D.4.2.1

Overall Network Middleware Requirements Report

Version 1.0

WP 4.2 Overall Network Middleware Requirements Report

Dissemination Level: Public

Lead Editor: Telenor

13/05/2005

Status: V1.0 Final

SIXTH FRAMEWORK PROGRAMME

PRIORITY IST-2002-2.3.1.18

Grid for complex problem solving

Proposal/Contract no.: 004293

Context

Activity 4	Detailed Architecture, Design and Implementation
WP 4.2	Mobile Network Middleware Architecture, Design and Implementation
Dependencies	WP3.1

Contributors:	*Reviewers:*
Dirk Haage (UPM) Vicente Olmedo (UPM) Víctor A. Villagrá (UPM) Jose I. Moreno (UPM) Jan Wedvik (Telenor) Per-Oddvar Osland (Telenor) Babak Farshchian (Telenor) Brynjar Viken (Telenor) Cristian Morariu (University of Zurich) Martin Waldburger (University of Zurich) Giovanni De Martino (CRMPA) Nuno Inacio (IT Aveiro) Isabel Alonso (TID) Arantxa Toro (TID)	Internal review by WP4.2 participants.

Approved by: Jan Wedvik, WP4.2 leader.

Version	Date	Authors	Sections Affected
1.0	2005-05-13	Se table above.	Final release.

b. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works.
c. For the avoidance of doubt, where the work is a musical composition:
    i. **Performance Royalties Under Blanket Licenses**. Licensor waives the exclusive right to collect, whether individually or via a performance rights society (e.g. ASCAP, BMI, SESAC), royalties for the public performance or public digital performance (e.g. webcast) of the Work.
    ii. **Mechanical Rights and Statutory Royalties**. Licensor waives the exclusive right to collect, whether individually or via a music rights society or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions).
d. **Webcasting Rights and Statutory Royalties**. For the avoidance of doubt, where the Work is a sound recording, Licensor waives the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions).

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats, but otherwise you have no rights to make Derivative Works. All rights not expressly granted by Licensor are hereby reserved.

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any reference to such Licensor or the Original Author, as requested.
b. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work or Collective Works, You must keep intact all copyright notices for the Work and give the Original Author credit reasonable to the medium or means You are utilizing by conveying the name (or pseudonym if applicable) of the Original Author if supplied; the title of the Work if supplied; and to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless

such URI does not refer to the copyright notice or licensing information for the Work. Such credit may be implemented in any reasonable manner; provided, however, that in the case of a Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE MATERIALS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.
b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

a. Each time You distribute or publicly digitally perform the Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
b. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

Executive Summary

This document describes the functionality that will be provided by the *network middleware layer* of the Akogrimo architecture. The network middleware layer shall provide a set of functions to the upper layers, allowing Akogrimo to present several enhancements to the standard GRID architecture (i.e. OGSA). Specifically, the network middleware layer will offer:

- Cross layer A4C (Authentication, Authorization, Accounting, Auditing, and Charging).
- Service discovery based on semantic service descriptions and queries.
- Presence and context management.

The cross-layer A4C service serves several purposes. Akogrimo intends to target mass-market services for mobile and roaming users. A single sign-on mechanism will make Akogrimo services more user-friendly, and hence lower the barrier for accepting such services. By having a unified authorization mechanism across all layers, access control rules are easier to manage, and security loopholes are less likely to appear. Cross-layer accounting and charging provides flexibility in tariffing schemes, and in how value chains are composed.

Traditional service discovery mechanisms only describe the syntax of a service in machine-readable form. The semantics of the service must be defined through some off-line agreement. Akogrimo will go beyond this by using a service discovery mechanism where services are described and located through specifications of service semantics. Service semantics will be expressed through onotologies. Ontologies for new application domains may be added as needed.

While much prior Grid research has focused on batch-mode supercomputing applications, Akogrimo introduces the mobile Grid, where interactive services involving mobile and nomadic users are of key importance. The lives of humans are much more varied and dynamic than those of computational resources, so when humans are introduced as resources in the Grid, there is a need to keep track of their context. By knowing this context, the system may easier choose the right people to participate in a workflow, and better adapt service behaviour to the situation of those people. Context has many facets:

- Presence: Is the user logged on? Is he idle or busy?
- Physical properties: User location, local time, body temperature etc.
- Device context: What terminals and other I/O units are available to the user and what are the hardware and software capabilities of those devices?

The definition of context is open-ended; the set of data that should be monitored will depend on the application domain. Akogrimo shall therefore provide a context infrastructure which allows relevant parties to keep track of user context in terms of (SIP-

)presence, user location and device capabilities. The context will be expressed in an ontology, which allows for extensions to cover domain specific context data.

Table of Contents

List of Figures

List of Tables

Abbreviations

A table with used abbreviations is strongly recommended

Term	Full text
A4C	Authentication, Authorization, Accounting, Auditing and Charging
Akogrimo	Access To Knowledge through the Grid in a Mobile World
API	Application Programming interface
CM	Context Manager
EMS	Emergency Medical Service(s)
GrSDS	Grid Service Discovery System
GSDS	General Service Discovery System
IETF	Internet Engineering Task Force
LSDS	Local Service Discovery System or Life Signs Detection System
NGG	Next Generation Grid
OGSA	Open Grid Services Architecture
OWL	Ontology Web Language
PA	Presence Agent
PANA	Protocol for carrying Authentication for Network Access
PIDF	Presence Information Data Format
P2P	Peer to Peer
PS	Presence Server
PUA	Presence UA
QoS	Quality of Service
RDF	Resource Description Framework

Term	Full text
RFC	Request For Comment
RFID	Radio Frequency Identification
RPC	Remote Procedure Call
RTP	Real Time Protocol
RTCP	Real Time Control Protocol
SAML	Security Assertions Markup Language
SD	Service Discovery
SDP	Session Description Protocol or Service Discovery Protocol
SIMPLE	SIP for Instant Messaging and Presence Leveraging Extensions
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SLP	Service Location Protocol
SOAP	Simple Object Access Protocol
UA	User Agent
UAC/UAS	UA Client / UA Server
UAProf	User Agent Profile
UML	Unified Modelling Language
UDDI	Universal Description, Discovery and Integration
UpnP	Universal Plug and Play (Microsoft)
VO	Virtual Organization
WAP	Wireless Access Protocol
WS	Web Services

Term	Full text
WSDL	Web Services Description Language (XML format for describing network services)
WSMF	Web Service Modeling Framework
WSMO	Web Service Modeling Ontology
WSRF	Web Services Resource Framework
XML	Extensible Markup Language

1. Introduction

The 6th Framework program IP *Akogrimo – Access to Knowledge through the Grid in a mobile World* – aims on the *technology level* at the integration of mobile communication into the Open Grid Services Architecture (OGSA). On the *application level* Akogrimo aims at validating the thesis that the vision of Grid-based computing and the future development of Grid technologies as well as the development of Grid infrastructures can and will substantially draw from the integration of a valid mobility perspective.

What/purpose

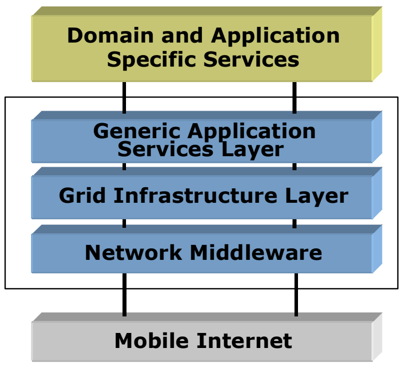
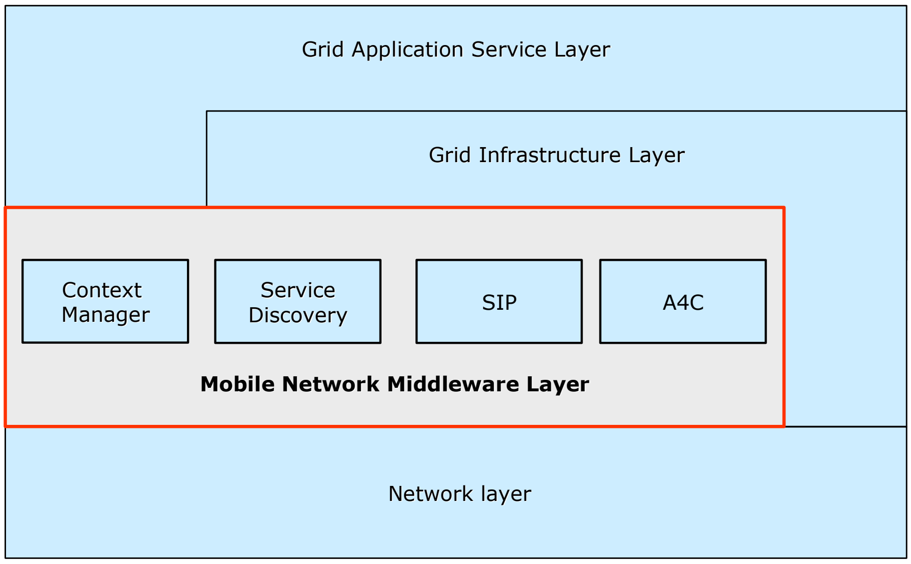

Domain and Application Specific Services
Generic Application Services Layer
Grid Infrastructure Layer
Network Middleware
Mobile Internet

Figure 1, overview of Akogrimo layers.

Figure 1 provides an overview of the Akogrimo architecture. The focus of the Akogrimo project is on the three middle layers, namely:

- The Network Middleware layer: This layer provides a set of basic infrastructure functions to the above layers, including cross-layer A4C, presence and context management, and semantic service discovery. This layer is also the topic of this document.
- The Grid Infrastructure Layer: This layer provides the Grid services/Web services infrastructure (as defined by WSRF/OGSA), execution management (allocating jobs to available resources), data management (replication, fragmentation, unification of heterogeneous storage), and Service Level Agreement (SLA) monitoring and enforcement.
- The Generic Application Services Layer: This layer provides Virtual Organization (VO) management and workflow management (orchestration).

The purpose of the Mobile Network Middleware layer is to provide a user-centred and programmer-friendly service platform for Akogrimo offering new opportunities towards the Grid layers. An overview of the mobile network middleware architecture is shown in Figure 2. The middleware platform will bring components such as session management, service discovery, A4C (Authentication Authorization, Accounting,..) and context management into the Akogrimo overall architecture. In particular, it is important to clarify the role of SIP in the overall Akogrimo architecture. This deliverable provides a detailed architecture and design (UML models, interface definitions, choice of 3rd party products to integrate, etc.) for the Mobile Network Middleware layer. An overall design of the middleware layer is proposed and each of the subsystem identified will be discussed in detail. For each subsystem, its scope, function and role will be assessed. Furthermore, relations and interfaces to other subsystem within the middleware layer and to other layers of the Akogrimo architecture (what is required from and what is provided to other subsystems, dependencies etc.) will be outlined. Use cases will be used to analyse how the different subsystems interact with each other and systems on higher and lower protocol layers.



Grid Application Service Layer
Grid Infrastructure Layer
Context Manager
Service Discovery
SIP
A4C
Mobile Network Middleware Layer
Network layer

Figure 2 Main components of Akogrimo Network Middleware architecture

The Mobile Network Middleware layer will be based on the results provided by Akogrimo's mobile network activities and will target higher layers to provide an integrated access point to Grid Infrastructure Layer and Grid application support layer.

Why

The Akogrimo middleware platform is the link currently missing between Grid layer and mobile network services. The architecture will augment the Grid-specific parts of the architecture with innovative components and services for user-centred and user-friendly access to the Next Generation Grid (NGG). The developed service platform will provide user-centred abstractions of services for programming the NGG. These abstractions will be available for programmers in form of APIs and SDKs. In this way application programmers can focus on their applications and their user requirements when developing NGG applications.

The innovative aspects of this service platform include its strong focus on user needs. The platform will not only provide access to conventional grid services such as resource discovery, usage and management, but will implement additional services aimed at increasing the user-friendliness of NGG. These additional services include support for personalization and gradual adaptation to evolving user and organization needs, context-awareness, knowledge discovery and human collaboration support.

2. Session Initiation Protocol and Simple Object Access Protocol

2.1. Introduction

SIP is the standard protocol for session setup and management and offers potentials for current and future mobile networks with its support of mobility and localization mechanisms. SIP session means the capability of exchange media information between SIP entities. SOAP on the other side supplies the framework for GRID computing and distributed services. This Chapter aims to describe possible combinations of these two protocols and to identify their value for the Akogrimo network.

SIP is the de-facto standard protocol for session setup and session management. Furthermore, the SIP architecture provides services like localization and with that, mobility management. In combination with SDP, the session description protocol, it is also possible to provide some context-awareness. The protocol is also simple and extensible to fulfil any future requirements.

SOAP as a higher level application protocol builds the framework for GRID computing as a transport of Web Services and Grid Services.

The goal of Akogrimo is to merge these two worlds, the world of mobility and the world of GRID computing (distributed computing). The usage of SIP and SOAP is a logical step to achieve this. Therefore, the following sections describe the two protocols in short and present different possibilities to merge their advantages and to add missing features to each other.

