request is sent to the server with the corresponding device URI indicated in the description header (in this case, the description of UA sip:example@mydomain.com) (Fig. 3). Then the server sends back the complete description of this UA's device.

The extended OPTIONS and MESSAGE methods are also used to retrieve the list of available media resources, described by name, short description and category. Clicking on a media item triggers the recommender function, which matches the media description to available device context and shows a pop-up list of devices recommendations, to which the media item can be streamed to.


Client
Server
[1] OPTIONS
description: sip:example@mydomain.com
[2] 200 OK
[3] MESSAGE
description: sip:example@mydomain.com
[4] 200 OK
Figure 3. SIP extension: complete device description retrieval

C. *Multimedia Streaming Notification*

Once the user has chosen a device from the recommended list of suitable devices for playing a particular media item, multimedia streaming is achieved by sending a MESSAGE request with header 'description:stream' from the client to the server. This message contains the media item as well as destination device UA information.

IV. ONTOLOGY DESIGN FOR DEVICE DESCRIPTION

We propose to use the Web Ontology Language-Description Logics (OWL-DL) for describing device multimedia capabilities. Ontology is a vocabulary that defines common concepts in a particular domain and the relationships among those concepts [14]. With heterogeneous devices of widely varying capabilities existing in the target communication environment, the ontology formalism allows a structured representation to information from diverse sources. Moreover, it promotes separation of domain knowledge (device description knowledge) from operational knowledge (software implementation) and the possibility to analyze domain knowledge once a declarative specification of the terms is available [14]. Each device UA hosts its own description ontology instance. After registration, it is uploaded to the server. The server can also obtain the capabilities from a web server if the device does not upload its capabilities but indicates the URI of its description.

The proposed domain ontology consists of the following three main classes: the 'DeviceUA' class has several data type properties to describe the model information of SIP device instances such as model name, manufacturer, device type, short description, etc. Besides, it also has two object properties to the 'DeviceList' and 'ServiceList' classes. The 'DeviceList' class has two sub-classes for describing hardware and software information of SIP devices. The hardware information is further described by subclasses including 'ConnectionTypes' (Wi-Fi, Ethernet and Bluetooth), 'CPU', 'Memory' and 'UserInterfaces' (audio I/O, camera, keyboard, pointing_device and Screen). Each of the classes has corresponding datatype properties to capture the related characteristics. For example, class 'Screen' has properties to describe its bits per pixel (BPP), resolution, color and dimension. The 'ServiceList' class describes the services supported on a device including file transfer, instant message, audio and video service. A partial view of the ontology is shown in Fig. 4.

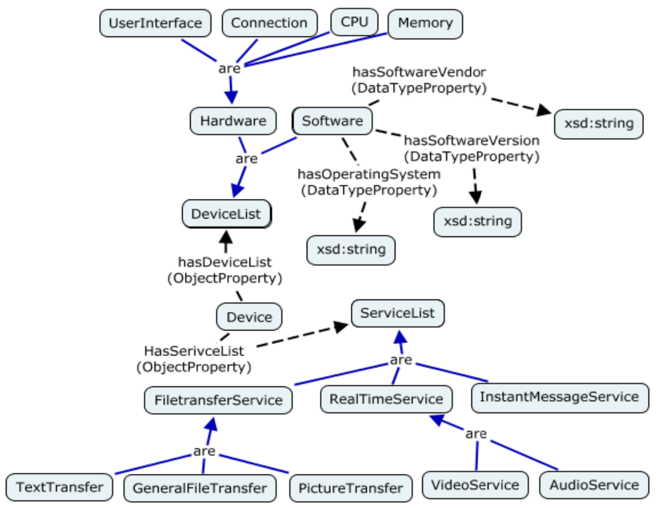


Figure 4. Device ontology

V. System Overview

The proposed SIP extensions and device description approach have been realized as a SIP server side application. The implemented system consists of three components: client UAs, SIP server side application and device UAs. The server side application is developed on Ericsson SDS [4] which simulates several different components of IMS such as Call Session Control Function (CSCF), DNS, Home Subscriber Server (HSS), SIP Registrar and integrates a SAILfin SIP container. It enables rapid design and test of IMS applications through high level APIs.

The client UA allows a user to configure the SIP related parameters, register or de-register with the server. It discovers the registered device UAs and media resources from the server. It then displays the general device and media information into different list views and presents the complete device description into web views. It also presents recommendations of relevant device options to the user for playing the different discovered streaming media resources, and sends the streaming preference notification to the server.