2.2. SIP overview

The Session Initiation Protocol (SIP) is described with more detail in the deliverable 4.1.1. This section will only give a short overview about the protocol with a particular focus on the features important for the combination with SOAP.

SIP is a protocol for initiating and managing sessions between multiple entities. The protocol is text-based and similar to HTTP by reusing its message structures and error codes. As SIP does not define a “Type of Session” by default, it is usable for any kind of service needing management of sessions, like audio/video conferencing or interactive games, and/or nomadic actors, like instant messaging.

The SIP architecture provides user localization services via SIP registrars. In combination with the session management capabilities, it provides several levels of mobility; in particular user and session mobility (i.e. nomadic actors, client mobility is better managed with Mobile IP (MIP)).

But SIP does only provide a “virtual” location, that means, it can provide the current access network of a user/terminal/service, but not its location in the real world (e.g. geo coordinates or “UPM Building B, office B.217”). This information can be published via SIP (cf. 2.2.2.2 SIP extensions for context management)

Context-awareness in the term of terminal capabilities is managed via SDP (Session Description Protocol) which is used during the setup of sessions. If any of the required capabilities can not be provided by the mobile terminal, some re-negotiation has to take place. This could be a setup of a reduced session (e.g. simple audio instead of video communication) or the localization of a more appropriate device (e.g. the big display to view the simulation results) - by providing device capabilities over enriched context information and further session transference – and further session transference to the other device.

2.2.1. SIP session management capabilities

The main goal of the SIP protocol is to provide session management capabilities. This means that using SIP the establishment, negotiation or termination of sessions between SIP entities are possible. In this sense SIP is a very suitable solution for managing multimedia sessions.

Furthermore, SIP provides the possibility for session renegotiation. There are several circumstances that can trigger a session renegotiation:

- The user moves to a new access network in which there are no available resources to achieve the requested QoS.
- The user transfer the ongoing session from a device to another with different capabilities.
- The IP address of the user changes due to a terminal or session handover.
- The end user application decides to remove one particular media stream from the communication (i.e. the video part from a mixed audio/video stream), either fully or to move this particularly stream to another device.
- Inviting new parties to or resigning from an ongoing session

SIP does not control the exchanging of the data information itself, this is done by other protocols like RTP (Real Time Protocol) and RTCP (Real Time Control Protocol) [RFC3550].

The explicit changes in a session (setup, move, suspend, etc.) are handled by the application by using the SIP protocol. SIP is not able to manage sessions transparent to the application (e.g. like MIP provides transparent terminal mobility). The applications need to be aware of changes in a session in order to handle them correctly. For example, moving a video stream from a mobile terminal to big screens involves the change of the endpoint (IP address and port) of the RTP stream.

Most of the current applications use this mechanism also to implement the **hold/resume** functionality. This can be achieved by setting specific values in some fields of the SDP body (in concrete the field port, which is set to zero to indicate a hold).

2.2.2. SIP extensions

Several extensions were already developed to provide additional features for SIP. Some of them are mandatory for Akogrimo, e.g. the extensions for presence information (SIMPLE), some of them are currently developed and may be useful for Akogrimo, e.g. context information management support for SIP ("Presence Information Data Format" (PIDF) [RFC3863] and IETF Internet Draft "Rich Presence Extensions to the Presence Information Data Format" (RPID) [PIDF]).

The following sections will explain these extensions with a special attention to their usage within Akogrimo.

2.2.2.1. SIMPLE

The SIP Instant Messaging and Presence Leveraging Extension (SIMPLE) provides the functionalities to send and receive real-time messages via SIP and to publish and request the current status of a user as well as subscription and notification of changes in the status. These protocol extensions are described in [RFC3428 "Session Initiation Protocol (SIP) Extension for Instant Messaging"[RFC3428] and [RFC3856] "A Presence Event Package for the Session Initiation Protocol (SIP)" [RFC3856]. The primary purpose of SIMPLE within Akogrimo is the provision of presence information. The next two sections show the basic mode of operation of the SIMPLE presence architecture as well as its components needed in addition to the standard SIP entities and some possibilities to use this mechanism to provide more sophisticated context information like device capabilities, location, etc.

2.2.2.1.1. Operation Model

The operation model is the Presence Service described in [RFC2778], the system allows users to subscribe to the presence status of each other and be notified when a change in that state occurs.

The presence service has two distinct sets of "clients":

- **Presentities**, which provide presence information to be stored and distributed.
- **Watchers**, which receive presence notifications. There are two kind of watchers:
    - **Fetchers** that simply request the current value of some presentity's presence information from the presence service.
    - **Subscribers** that requests notification from the presence service of (future) changes in some presentity's presence information (this changes to presence information are distributed to subscriber via notifications).

The same entity can play the presentity and watcher role. [RFC3856] and [RFC3856] introduces some additional definitions:

- **Presence User Agent (PUA):** this element manipulates presence information of a presentity. Multiple PUAs are allowed per presentity, which means that many user devices can provide a portion of the overall presence information of the user.
- **Presence Agent (PA):** A presence agent is a SIP entity which is capable of receiving presence requests, responding to them, and generating notifications of changes in presence state.
- **Presence Server (PS):** A presence server is a physical entity that can act as either a presence agent or as a proxy server for presence requests. When acting as a PA, it is aware of the presence information of the presentity through some protocol means. When acting as a proxy, the presence requests are proxied to another entity that may act as a PA.
- A User Agent Client (UAC) that publishes event state is labelled an **Event Publication Agent (EPA).** For presence, this is the Presence User Agent (PUA). The entity that processes the presence publication messages is known as an **Event State Compositor (ESC).** For presence, this is the role of the Presence Agent (PA) defined in [RFC3856] [RFC3856].

The PA can obtain presence information from a PUA in several ways ([RFC3856]):

- Co-location: the PUA and the PA is the same entity.
- The PA has access to the registration database and the PUA uses the REGISTER message.
- The PUA uploads presence documents using PUBLISH method, which is the recommended option.

So the basic operation scenario is the following:

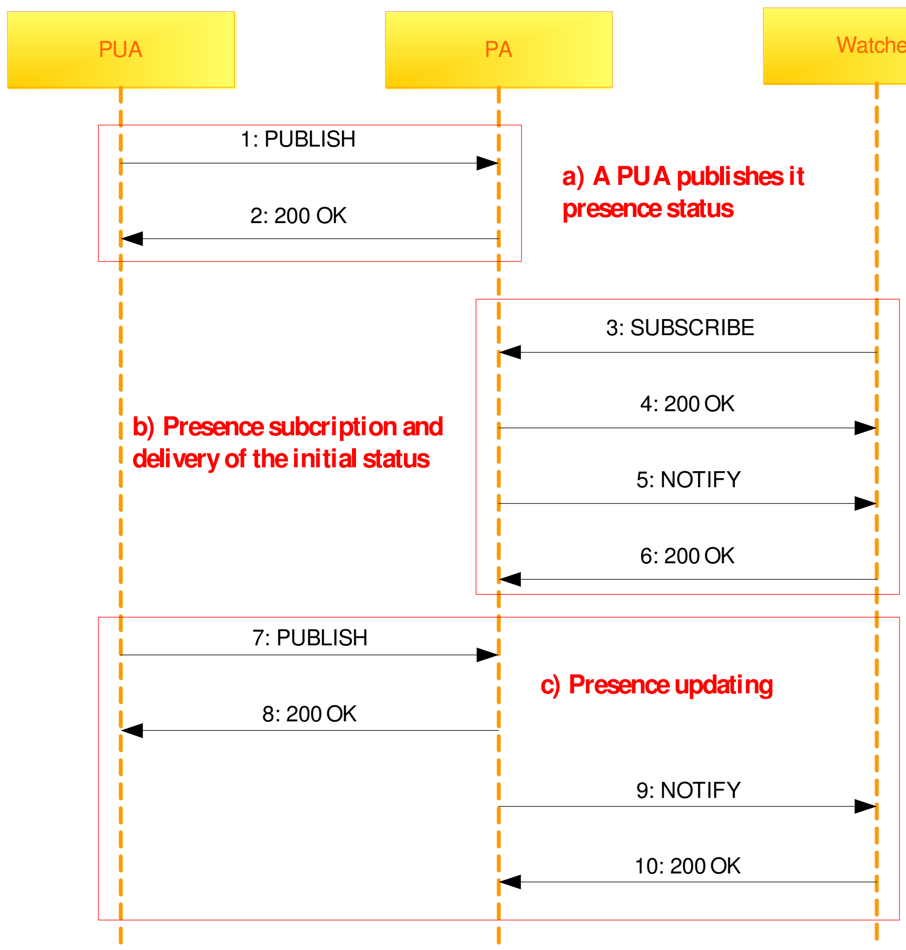


Figure 3 SIP presence basic scenario

Requirements for the Akogrimo architecture:

- Mobile terminals (or end user SIP-aware applications running on terminals) should contain:
    - a presence User Agent to publish presence/context information.
    - a presence watcher to query for and receive presence/context information from other presentities.
- The Presence Agent could be co-located with:
    - SIP Registrar, to be aware of registration status of presentities. In this way the WP4.2 entity in charge of the context management (Context Manager) may implement a presence watcher interface to ask for presence status to the Presence Agent.
    - Context Manager. An additional interface with the SIP registrar is required.

2.2.2.2. *SIP extensions for context management*

The aim of this section is to describe how Session Initiation Protocol can provide enriched presence information, which could be interpreted as context information. Mechanisms described here rely on the following SIP extensions:

- "Session Initiation Protocol (SIP)-Specific Event Notification". [RFC3265] , which describes an extensible framework by which SIP nodes can request and receive notifications from remote nodes indicating that certain events have occurred.
- "Session Initiation Protocol (SIP) Extension for Event State Publication" [RFC3903], which describes an extension to the Session Initiation Protocol (SIP) for publishing event state used within the SIP Events framework.
- "A Presence Event Package for the Session Initiation Protocol (SIP)" [RFC3856], which describes how to use SIP as a presence protocol.
- Presence Information Data Format (PIDF)". http://www.ietf.org/rfc/rfc3863.txt?number=3863
- "A Presence Event Package for the Session Initiation Protocol (SIP)" [RFC3856], which describes how to use SIP as a presence protocol.
- "Presence Information Data Format (PIDF)"[RFC3863], which defines a common presence data format (PIDF).
- The internet draft "A Data Model for Presence" [SIMPLEDM] http://www.ietf.org/internet-drafts/draft-ietf-simple-presence-data-model-02.txt, which defines the underlying presence data model and used by SIP for Instant Messaging Leveraging Presence Extensions (SIMPLE) presence agents. This Internet Draft will expire in August 22, 2005.
- The internet draft "Rich Presence Extensions to the Presence Information Data Format (PIDF)" [RPID] http://www.ietf.org/internet-drafts/draft-ietf-simple-rpid-05.txt, which is an extension that adds optional elements to the Presence Information Data Format. This Internet Draft expired in August 19, 2004; however it is still active.

These extensions and mechanisms allow SIP to provide additional information about characteristics and status for person, services and devices, such as what the person is doing, a grouping identifier for a service, when a service or device was last used, the type of place a person is in (e.g. inside, outside, car, plane, etc.), what media might be private, the relationship of a service to another presentity, the person's mood, the time zone it is located in, the type of service it offers and the overall role of the presentity. Much of this info can be derived automatically, e.g., from calendar files or user activity. As most of the extensions are XML-based, they also can be enhanced to provide other context information if necessary.

2.2.2.2.1. Data formats for enriched presence information

The basic data format for enriched presence was specified in [RFC3863]. Further internet drafts ([SIMPLEDM] and [PIDF]) that still remain active have added new features that enhance the original specification. Details on the data format are described in annex A.1.

PIDF:

A PIDF object is a well formed XML document, containing an encoding declaration (UTF-8) and using the content type application pidf+xml. This information in included in the body field of the PUBLISH SIP message to upload presence information.

PIDF define the format of presence data for a presentity. This format enables the provisioning of segmenting presence information. Protocols or applications may choose to segment the presence information associated with a presentity for any number of reasons, for example, because components of the full presence information for a presentity have come from distinct devices or different applications on the same device, or have been generated at different times.

For more details see annex A.1.

SIMPLE data model for presence:

[RFC3863] does not give guidance on exactly what is the meaning on the described elements in the model, and how to map real world communications systems into the presence document. In this way this document introduces the concepts of service, person and device, related attributes that could be specified and how to map them in a presence document based on the xml-pidf format.

For more information see annex A.2.

RPID:

RPID defines additional presence attributes to describe persons, services and devices data related elements in a more refined way that the SIMPLE data model for presence. An important advantage of RPID is that it is easy to derive info from other information sources, such us calendars, the status of communication devices such as telephones, typing activity, physical presence detectors, etc.

For more details see annex A.3.

2.3. SOAP overview

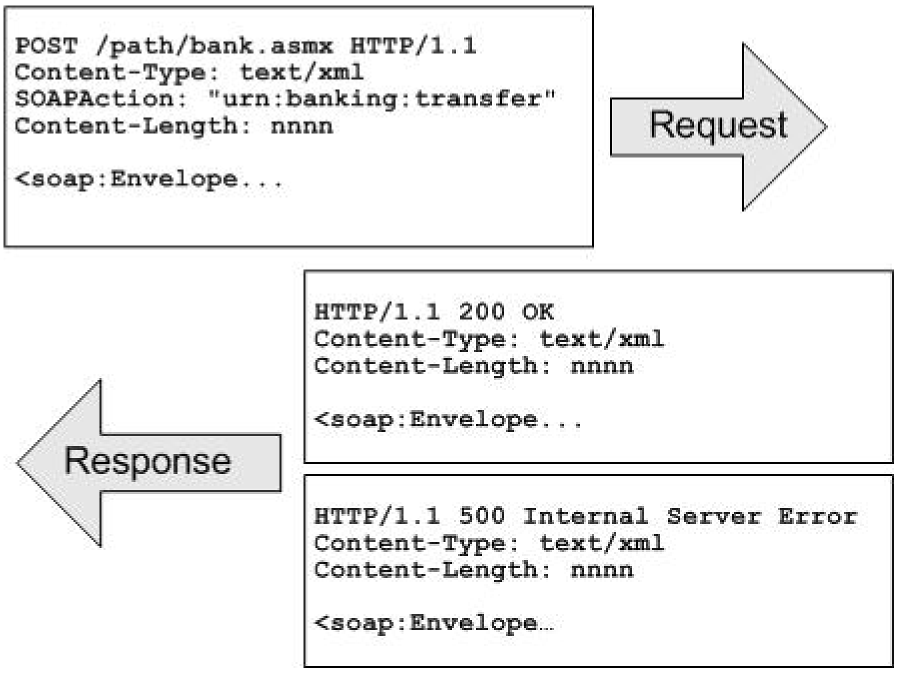
SOAP (formerly an acronym of Simple Object Access Protocol) is a light-weight protocol for exchanging messages. It was first developed to allow mechanism like RPC (CORBA, DCOM) functioning over the internet and independent of the operating system (→ object access). The current version is not limited to that, but the name remains the same. Today's official definition, found in the most recent SOAP 1.2 specification, doesn't even mention objects:

SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. SOAP uses XML technologies to define an extensible messaging framework, which provides a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics.

This definition centralizes most of the important features of SOAP. SOAP defines a way to exchange XML encoded messages between services. It does this by providing an XML-based messaging framework that is easy extensible, independent of the underlying network/ transport protocols and independent of the programming model, language or operating system (or at least should be). The utilization of XML allows easy definition of (new) higher level application protocols.

SOAP is not bound to any transport protocol, there are implementations for the usage of TCP, HTTP, SMTP or even MSMQ (Microsoft Message Queuing), but mainly HTTP utilized. The HTTP protocol (see Figure 4) binding defines the rules for using SOAP over HTTP. SOAP request/response maps naturally to the HTTP request/response model.

The content type header for the HTTP messages must be set to text/xml or for SOAP 1.2 application/soap+xml) as defined in [RFC3902].



```
POST /path/bank.asmx HTTP/1.1
Content-Type: text/xml
SOAPAction: "urn:banking:transfer"
Content-Length: nnnn

<soap:Envelope...
```

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnnn

<soap:Envelope...
```

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml
Content-Length: nnnn

<soap:Envelope…
```

Figure 4: SOAP HTTP Binding

But due to its simplicity and independence of the used transport protocol SOAP lacks support for any security, routing (localization), reliability, etc. It relies on the lower level protocols to provide these features.

Furthermore, SOAP defines a message processing model, RPC encodings and internal fault codes. For the technical description of these features, please see [Sk03].

2.4. Combining SIP and SOAP

In the context of AKOGRIMO, SOAP is used for the transport and control of Grid sessions, as it is used in a traditional Grid scenario, but now actors are mobile and/or nomadic, so the traditional grid management, designed for fixed service actors, is not adequate for this new scenario. Mobile and nomadic actors are the traditional scope of the protocols Mobile-IP (for mobility) and SIP (for nomadic users and session management). So it is needed to combine the traditional Grid management, based on SOAP, and the mobility and session management, based on SIP. To combine SIP and SOAP, there are several possibilities:

- First, the usage of SOAP and SIP in parallel. Grid applications have to be modified in order to use SIP to achieve the signaling requirements mentioned below, and then use SOAP in the same way it is used in traditional, non mobile GRIDS.
- Second, the utilization of SIP as transport protocol for SOAP messages. Applications are not aware of SIP and therefore, do not need to be modified, but the communications infrastructure has to be modified in order to transport SOAP over SIP and a control mechanism has to be included in order to use SIP.
- There is another possibility, which is using SOAP as transport for a signalling protocol which will include the session management and user mobility. This approach could be achieved by two means:
    - Applications implement an internal signalling protocol in order to manage sessions and mobility, and transport this protocol on top of SOAP, like the mobility handling included in OGSA.
    - Applications will use SIP for session and mobility management and transport it on top of SOAP.

The following sections will describe these approaches with their advantages and disadvantages.

2.4.1. SOAP for application, SIP for signalling

In this approach, applications use SIP for signaling, i.e. to manage session and mobility information, locate resources, get additional details about availability/status, etc. Once the session is established and both applications (client and server) have all the data needed to perform the service, they use SOAP to exchange service requests and responses, as they would do in a traditional, fixed GRID.

This approach has been successfully used before on similar scenarios involving signaling (telephony, Voice over IP). However, nomadic GRID services and applications must be changed in order to add SIP-awareness to enable mobility.

Mobile Terminals can be client and/or server (e.g. the doctor in the eHealth scenario, providing his location to the GRID).

The GRID gateway is also mentioned in the general network architecture (e.g. to translate between IPv4 and IPv6).

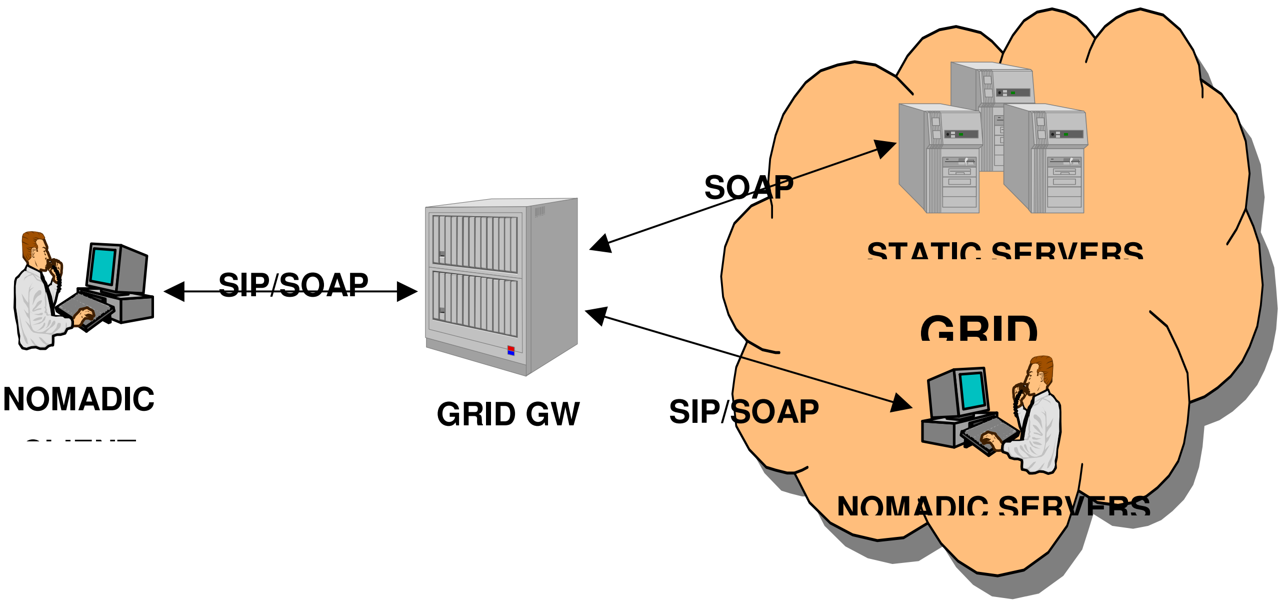


Figure 5: SIP for mobility/session handling, SOAP for application

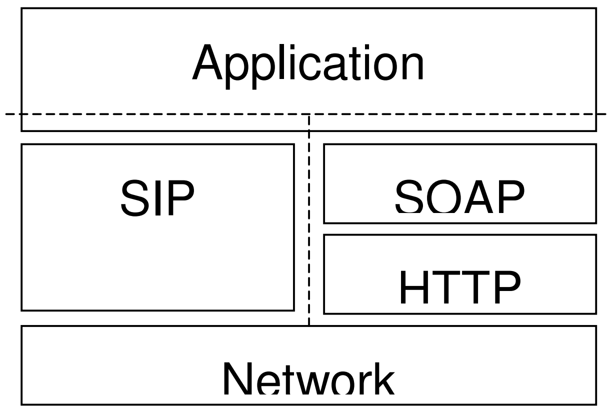


Figure 6: Proposed Protocol Stack

2.4.1.1. *Requirements on SIP for the applications*

In order to analyze the impact of SIP on the applications, it is very useful to characterize the different applications depending on their role in the service. In this way, we have:

- Client applications could be used by any service actor (client or server), and will only emit requests and wait for answers, i.e. they do not need to listen in order to receive requests or asynchronous notifications
- Server applications could also be used by any service actor (client or server) and do need to listen in order to receive requests and/or asynchronous notifications.

Note that a service client could be composed by client applications (e.g. a web browser or a traditional Grid client) as well as server applications (e.g. to be notified of an available resource or to be provided an asynchronous simulation results). Service servers will be composed by server applications and could also include optionally client applications (e.g. a doctor could have a server application in order to be contacted and also client application in order to request some results analysis to the Grid).

With this classification, we can analyse the different demands on SIP from the client part of services/applications and the server part.

Simple client applications will need SIP for a) session management and b) session mobility. For this, the application itself has to be modified in order to:

- understand changes in source and/or destination of a session (i.e. the ability to transfer a session to another node, restoring it afterwards),
- provide further session management like session save/load/restore (may be handled via SIP).
- react to context awareness in the form of terminal capabilities which will be handled via SDP (Session Description Protocol) and SIP.

For mobile and/or nomadic applications that provide some sort of service (i.e. server applications) to the GRID (or other mobile terminals), these requirements have to be extended, so that server applications have to:

- understand changes in source and/or destination of a session (i.e. the ability to transfer a session to another node, restoring it afterwards),
- provide further session management like session save/load/restore (may be handled via SIP),
- react to context awareness in the form of terminal capabilities which will be handled via SDP (Session Description Protocol) and SIP.
- keep their contact/location information updated (in the SIP Registrar), allowing clients to contact them when needed,
- provide additional information, namely presence information (availability, contact means, etc.) to the clients.

Both location and presence can be achieved via SIP. Though, the localization with SIP provides only a virtual location (the current access network of an entity) not a real-world location. Being in the same network does not automatically imply closeness!

Static server applications do not necessarily need to be SIP-aware as long as they don't have to locate mobile nodes.

2.4.2. SIP serving as transport protocol for SOAP

The usage of SIP as transport protocol for SOAP messages combines the advantages of SIP with the flexibility of SOAP. This approach automatically adds session management mechanism, localization, security (privacy) handling to SOAP. Furthermore, GRID gains (implicit) mobility-awareness. This awareness is implicit and without extra changes for the GRID services because it is handled by the SIP infrastructure, namely the SIP registrars and SIP gateways.

SOAP messages are sent directly to a user/service via SIP, the localization is done by the SIP architecture, and the services themselves do not have to know about mobile users or their current location.

In this approach, current GRID services can be used with minimal, or even, without changes. However, deeper changes within GRID toolkits are required to use SIP instead of HTTP to transport SOAP messages.

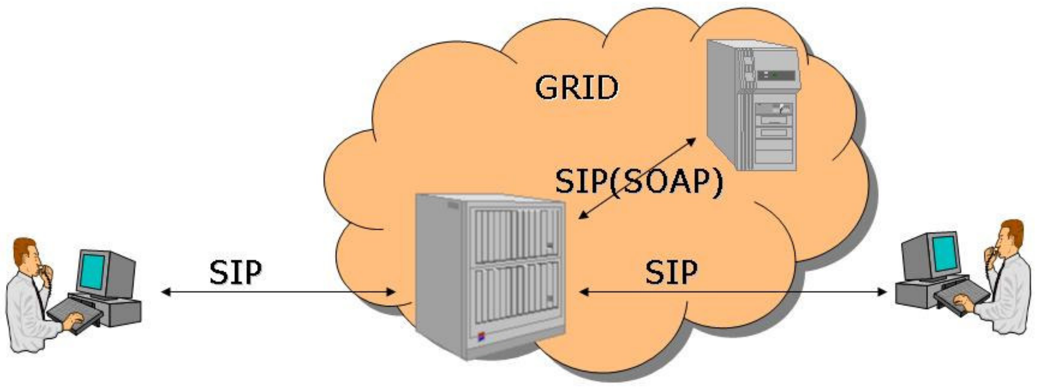


Figure 4: SIP serving as transport protocol for SOAP

2.4.3. Mobility handling with OGSA

Besides the possibility to use SIP for session mobility and localization, there are already mechanisms within OGSA [FoKa, Tu03] to support these features, even if they are currently named and used differently.