The IMS server-side application enables the logic of extended SIP signaling with client UAs and device UAs. It accepts register/de-register messages from the client and device UAs. It receives, creates and processes the device description ontology files from device UAs, creates HTML-based complete device description for these UAs. It also stores local multimedia resources. Based on an evaluation of the device capabilities, it can generate possible device recommendations of streaming these resources. When it receives a streaming preference notification from the client, it establishes a multimedia streaming session by SIP and opens a HTTP port to stream the multimedia file to the selected device UA. The Device UA refers to the SIP user agent of a device which can be used to demonstrate multimedia playback. It can register or de-register with the server, upload its ontology file to the server and play multimedia resources depending on its capabilities as described by its ontology instance.

VI. Implementation

The server-side application is developed using JSR 289 SIP servlet API [15]. It can be divided into three main modules (as shown in Fig. 5): SIP servlet, ontology handling module and media handling module.

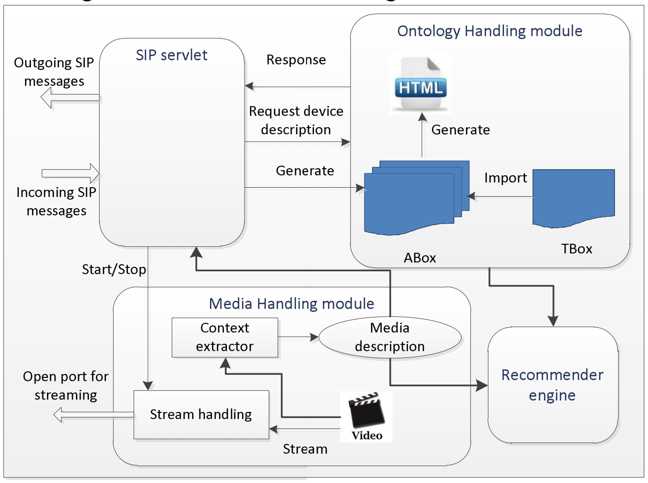


Figure 5. SIP Server-side application architecture

The SIP servlet is responsible for receiving SIP messages from client or device UA via CSCF and sending the response back. It also controls the other two modules. The ontology handling module is responsible for parsing the ABox files uploaded by device UAs. The parsed information is passed on to the SIP servlet to generate the discovered device list and to create a HTML representation of each ABox. The discovered device capabilities are also employed by the Recommender engine for reasoning about devices that are capable of streaming specific multimedia files from the multimedia repository. The Media handling module receives the command from SIP servlet to start or stop multimedia streaming by opening a local HTTP port using VLCJ API [16]. Since so far, there's no automatic multimedia resource context retrieval function available, this module implements a context extractor to provide basic media information such as name, location, description and category. The service provider can specify local media resources manually according to this structure.

The SIP client has been implemented as an Android application running on a Samsung Galaxy tab. The device UA runs on a laptop. Both of these applications use JAIN SIP API [17] for the SIP stack layer. Their respective main UIs are shown in Fig. 5. The Android client can be divided into two layers: SIP communication stack layer - this layer is located between the transport and application layers. It retrieves SIP related parameters (nick name, SIP user name, domain name and port number) from the UI layer and creates the SIP stack. It listens on the SIP stack, sends and receives SIP messages and handles the SIP timers.

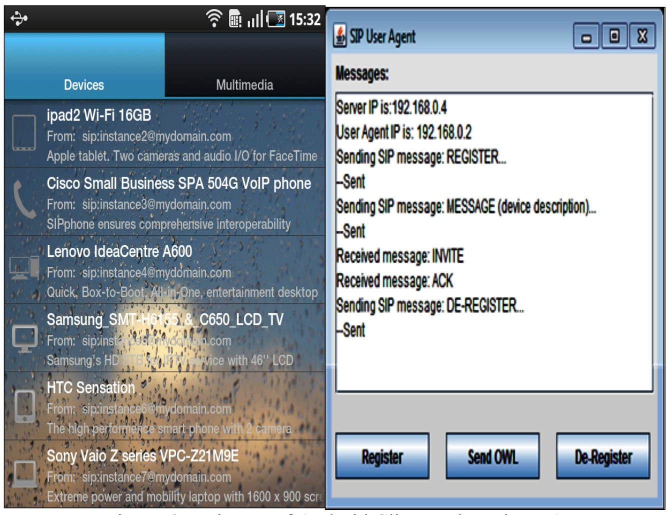
UI layer (Android activities): the main activity TabUI hosts two tabs for the ClientUI and MediaUI, respectively.

The ClientUI implements the device browser which displays a list view of discovered devices. It listens on the SIP stack, retrieves and processes SIP messages. Then, it updates its list or starts the DeviceDescription activity which displays a web view to render the received string into a web page when the complete device description is received. The MediaUI activity tab displays a list view for discovered media resources. When one of the media items is selected, MediaUI shows a dialog for the device recommendations for multimedia streaming. When the user chooses one of the options, the MediaUI sends a 'stream' message to the server with the selected device and media URI.

The device UA application has the same layered structure as the client. Besides, it implements a VLCJ media player to play video on receipt of an INVITE SIP message from the server.