The GIS (GRID Information Service) is based on UDDI (Universal Description, Discovery and Information protocol) [UDDI0, UDDI1, WS04]. Its architecture equates more or less the SIP registry architecture.

It can not only be used to describe services (and their technical specifications) but also organizations and persons. Furthermore, it allows changes in location and description of the different entities. This could be used not only to locate services (and mobile services) but also to locate persons.

Additionally, the OGSA requires treatment of migration and relocation of services transparent to the user. The first means the change of location of a service at all; the second means a change of location even during active sessions.

Migration is already handled via GIS and UDDI; this would match user mobility and terminal mobility if these mechanisms would be used for the clients. If relocation is already transparent to the user, this could also be used to provide session mobility for mobile terminals and mobile servers.

2.5. Role of SIP and SOAP in Akogrimo

All approaches have benefits and drawbacks. The first approach is less intrusive with current GRID services architecture and, although the second one adds seamless mobility and session support to GRID services, the changes needed would likely be much more complex, and would lead to greater (if not unacceptable) development efforts in the used GRID toolkits.

Using SIP as transport protocol for SOAP would include too many changes in the framework of the GRID services. Even if this approach provides some advantages in a simple way, these changes may not be manageable during this project.

The protocol stack as shown in Figure 6 is a more sensitive solution. It will affect only the GRID services that have to be SIP-aware in some way. This includes all services that need the localization functionality provided by SIP and that require session mobility.

3. A4C Infrastructure

A platform for mobile grids will deliver its services using Internet technologies. These services require security support in many different aspects, including access control, secure communication, non-repudiation, and auditing. Cryptography provides support for secure communications, while the A4C infrastructure provides access-control and service usage accounting and charging.

Akogrimo's A4C enhances the generic AAA (Authentication, Authorization and Accounting) architecture proposed by the Internet Engineering Task Force (IETF) and the Internet Research Task Force (IRTF) in [LGGVS00] with auditing and charging capabilities.

The A4C system is a central component in Akogrimo and shall perform the following functions:

- Authentication of users, services, and devices.
- Granting access to network and grid services to authenticated users.
- Performing accounting of consumed resources in the mobile grid.
- Assuring non-repudiation of mobile grid services using its Auditing and Non-Repudiation component.
- Computing charges for a user's service consumptions and aggregating all these charges into a single bill.

The A4C system is based on the DIAMETER protocol as specified by the IETF, adapted with mechanisms capable of delivering AAA services to mobile grid services, and enriched with Auditing, Non-Repudiation, and Charging capabilities.

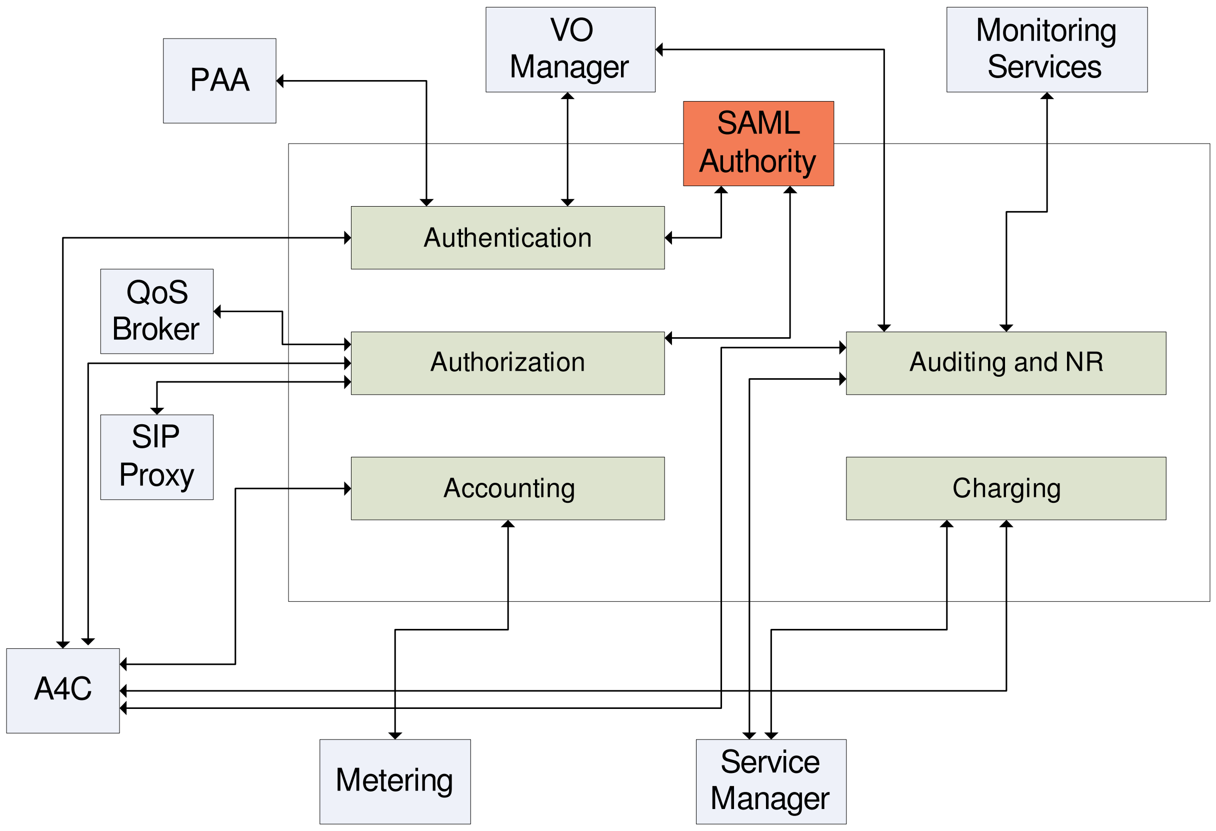


Figure 7: A4C – Functional Architecture

Figure 7 illustrates the functional architecture of the A4C system and its interactions with other Akogrimo entities. The current graph presented here serves as the preliminary and proposed version of the A4C functional architecture. It may be slightly updated if the architecture design in all the four layers of Akogrimo project requests this.

3.1. A4C Tasks

Challenges in designing and implementing the A4C system arise from the need to use mechanisms developed initially for network services and delivering their functionalities to mobile grid services. In the following sections A4C tasks are described and key requirements for all other Akogrimo layers are specified.

The main tasks of the A4C infrastructure include:

Authentication, *Authorization*, *Accounting*, *Auditing*, and *Charging*.

3.1.1. Authentication

The authentication mechanism defines a process for verifying a user's identity. A large variety of mechanisms for verifying a user's identity exists today and most of them can be classified as follows [LGGVS00].

- *Knowledge-based* authentication founds on the knowledge of shared secrets, such as PINs (Personal Identification Number) and passwords.
- *Cryptography-based* authentication includes digital signatures, challenge-response mechanisms, and message authentication codes. The user owns a private key as a characteristic.
- Authentication based on *biometrics* uses inherent information on subjects like fingerprint, voice, and eye characteristic.
- Authentication based on *secure tokens* binds the subject to some kind of ownership, e.g. the ownership of a smart card. It is combined mostly with cryptographic mechanisms to transfer the information on the token to the authenticator.
- *Digitized signatures*, including digital images of handwritten signatures and signature dynamics (i.e., measurements of the direction, pressure, speed, and other attributes of a handwritten signature) are not widely used so far.

An authentication policy describes which authentication mechanism shall be used for identifying the user. In Akogrimo multiple mechanisms shall be allowed, but the authentication itself shall be done by a single unit and that is the one component within the A4C system.

Diameter was designed to enable roaming between administrative domains. Diameter servers can therefore be organized in a hierarchical structure, and the protocol supports different kind of agents to forward messages between client and server. Agents include proxies, redirects and relay agents supporting relaying, proxying and redirecting of Diameter messages through the server hierarchy. Message routing is based also on realms, which identifies the domain of the user. Additionally, Diameter specifies IPSec (Internet Protocol Security) [IPSEC] and TLS (Transport Layer Security) [TLS] for securing communication between Diameter nodes, where IPSec is suggested to be used primarily for intra-domain exchanges and TLS for protection of inter-domain communication. In environments where there are untrusted third party agents end-to-end security is needed.

Figure 8 shows an example in which a Mobile Node (MN) makes an authentication request for accessing Network Provider A (NPA), while he is a client of network provider NPB. The MN uses PANA [FOPTY05] as the transport protocol for authentication information between itself and the AR (Access Router). The AR uses DIAMETER [CLGZA03] for the communication with the A4C server that performs the actual authentication. In this example the AR will contact the A4C server in the user's home network.

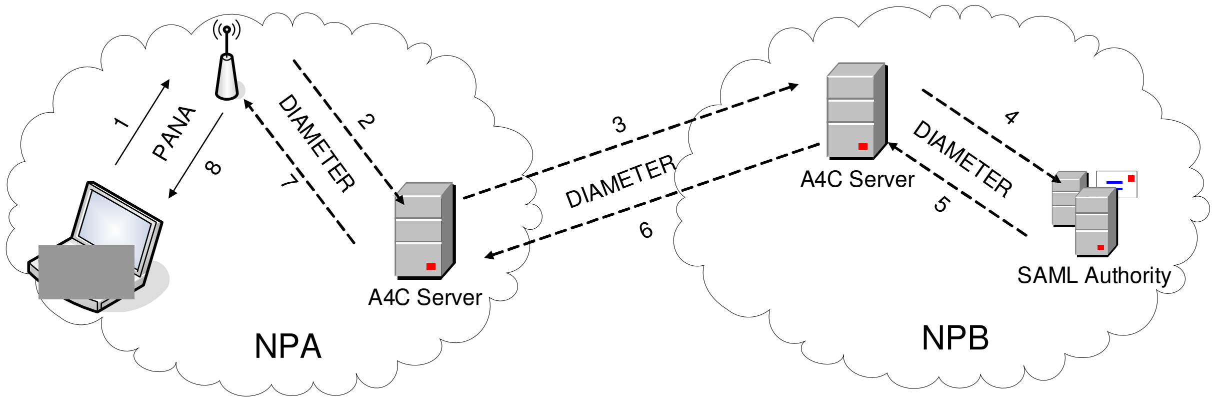


Figure 8: Proposed Authentication and SAML Assertion Creation Scenario

The graphic shown is a network authentication example, but A4C shall support authentication requests also from the other layers. Any component that wants to request authentication services from the A4C needs to implement a Diameter client for the communication with the A4C server.

One challenge for the A4C system is providing support for federated services. SAML (Secure Assertion Mark-up Language [MCKP03]) could be the solution for providing support for Single Sign-On in a mesh of trusted domains. For providing such services, the A4C system shall use the services provided by a SAML Authority. As soon as the user is authenticated, an assertion is made by this SAML Authority and a token is created as a reference to this assertion. The token and the created assertion will be later used for checking the user's identity by any service that has been requested by the user as follows:

- After the user is authenticated the assertion is created and the reference token is sent to the user.
- When entering a new domain, this token will be used for proving user's authentication by his home network (please refer to the authorization example in section 3.1.2 for more information about using the token in authorization of services).

3.1.2. Authorization

Authorization is the process of decision on an entity's allowance to perform a particular action or not. The Authorization decision depends on service specific attributes (e.g. service class for QoS service, device requirements) and user-specific attributes (e.g., name, affiliation to a certain group, age, etc.) There are two classes of mechanisms for performing authentication [LGGVS00]:

- Authentication-based schemes, in which the entity identified during the authentication process is checked against Access Control Lists (ACL) for finding if it is allowed to perform the requested action.
- Credential-based schemes apply credentials, which provide trustworthy information being held by the algorithm performing the authorization

decision. In this case the entity is allowed to perform the action if it holds a certain set of credentials previously obtained.

Akogrimo will use A4C authorization mechanisms for providing access control for network layer services and requesting grid applications. After a successful request of such an application, all the further authorization decisions for performing the grid service are taken by the VO (Virtual Organization) manager using Grid-specific authorization mechanisms. Although the VO Manager performs by itself authorization decisions for accessing grid resources, the A4C system shall be informed about these decisions in order to provide further Auditing services to the upper layers.

Figure 9 shows an authorization request scenario in which a mobile user of NPB is using the access network of NPA and after a successful authentication he requests a grid application (Service X) from Service Provider SP1.

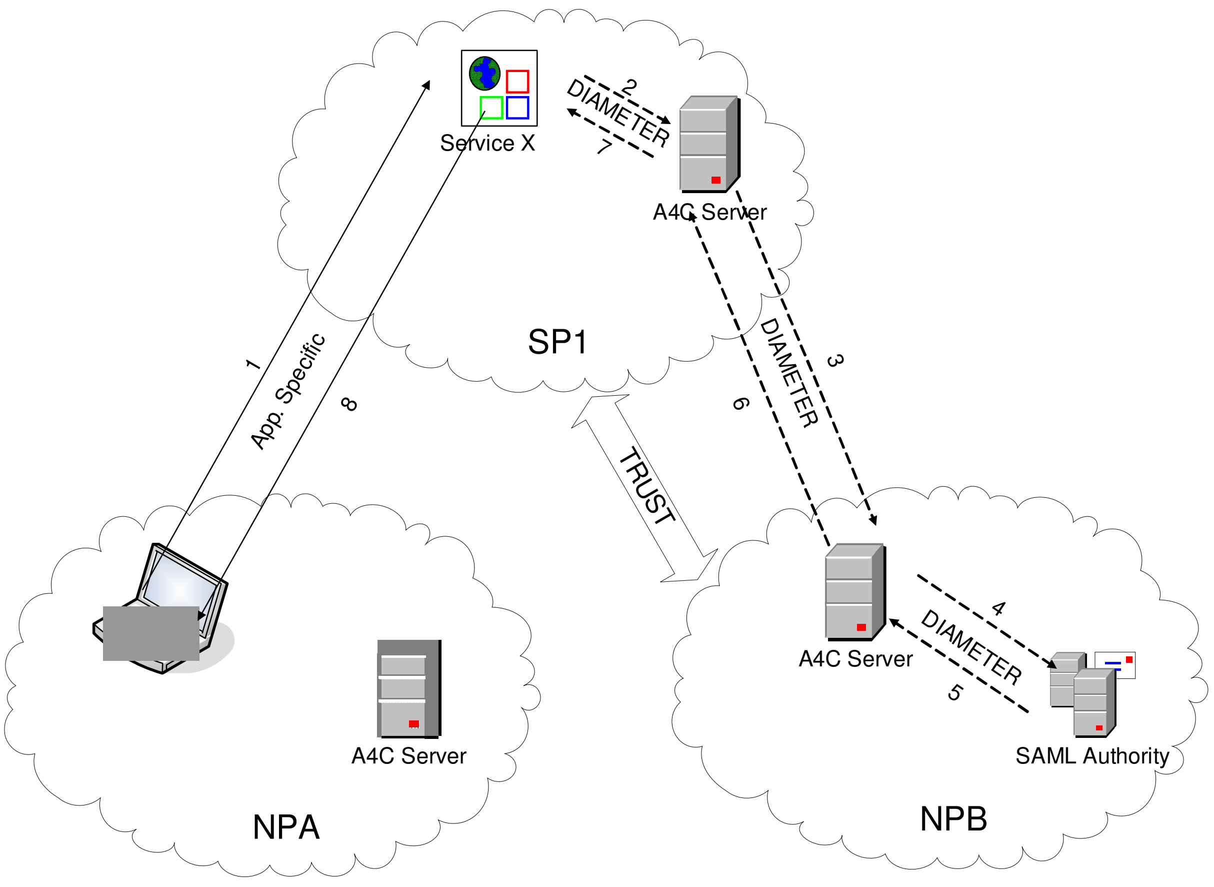


Figure 9: Proposed Authorization Mechanism for a Service Request

During the authentication process, some assertions were made by the SAML Authority in the user's home domain and the user received an artifact referencing those assertions. The following steps are necessary for authorizing the user to start Service X.

- The user tries to access a service grid using an application specific protocol. In the request message he also sends the SAML artifact he received during authentication

(1). Service X will use this artifact to check if user was authenticated by his home network and if so, it might request some of his assertions from the SAML authority in the user's home network.

- The service wants to check if the user was authenticated. For this, it sends the SAML artifact it received to the A4C server in his domain using the DIAMETER protocol (2). The A4C server recognizes that the user is client of NPB and forwards the request to the A4C server of NPB (3).
- The A4C server of provider NPB asks the SAML authority for the user's assertions using the artifact it received from SP1. (4,5)
- The answer is forwarded to the A4C server of SP1 (6) and then finally delivered to the service that requested it. (7)

These previous steps show a credential-based authorization mechanism. From this point of view, the service can use assertions it received from the SAML Authority for deciding on the user's allowance to start the service or not. Besides those SAML assertions, the service might also request for the User's Profile stored in the user's home network (NPB) or in SP1 network. This information might also be used for taking authorization decisions.

3.1.3. Accounting

The accounting system performs two main tasks. The first is collecting data on resource consumption from the services via a metering component while the second task is retrieving stored accounting data whenever this is requested by a legitimate entity. Such requests could be:

- A user wanting to see a detailed record of the services he used.
- The charging component wanting the accounting records for a certain user in order to create a bill.

In a mobile grid resources that have to be accounted for are of much more diversity than the traditional network accounting (such as CPU usage, memory consumption, storage size, accessed content [GGF05]). Every resource in the mobile grid shall be seen as a *service.* Thus, *service consumption* also means *resource consumption.* Considering resources as services also allows for solving the problem of authorizing the access to resources in the same way the service authorization is performed.

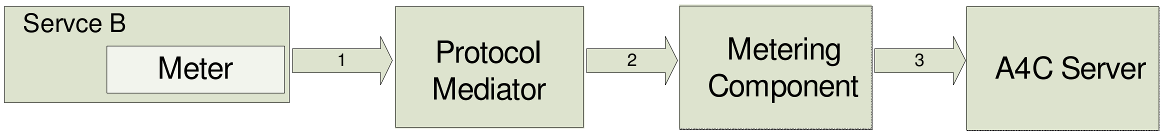


Figure 10: Proposed Architecture for Metering Component

An important component in the accounting architecture is the *metering component (MC)* (Figure 10). Every service that require accounting services need to measure their usage using service-specific meters. (e.g. NetFlow[NETFLW], Netramet [NETMET], SNMP [SNMP]) The MC has the task of collecting the service usage information from a set of meters (1), aggregating this data based on existing accounting policies and sending the result to the Accounting system of the A4C server as accounting records using DIAMETER as the transport protocol (3). Every service shall implement a meter for retrieving data about its usage. As the meter is a service-specific component, it shall use a Protocol Mediator for transforming the data collected into DIAMETER understandable data types. The meter shall provide configuration mechanisms so that accounting policies would be able to steer the whole metering process starting from the service meter. The Usage Data Aggregator shall encapsulate metered data received from the service meters into *Custom Accounting Data Records (CADR)* based on different metering policies received from the A4C server. Custom Data Records are necessary because of the multiple metering schemes possible for different services. Service usage metering and accounting is performed by the Service Provider mainly for later charging purposes, so the charging strategy of the provider influences the metrics he uses for his services. It is easy to consider a scenario in which two service providers exist allowing jobs to be run on their clusters of supercomputers. Based on their different business strategies, the first one wants to charge its users based on the CPU time they used while running jobs on his machines while the second wants to take into consideration also the amount of memory consumption while running the job. They both need to send their metered data to the same Accounting server, but their records will be slightly different:

- SP1: *acct_req(sessid, uid, serv_id, time_start, time_end, cpu_time)*
- SP2: *acct_req(sessid, uid, serv_id, time_start, time_end, cpu_time, DRAM_used)*

In order to support such CADRs the Accounting server needs mechanisms for attaching a *Custom Accounting Data Record Format* (CADRF) for every service for which it accepts accounting records. These CADRFs will be used by the accounting system for checking the correctness of the accounting records received.

3.1.4. Auditing

Auditing defines the process of storage and retrieval, when needed, of information on events taking place in the system, history of the service usage, SLA (Service Level Agreement) compliance, and customer charging and tariff schemes applied. The auditing component shall keep track of all events in a sequential order based on their timestamp. All Auditing records shall provide sufficient information for identifying entities involved in an event, the time the event was produced, and the outcome of that event. Its task is to support creation of trust relationships by verifying that the running processes are reasonable. Auditing mechanisms to be used by the A4C are:

- Logging of requests and their approval/denial
- Logging of session status records
- Trusted 3rd party logging for fairness

- Arbitrary checking that appropriate services are running
- Comparing of log entries from cooperating A4C servers

3.1.5. Charging

Charging is the task of calculating the price for a given service consumption based on accounting information. Charging maps technical values in monetary units and then applies a previously established contractual agreement between service provider and service consumer upon a tariff. One big influence in the final price paid by the user for the service but also influencing the revenue of a service provider is the charging scheme applied for a service. Multiple charging schemes exist in the mobile telecommunication and Internet providing business, but these have to be adapted in the context of Akogrimo. It is hardly likely that a single charging scheme could be used for all mobile grid services, therefore, the charging subsystem needs to be able to use different charging schemes for different services. Each service shall define its own metrics to be accounted and shall be able to define its own charging scheme to be applied on those metrics. However, a service could also use several charging schemes at the same time, for example for users with different subscription types.

A simple charging scheme example is shown in the following lines. Each of the following charging schemes can be used by service provider SP1 (section 3.1.3) for computing prices for service consumptions:

- *cpu_time * cpu_tariff_function = final_price*
- *flat_rate + cpu_time * cpu_tariff_function = final_price*
- *(time_end – time_start) * time_tariff_function = final_price*
- *any other combination of the previous metrics*

A charging scheme like *cpu_time * cpu_tariff_function + DRAM_used * DRAM_tariff_function = final_price* could be very easily be used by SP2, but it cannot be used by SP1, since it will never be able to retrieve the memory usage during its service consumptions based on its accounting records shown in section 3.1.3. The example clearly shows there is a strong, one-way connection between the charging scheme for a certain service and its accounting record format. The service provider has to find the appropriate metrics he needs to account for using the desired charging scheme. The A4C system shall provide the means for specifying the accounting record format a service provider wants to use and the charging schemes and tariff functions applicable for a service.

3.2. A4C Session Model

This section proposes an accounting session model for Akogrimo. It introduces the term of Session ID, and shortly defines the key requirements.

Most of the time the service delivered to the end-user is not a single service, but a composition of several services, sometimes delivered by different service providers and aggregated so that they are seen as one single service by the end-user. Although the requested service can be composed of several other services the user only wants to receive a bill for the aggregated service he requested. This requirement can be satisfied by a mechanism that binds together all A4C sessions involved in the service delivery process.

The concept of a session in the context of A4C is needed for the following:

- Binding together Authentication and Authorization transactions to the main service session.
- Binding together accounting messages to the same session.

Binding together sub-sessions which belong to the same service session or 'super-session".This section proposes an accounting session model for Akogrimo. It introduces the term of Session ID, and shortly defines the key requirements.

Most of the time the service delivered to the end-user is not a single service, but a composition of several services, sometimes delivered by different service providers and aggregated so that they are seen as one single service by the end-user. Although the requested service can be composed of several other services the user only wants to receive a bill for the aggregated service he requested. This requirement can be satisfied by a mechanism that binds together all A4C sessions involved in the service delivery process.

The concept of a session in the context of A4C is needed for the following:

- Binding together Authentication and Authorization transactions to the main service session.
- Binding together accounting messages to the same session.
- Binding together sub-sessions which belong to the same service session or 'super-session".

3.2.1. A4C Session Definition

A session in the context of AAA is defined as service provisioning over time [LGGVS00]. The lifetime of a session is the period of time in which two (or more) parties are in intermittent communication with each other and it has three stages: *Session Establishment*, *Service Session, and Session Termination*.

A *session* is initiated by either party and it is ended in one of the following circumstances:

- One of the parties terminated the session.
- A timeout period has been reached without any communication between the parties.
- A maximum time limit for the session has been reached.
- A maximum communication volume permitted for one session has been reached.

- A controlling/brokering entity decides to terminate the session between two parties.

In the context of A4C a session is defined as the collection of A4C messages exchanged between an A4C server and an A4C client during the delivery of a service to the user. The A4C session is composed by authentication messages, authorization messages, accounting messages, auditing messages, and charging messages. These messages are needed to perform auditing functions later, in case of charging calculation or service infringement.

3.2.2. Session ID Requirements

The following are the main requirements for an A4C session ID that were identified in [Moby02] and [VCZZ01]:

- Session ID must be globally unique.

  To uniquely identify a session, a session ID must be globally unique which means unique in time and location. The might be relaxed to unique within a given time period or within a certain spatial scope. For instance, if we have to audit session data for 10 years the session ID must be unique within this period. If it is assured that a session does not cross certain boundaries the uniqueness requirement may be relaxed to ″unique within these boundaries″.

- It shall support fine granular auditing.

The session ID must allow for the finest grained auditing possible of a provided service.

- It must support flexible accounting.

  The binding model of the session IDs should not limit the accounting and auditing capabilities

- The model should be efficient and scalable.

  The generation and linkage of session IDs must be sufficiently fast compared to the overall process of service authentication and authorization to avoid unnecessary latency.

3.2.3. Generation of a Session ID

An A4C session ID is assigned by the first A4C entity who issues an A4C request for a new service. This entity can be either an A4C Client, either an A4C server. Most of the time the first request will be either an authentication or an authorization request. In order to satisfy the first requirement, global uniqueness, every A4C entity needs to identify its peers by a unique identifier (IP address or any other global unique identifier). Generating an A4C session ID consists of appending a locally unique identifier to the global unique identifier of the A4C entity that generates the respective A4C session ID. This A4C session ID shall be used in all subsequent messages of this A4C session until this session is terminated by the A4C entity.

3.2.4. Binding of Messages

An A4C session needs to associate a service consumer with the service usage. It shall contain information about the consumer of the service (e.g. reference to a user's profile) and information about the service consumption itself: authentication information, authorization information, accounting records. Figure 11 shows an example of an A4C session structure.