VII. Demonstration

In the demonstration scenario, the device UA registers and uploads its instance ontology when the user presses the 'Register' and 'Send OWL' keys on the UI, respectively (Fig. 6). The server stores and processes the received instance ontology and generates a human readable HTML file for this ontology in preparation for full device description request from client. If the client is registered, it sends the discovered device list information to the client. The Android client application processes the list message and displays it into an Android list view (Fig. 6). Clicking on a list item displays a HTML web page with the detailed device description. To play a multimedia resource, clicking the corresponding item displays the relevant device recommendations in a dialog. After the user makes a choice, the chosen video is streamed to the designated device UA.



ipad2 Wi-Fi 16GB
From: sip:instance2@mydomain.com
Apple tablet. Two cameras and audio I/O for FaceTime

Cisco Small Business SPA 504G VoIP phone
From: sip:instance3@mydomain.com
SIPphone ensures comprehensive interoperability

Lenovo IdeaCentre A600
From: sip:instance4@mydomain.com
Quick, Box-to-Boot, All-in-One, entertainment desktop

Samsung_SMT-H6155_&_C650_LCD_TV
From: sip:instance5@mydomain.com
Samsung's HD STB for IPTV service with 46'' LCD

HTC Sensation
From: sip:instance6@mydomain.com
The high performence smart phone with 2 camera

Sony Vaio Z series VPC-Z21M9E
From: sip:instance7@mydomain.com
Extreme power and mobility laptop with 1600 x 900 scr

```
Server IP is:192.168.0.4
User Agent IP is: 192.168.0.2
Sending SIP message: REGISTER...
--Sent
Sending SIP message: MESSAGE (device description)...
--Sent
Received message: INVITE
Received message: ACK
Sending SIP message: DE-REGISTER...
--Sent
```

Figure 6. Main UIs of Android Client and Device UA

VIII. Conclusions

In this paper, we defined a SIP extension to enable device and multimedia resource discovery. This paper has also presented an ontology approach for describing a SIP device UA. Though the demonstration has focused on showcasing the SDP aspects and not on the IMS functionalities, the proposed ontology-based device description mechanism can enhance the IMS admission control framework for multimedia sessions by supplementing the SPR and AF information to provide personalized multimedia streaming.

References

[1] M. Q. Kuhnen, D. Kraft, A. Schulke, J. Bauknecht, J. Haussler, and M. Lischka, "Personalization-based Optimization of Real-time Service Delivery in a Multi-Device Environment," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2009.

[2] Z. Technologies. (2008), IMS and SDP: The Key Enablers of Business Re-engineering. Available:http://wwwen.zte.com.cn/endata/magazine/ztetechnologies/2008year/no4/articles/200804/t20080424162019.html

[3] J. Rosenberg et al., "SIP: Session Initiation Protocol," RFC3261, 2002.

[4] Ericsson, "Service Development Studio (SDS) 4.1 Developer's Guide," 2009.

[5] H. Schulzrinne and J. Rosenberg, "Guidelines for Authors of Extensions to the Session Initiation Protocol (SIP)," RFC4485, 2006.

[6] J. Rosenberg, H. Schulzrinne, and P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)," RFC3840, 2004.

[7] M. Cheboldaeff, "Interaction between an Online Charging System and a Policy Server," in *The Tenth International Conference on Networks (ICN)*, 2011, pp. 47-51.

[8] 3GPP, "Policy and charging control architecture (TS 23.203) Rel. 10," 2010.

[9] Open Mobile Alliance (OMA). (2003), User Agent Profile. Available: http://www.openmobilealliance.org

[10] W. Xiao, "Categorized Multimedia Information Sharing Service in IMS," in *Australasian Telecommunication Networks and Applications Conference*, Adelaide, SA, Australia, 2008, pp. 103-105.

[11] W3C, "Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies," W3C Working Draft, 2002.

[12] UPnP Forum. (2000), UPnP Device Architecture. Available: http://www.upnp.org/download/UPnPDA10_20000613.htm

[13] R. Kernchen, M. Boussard, R. Haensel, and K. Moessner, "Device description for mobile multimodal interfaces," in *Proc. 15th IST Mobile & Wireless Communication Summit*, 2006.

[14] N. F. Noy and D. L. McGuinness, (2001), "Ontology Development 101: A Guide to Creating Your First Ontology." Stanford Knowledge Systems Laboratory Technical Report. Available: http://www.ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness.pdf

[15] JSR 289: SIP Servlet v1.1. Available: http://www.jcp.org/en/jsr/detail?id=289

[16] VLCJ Wiki: Streaming. Available: http://code.google.com/p/vlcj/wiki/Streaming

[17] JSR 32: JAIN SIP API Specification. Available: http://jcp.org/en/jsr/detail?id=32

]