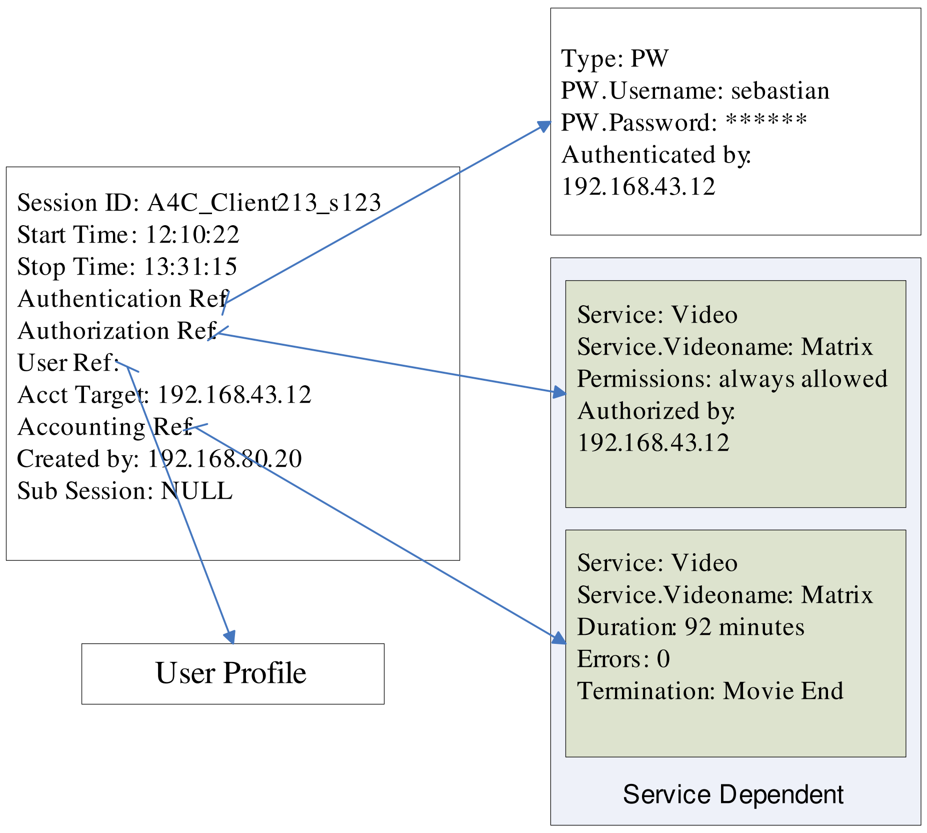


Figure 11: A4C Session Structure Example (Adapred from [VCZZ01])

Two methods are defined in [VCZZ01] for binding the sessions together:

- Hierarchical binding: With a hierarchical binding sub-session ID are derived from the super-session (or parent-session) ID.
- Peer-to-peer binding: With a peer-to-peer binding two sessions at an equal level are concatenated (e.g. an audio-session that consists of two audio-sessions in different provider networks). Information on the binding between two sessions can be stored at the concatenation points where both sessions are known (e.g. A4C server of an aggregator service).

DIAMETER defines a session ID for binding different A4C messages together into one session. The Session-ID is then used in all subsequent messages to identify the user's session. The session state (associated with a Session-ID) is freed upon receipt of the Session-Termination-Request, Session-Termination-Answer and according to rules established in a particular extension/application of DIAMETER. DIAMETER supports binding of several sessions together (e.g. for accounting and charging purposes) by defining a Multi-Session-ID. A service composed by an audio and video stream will generate two different A4C sessions with different A4C session IDs but having the same Multi-Session-ID.

3.3. Use Case Modelling

The following section describes the identified use cases in which the A4C system is involved. The main actors in these use cases are the following:

- PANA Attendant – located usually in the Access Router, it converts authentication information sent using PANA into DIAMETER Protocol.
- Metering components – metering units located in network and grid layer.
- VO manager – makes auditing records, authentication requests.
- Grid Service Manager – defines the accounting formats for a certain service and the charging scheme attached.
- A4C Server – an A4C Server in a foreign domain

The current list of actors and use-cases is not complete, new items shall be added as the design process in all the four layers advances. The first four actors can be grouped into one single class of actors: A4C Client.

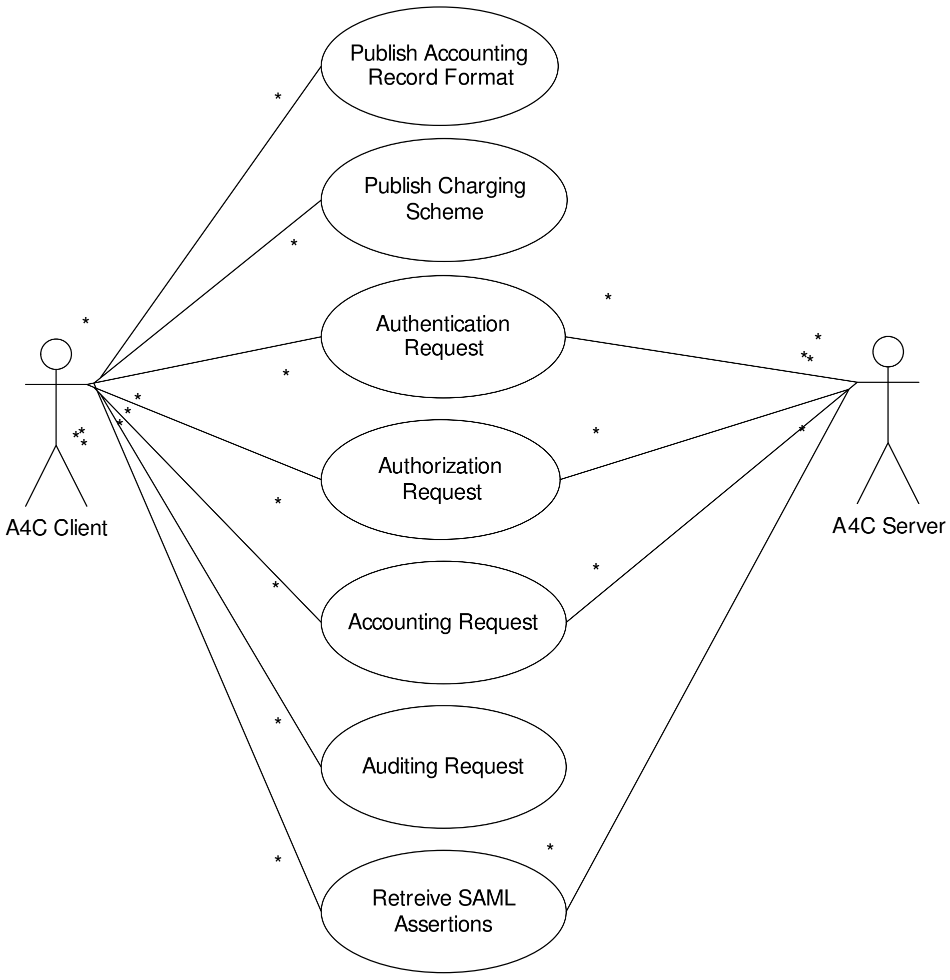


Figure 12: A4C Identified Use Cases

3.3.1. Publish Account Record Format

Use Case ID:	A4C_01		
Use Case Name:	Publish Account Record Format		
Created By:	CM	Last Updated By:	CM
Date Created:	14.03.2005	Date Last Updated:	14.03.2005

Actors:	A4C Client (Grid Service Manager)
Description:	Used by any grid service for publishing its accounting record format
Preconditions:	1. A4C client is authorized to make accounting record format modifications

Postconditions:	1. The new accounting record format has to be “mappable” to the service’s charging scheme
Normal Flow:	1. The A4C Client requests an accounting record format modification for a service 2. Client’s authorization is checked to see if it is authorized to make accounting record format modifications 3. The new accounting record format is registered in the A4C server
Alternative Flows:	
Exceptions:	
Includes:	
Priority:	
Frequency of Use:	Every creation of a new service Every accounting policy change for a service
Business Rules:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

3.3.2. Publish Charging Scheme

Use Case ID:	A4C_02		
Use Case Name:	Publish Charging Scheme		
Created By:	CM	Last Updated By:	CM
Date Created:	14.03.2005	Date Last Updated:	14.03.2005

Actors:	A4C Client (Grid Service Manager)
Description:	Used by any grid service for publishing its charging scheme
Preconditions:	1. A4C client is authorized to make charging scheme modifications
Postconditions:	1. The charging scheme has to be “mappable” to the service’s accounting record format

Normal Flow:	1. The A4C Client requests a charging scheme modification for a service 2. Client's authorization is checked to see if it is authorized to make charging scheme modifications 3. The new charging scheme is registered in the A4C server
Alternative Flows:	
Exceptions:	
Includes:	
Priority:	
Frequency of Use:	Every creation of a new service Every change of the charging policy for a service
Business Rules:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

3.3.3. Authentication Request

Use Case ID:	A4C_03		
Use Case Name:	Authentication Request		
Created By:	CM	Last Updated By:	CM
Date Created:	14.03.2005	Date Last Updated:	14.03.2005

Actors:	A4C Client (PANA Attendant), A4C Server (in foreign network)
Description:	When a user wants to attach to a network provider, he needs to be authenticated. An entity in the access network translates the information from an specific authentication protocol into DIAMETER protocol and forwards this information to the A4C Server for authentication.
Preconditions:	
Postconditions:	
Normal Flow:	1. A4C client sends an authentication request to the A4C server

	2. A4C server checks the credential presented by the user to its user database or it contacts an Identity Provider for further processing this request 3. If the user has the proper credentials, a SAML assertion proving that the user is authenticated is requested 4. The SAML artefact referencing the assertion as well as the authentication response is sent to the A4C client 5. The Auditing Component is informed of the successful authentication
Alternative Flows:	Instead of the A4C client, the request could come from an A4C server in a foreign domain.
Exceptions:	
Includes:	
Priority:	
Frequency of Use:	Every time a user connects to a network.
Business Rules:	
Special Requirements:	

3.3.4. Accounting Request

Use Case ID:	A4C_05		
Use Case Name:	Accounting Request		
Created By:	CM	Last Updated By:	CM
Date Created:	14.03.2005	Date Last Updated:	10.04.2005

Actors:	A4C Client, A4C Server
Description:	In order to perform charging, data about resource usage has to be stored by the A4C server.
Preconditions:	The actor making this request has to be authorized.
Postconditions:	Accounting record is stored in the accounting database.
Normal Flow:	1. The actor makes an accounting request and sends the service ID and the accounting record 2. A4C server checks if the requester if authorized to perform this action 3. The accounting record is checked using the accounting

	record template for that service 4. Accounting record is stored in the database.
Alternative Flows:	
Exceptions:	
Includes:	
Priority:	
Frequency of Use:	Depending on the service, at least twice – when the service begins and when it is finished.
Business Rules:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

3.3.5. Auditing Request

Use Case ID:	A4C_06		
Use Case Name:	Auditing Request		
Created By:	CM	Last Updated By:	CM
Date Created:	14.03.2005	Date Last Updated:	10.04.2005

Actors:	A4C Client, A4C Server
Description:	This interface is used for storing events in the auditing database.
Preconditions:	Requester needs to be authorized.
Postconditions:	An event is stored in the auditing database.
Normal Flow:	
Alternative Flows:	
Exceptions:	
Includes:	
Priority:	
Frequency of Use:	

Business Rules:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

3.3.6. Retrieve SAML Assertions

Use Case ID:	A4C_07		
Use Case Name:	Retrieve SAML Assertions		
Created By:	CM	Last Updated By:	CM
Date Created:	14.03.2005	Date Last Updated:	10.04.2005

Actors:	A4C Client, A4C Server
Description:	An A4C entity might ask for some of the user's assertions for verifying his authentication status or authorization policies for certain services.
Preconditions:	The requester needs to be authorized to request this information.
Postconditions:	The SAML assertions mapped to the artefact in the request are sent to the requesting entity.
Normal Flow:	
Alternative Flows:	
Exceptions:	
Includes:	
Priority:	
Frequency of Use:	Every time a user makes a service request.
Business Rules:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

4. Context

4.1. Introduction

The vision of Akogrimo involves context-aware mobile Grid services. Grid systems have been developed with a focus on technology rather than offering services that are easy to use. Such solutions often call for user attention and require that the users have technical competences. The objective of pervasive computing [Weis92] is to hide technology from the user by seamless integration into everyday life. The focus is on the services that allow the user to solve his tasks while devices and technology fade away into the background. Context-aware systems are a vital part of pervasive computing. Context information gives applications entirely new opportunities to offer new services and adapt their behaviour according to the situation of the mobile user. This increases system usability tremendous by reducing the demands on the end-user and minimizing the need for user attention.

Obviously, it is not desirable that every application implements a context engine to gather and process specific context information; rather a generic context platform/infrastructure that supports applications by gathering, processing and managing context information is needed. Such a generic context infrastructure is an essential component of pervasive computing.

The dictionary defines context as "The interrelated conditions in which something exists or occurs". In the literature many definitions and examples of context information are found. [Dey01] reviewed previous context definitions, and provided the following general definition of context: "Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves." Further, [Dey01] defines a context-aware system as follows: "A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevance depends on the user's task". Hence, any information that can be used to describe the situation of a participant in an interaction is context. Given this wide definition of context, the amount of information that can be included is enormous and it cannot easily be delimited. Further, it is seen that context information cannot be completely specified for all purposes. The context information must be enumerated for a given scenario/application.

[SAW94] states that the important aspects of context are: where the user is, who the user is with, and what resources are nearby. Context information is classified as follows:

- *User context* is information about the situation of a person such as role, identity, location, preferences, social situation, permission profile, activity etc.
- *Environment/Physical context* describes properties of the physical environment, e.g. temperature, light, noise, traffic conditions and so on.

- *Computing context* includes nearby services (printers, displays etc), connectivity, and network services.

Furthermore, one has *time context* [Most03] that relates to the time of a day, week, month or season and *context history* that involves recording of context information over time.

Context attributes can also be classified as internal (logical) and external (physical) [BaDu04]. The external properties can be measured by hardware sensors while the internal context dimension mainly is specified by the user or be captured by monitoring user's interactions such as open web-pages, documents or conversations. Examples of external attributes are location, sound and air pressure while internal properties can be e.g. user's work context or emotional state.

Characteristics of context information are also vital for designing context-aware systems. [HIR02] have classified four characteristics of context information:

- Context information exhibits a range of temporal characteristics: Static context information is any information about the user's environment that is invariant and can be retrieved directly from the user. However, most context information is dynamic and must be updated continuously.
- Context information is imperfect: this involves the validity of context information. The concern is that context information changes very rapidly, and raw context data must be processed before it can facilitate the work carried out by the user, making it difficult to provide correct information to context consumers.
- Context information has many alternative representations: there is a considerable gap between raw context data gathered and processed context information useful to a context-aware system.
- Context information is highly interrelated: the characteristics of derived information (its persistence, quality and so on) are closely linked to the properties of the information it is derived from.

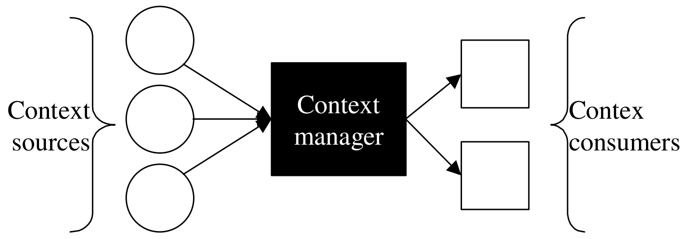


Figure 13, generic flow of context data.

Figure 13 illustrates the general operation of a context management system. Context data is obtained form a number of sources, and distributed to interested parties, i.e. context-aware services. In the trivial case, sources and consumers would interact directly with each other, but as shown in the figure, there may be a mediator infrastructure in the data flow, know as the context manager. The context manager would address issues such as:

- Filtering out relevant data and forwarding those to the interested parties.
- Converting heterogeneous context data to a uniform format.
- Resolving situations where context data are incomplete or inconsistent.
- Inferring higher-level context data from basic data, e.g. by mapping the location of a user onto a map of a building.

Introducing a context manager hence improves the overall scalability of the context system, and also factors out functionality that would otherwise have to be duplicated in each context-aware service.

4.2. Akogrimo context management

From the above discussion it is seen that context information cannot be completely specified for all purposes. The context information must be enumerated for a given scenario/application. In the following, the focus is on context requirements derived from the Akogrimo testbed scenarios.

The initial work presents a simple conceptual information model aiming to capture fundamental context information needed by a majority of Akogrimo context consumers. It is not realistic that a context infrastructure can provide every representation of context information needed for any domain specific purpose. The approach is therefore that the context system gathers basic context data, processes and refines it according to the Akogrimo basic context requirements and allows context consumer to add extension modules to handle domain specific context inference. In the Akogrimo architecture, the component that handles context is denoted Context manager.

The functional description is based on use-case models and Message Sequence Charts (MSCs) developed in WP 4.2 and WP 3.1, respectively. The context manager gathers raw context data from different context data sources (users, devices/device managers, sensors, etc) across well-defined interfaces. Given the variety of possible context data sources, the context manager needs to handle multiple protocols for collecting data from different sources. Context data sources either push context data to the context manager (at regular intervals or when particular events occur), or the context manager polls context data sources (e.g. if context is not provided from sources as expected or if context is wanted from sources that normally do not provide it).

The context manager has a set of rules (policies) describing the logic for context gathering (e.g. which sources provide context, at what schedule, action to take when something fails etc.). These rules are configured through an administrative interface.

The context manager refines raw context data to provide standard formats (e.g. geographical coordinates, temperatures, etc.), performs processing according to Akogrimo basic requirements and stores the context data in a context database.

Context consumers can either execute queries towards the context manager to get specific context information or subscribe for notifications about changes in certain context data (using a suitable semantic language, e.g. RDF, OWL). The context manager is responsible for distributing context data to context consumers accordingly.

To handle domain specific context inference, the context manager offers interested parties interfacing with extension modules. One example is the mapping of location coordinates to a map of a building, such that the context will also specify which room a person is in, and what physical facilities. Context extension modules also allow context consumers to combine basic context information with additional domain specific context sources, e.g. combining user location with data from specific medical sensors. The context manager must interact with A4C to check authorization of a given request (or use cached information).

As shown in Figure 14, the context approach in Akogrimo is based on a layered architecture.

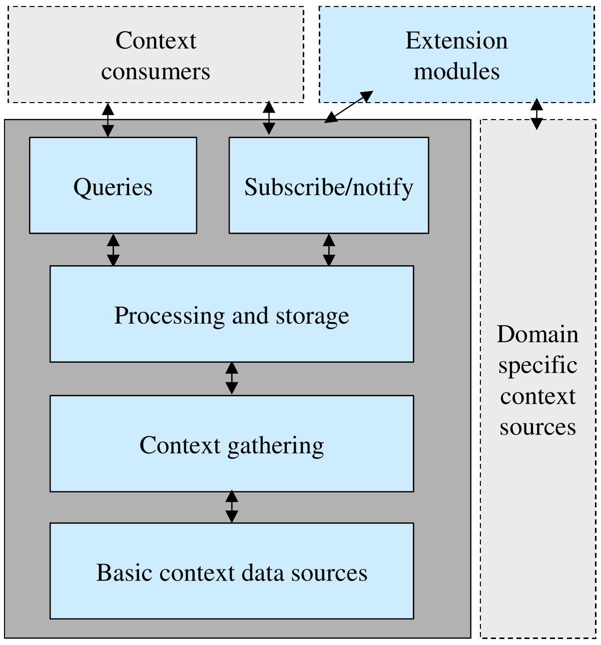


Figure 14 Context management layers

In the following, some issues that are of particular importance for the design of context management are discussed.

The separation of basic context gathering and processing from the applications has several advantages such as:

- Hiding low level-details from applications
- Accessing standardised APIs facilitates the usage of context by applications

- Applications can add domain specific context handling as needed
- Context data sources are reused across applications
- End-systems (e.g. mobile terminals) do not have to perform context processing

Performance and scalability are two major concerns for the design of a context management infrastructure. The factors contributing to the problem is the highly dynamic nature of context information, large number of mobile users, the huge amount of context data collected and processed from a variety of sources, and near real-time requirements for delivery of context info to consumers. Dynamic context information is useless for context consumers if it is out-of-date or incorrect. Some important design parameters for a scalable context infrastructure are the following:

- Context infrastructure architecture
  - Centralised
  - Distributed
  - Hierarchy
- Gathering of context data
  - How is data gathered?
    - Pushed by context data sources
    - Pulled by context manager
  - When is context data collected?
    - Continuously, at regular intervals, when context changes occur etc.
  - Who performs context gathering?
    - Context client on mobile terminals gathers as much context info as possible, and reports it to context manager. Context manager gathers additional info from other sources that are not available to context client.
    - Mainly centralised gathering of context data. Context client gathers and reports some presence parameters.
- Processing and distribution of context information
  - Context manager must provide basic context information in near real-time.
  - Context queries
    - Reactive processing
    - Pro-active processing
  - Subscription to context changes – proactive processing
    - Flexibility must be limited to handle processing

Another critical issue for context-aware systems is user security and privacy. The user must have control over which context attributes are exposed to whom. Only authorized actors are allowed to access context data. For some applications, such as traffic planning anonymous context data (e.g. location data) can be made available. It is possible that some applications will be allowed to override the user's privacy settings, such as disaster and crisis management applications.

4.3. Model of basic context information

As discussed in Section 4.1, the amount of context information available is enormous and a context information model cannot be completely specified for all purposes. The context information model presented is therefore targeted to the requirements seen from the Akogrimo testbed descriptions. The approach is to define a context information model that captures information common among multiple Akogrimo applications. This information model must be flexible such that it can be extended according to domain specific requirements handled by extension modules. The Akogrimo information model focuses on the situation of the user. The main entities that are included in the initial information model are user, device and local services, as shown in Figure 15.

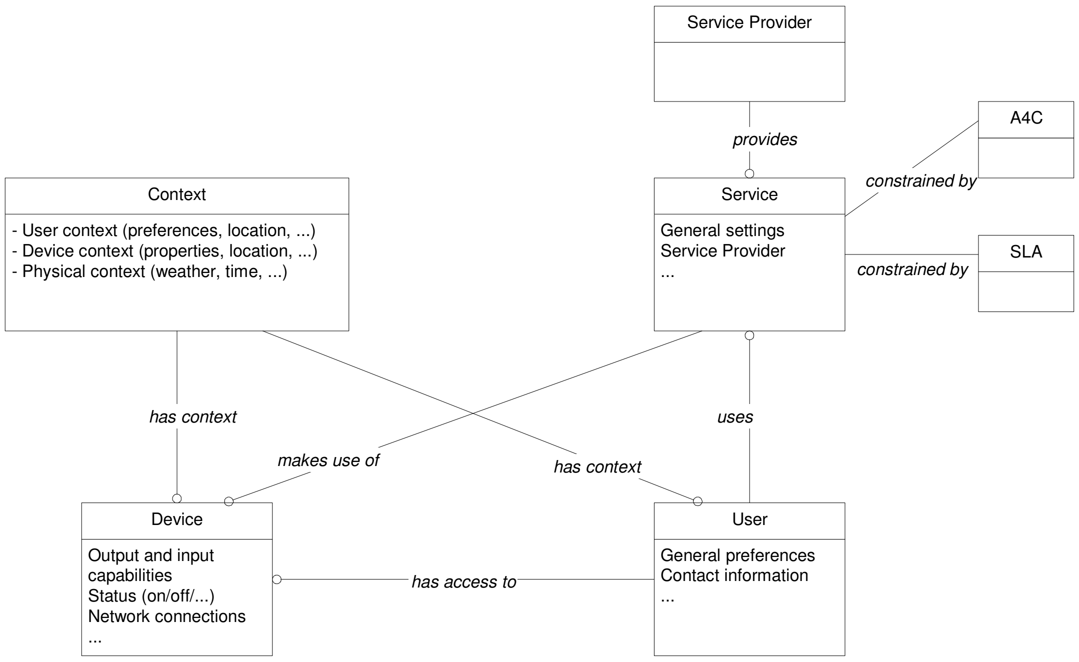


Figure 15 Information model with focus on context.

Generally, the end user must be allowed to control access to all context information. However the user is not allowed to lie about context information (e.g. location) collected automatically. The user should be able to set the role he or she is in and give rights of using/viewing context information to other based on the given role.

In the following, relevant context attributes for the different entities as identified from Akogrimo testbeds are discussed. Furthermore, relevant standards are briefly discussed.

4.3.1. User context

From the scenarios in D2.3.1 [D2.3.1] the following User context information is needed to provide general services:

User Profile/information

The user information contains more or less static information about the user. A generic user profile consists of contact information (such as name, address, e-mail and phone numbers), demographics (e.g. age, sex, language etc.) and application data (application preferences and configurations). Some information is static (e.g. age) while some parts will change infrequently (e.g. change of address, phone numbers etc). The information will be registered when the user starts using the Akogrimo system.

User role

A user has certain roles that he/she plays in different social settings (e.g. work, private). Examples of roles are medical doctor, nurse, paramedic, patient, teacher, student, customer, mother and father. The role a user has will influence the permissions a user has, privileges, status, what context information is available to whom etc. It is necessary to determine the roles available to a particular user, and who are permitted to activate a certain role. Some roles will be open for everybody (e.g. customer) while only certified users can take other roles (e.g. medical doctor). The user itself should be allowed to activate available roles. Additionally, an administrator could be allowed to activate certain roles. The user role is dynamic of nature, however changes will usually occur infrequently.

Presence

Presence information is the basis of instant messaging (IM) and refers to information about the state of users such as availability, reachability and other information set by the user (e.g. mood, interests, etc.). This allows users to detect whether their friends/colleagues are online and if it is possible to communicate with them. In addition information about planned future availability of the user given in a calendar can be included. Presence information is dynamic and may change frequently. The changes can occur manually by user interaction or automatically based on available context information (e.g. location, user talk on the phone etc.).

Currently, there are multiple protocols for handling presence information such as:

- Parlay PAM
- Parlay X Presence
- OMA Wireless Village
- IETF Working groups:
    - IMPP (Instant Messaging and Presence Protocol)
    - SIMPLE (SIP for Instant Messaging and Presence Leveraging Extensions)

- APEX (Application Exchange)
- PRIM (Presence and Instant Messaging Protocol)

For Akogrimo, SIMPLE [SIMPLE] is chosen for supporting Presence, this due to the use of SIP as a main technology in the project. SIMPLE does not support planned availability.

Location

Location systems deliver geographic coordinates of people (things and devices). The geographical coordinates can be used to infer details about where the user is, e.g. room, street, at home, in the car. There exists a multitude of technologies for locating and tracking individuals (based on GPS, WLAN, GSM, RFID, active badges etc) that are becoming increasingly more common. How the location information is gathered and the quality (accuracy, up to date etc) depends ultimately on the location technology being used. A location system will probably have to combine different location technologies to be able to track users both indoor and out door.

The Akogrimo approach is to include basic location information (e.g. coordinates), allowing domain specific applications to infer additional information from it.

There is no common protocol for exchange of location information, examples of available protocols are:

- Parlay (X) Location
- OMA Mobile Location Protocol [MLP]
- Ericsson proprietary Mobile Positioning Protocol [MPP]

Additionally, domain specific applications may require additional user context information.

4.3.2. Device context

From the scenarios in D2.3.1 [D2.3.1] the following Device context information is needed to provide general services.

- Screen Size ++ (pixels, color depth etc.)
- Operating System (version)
- Memory
- Network connections
- Browser
- Installed SW and players

The two main contenders for describing device context are UPnP Device descriptions and UAProf.

From [UPnP] the UPnP description contains the following:

“The UPnP description for a device is expressed in XML and includes vendor-specific, manufacturer information like the model name and number, serial number, manufacturer name, URLs to vendor-specific web sites, etc. The description also includes a list of any embedded devices or services, as well as URLs for control, eventing, and presentation.”

The UAProf work group has defined an extensible framework for describing and transporting information about a device. From [UAProf] the schema for WAP (Wireless Access Protocol) User Agent Profile (UAProf) contains the following:

“HardwarePlatform: A collection of properties that adequately describe the hardware characteristics of the terminal device. This includes, the type of device, model number, display size, input and output methods, etc.

SoftwarePlatform: A collection of attributes associated with the operating environment of the device. Attributes provide information on the operating system software, video and audio encoders supported by the device, and user’s preference on language.

BrowserUA: A set of attributes to describe the HTML browser application

NetworkCharacteristics: Information about the network-related infrastructure and environment such as bearer information. These attributes can influence the resulting content, due to the variation in capabilities and characteristics of various network infrastructures in terms of bandwidth and device accessibility.

WapCharacteristics: A set of attributes pertaining to WAP capabilities supported on the device. This includes details on the capabilities and characteristics related to the WML Browser, WTA etc.

PushCharacteristics: A set of attributes pertaining to Push specific capabilities supported by the device. This includes details on supported MIME-types, the maximum size of a push-message shipped to the device, the number of possibly buffered push-messages on the device, etc.

Additional components can be added to the schema to describe capabilities pertaining to other user agents such as an Email application or hardware extensions.”

Since the UPnP descriptions are more limited then UAProf and only UAProf covers the elements needed in the scenario descriptions [D2.3.1] the latter should be chosen for describing Device context (profile).

4.3.3. Local services

The local service discovery agent on mobile terminals will use protocols for local service discovery to find nearby services. Different protocols for local service discovery (e.g. SLP, bluetooth etc.) have information models that can be used. Alternatively, the local discovery agent or the context manager can translate information about service to a more generic service model. How this will be dealt with is currently an open issue.

4.4. Use Case modeling

In this section we describe Use Cases relevant for context handling. Relevant Actors are listed in Table 1, while Figure 16 gives an overview of relations between Use Cases and Actors.

Table 1 Use Case List

Primary Actor	**Description**
Context Manager	Logical unit provided by WP 4.2
User	The physical person for whom context exists
Local service discovery	Logical unit for discovering devices and services local to the user
Call handler	This is basically SIP for call handling
Service logic	WO and/or other logical units in WP4.3 and 4.4
Service support (A4C, SLA, …)	For issues related to A4C, SLA, QoS, …

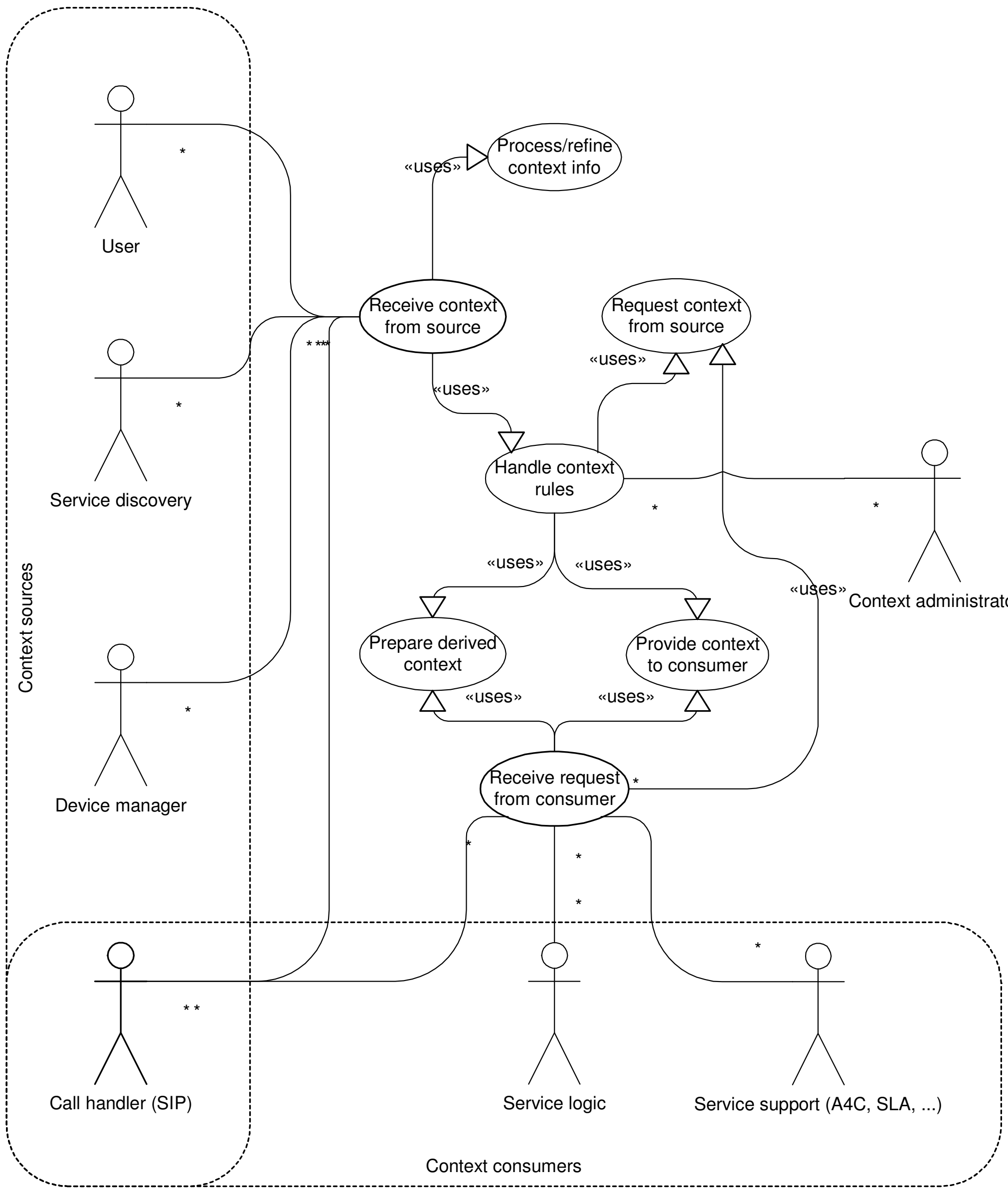


Figure 16 Actors and Use Cases relevant for context

4.4.1. Receive context from source

Use Case ID:			
Use Case Name:	Receive context from source		
Created By:	PO	Last Updated By:	PO
Date Created:	17.02.2005	Date Last Updated:	09.04.2005

Actors:	User, Local service discovery, Call handler

Description:	Receives context from sources (users, local service discovery, sensors, etc) through well-defined APIs. Any kind of context information may be received.
Preconditions:	1. Sources agree to send context, i.e. they are aware that context must be sent regularly or when particular events occur 2. Sources have been requested to send context 3. Sources may be identified as legal for context provision
Postconditions:	
Normal Flow:	1. An external source initiates a session for context transfer 2. The Use Case verifies A4C and the transferred context 3. Context is stored as persistent data 4. Other Use Cases are invoked (see Include list below)
Alternative Flows:	
Exceptions:	
Includes:	- Process/refine context info - Handle context rules
Priority:	
Frequency of Use:	
Business Rules:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.4.2. Request context from source

Use Case ID:			
Use Case Name:	Request context from source		
Created By:	PO	Last Updated By:	PO
Date Created:	17.02.2005	Date Last Updated:	09.04.2005

Actors:	-
Description:	This Use Case is used in the following situations: - If context is not provided from sources as expected - If context is wanted from sources that normally do not provide it

Preconditions:	1. The identification of the source is known 2. There is a need for context from this source
Postconditions:	
Normal Flow:	1. Request is sent to the relevant source through a well-defined API 2. Verify that context was actually received 3. If context was not received, retry or give up (depending on rules)
Alternative Flows:	
Exceptions:	
Includes:	
Priority:	
Frequency of Use:	
Business Rules:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.4.3. Process/refine context info

Use Case ID:			
Use Case Name:	Process/refine context info		
Created By:	PO	Last Updated By:	PO
Date Created:	17.02.2005	Date Last Updated:	09.04.2005

Actors:	-
Description:	Process or refine info to provide context on standard formats. Examples: Geographical coordinates converted to latitudes and longitudes, miles and yards converted to meters and kilometres, Fahrenheit converted to Celsius, etc.
Preconditions:	1. Set of rules for conversion
Postconditions:	
Normal Flow:	1. Context received 2. Check rules, refine context if necessary

Alternative Flows:	
Exceptions:	
Includes:	
Priority:	
Frequency of Use:	
Business Rules:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.4.4. Handle context rules

Use Case ID:			
Use Case Name:	Handle context rules		
Created By:	PO	Last Updated By:	PO
Date Created:	17.02.2005	Date Last Updated:	09.04.2005

Actors:	Context administrator
Description:	This Use Case hosts logic for context handling. This is based on a set of rules that specify - the sources (who should provide context, and at what schedule) - the consumers (who subscribe to context, and at what schedule) - need for derived context (who needs, what to derive, etc) - actions to take when something fails. Rules are being created and modified by the Context administrator
Preconditions:	1. Context rules have been uploaded from the Context administrator
Postconditions:	
Normal Flow:	1. Verify rules 2. Perform actions according to rules 3. If necessary, engage "Request context"-Use Case 4. Upon notification from other Use Cases, check if actions

	are necessary. This could be to provide context to consumer when context has been updated 5. Check if consumers require preparation of derived context. If so, engage "prepare derived context" use case 6. Engage "Provide context"-Use Case to send context
Alternative Flows:	
Exceptions:	Notify Context administrator if problems persist
Includes:	Request context from source Prepare derived context Provide context to consumer
Priority:	
Frequency of Use:	
Business Rules:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.4.5. Receive request from consumer

Use Case ID:			
Use Case Name:	Receive request from consumer		
Created By:	PO	Last Updated By:	PO
Date Created:	17.02.2005	Date Last Updated:	09.04.2005

Actors:	Call handler, Service Logic, Service Support
Description:	Used by consumers to specifically request context through a well-defined API
Preconditions:	
Postconditions:	
Normal Flow:	1. Receive external request 2. The Use Case verifies A4C and the semantics of the request 3. Context is retrieved - from persistent data, or - requested from sources if necessary (by using the

	"Request context"-Use Case) 4. Engage the "Prepare derived context"-Use Case if necessary 5. Resolve conflicts concerning inconsistent context data. 6. Engage the "Provide context"-Use Case to reply to the request (reply with an error message in case of errors)
Alternative Flows:	
Exceptions:	
Includes:	Request context from source Prepare derived context Provide context to consumer
Priority:	
Frequency of Use:	
Business Rules:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.4.6. Prepare derived context

Use Case ID:			
Use Case Name:	Prepare derived context		
Created By:	PO	Last Updated By:	PO
Date Created:	17.02.2005	Date Last Updated:	09.04.2005

Actors:	
Description:	In some cases the consumer might want to subscribe to derived or combined content. This could for instance be a combination of users and devices, e.g. all users within rang of a particular camera, or all cameras within 50 meters range of a particular user.
Preconditions:	1. Definitions/rules for derived context
Postconditions:	
Normal Flow:	1. Receive instructions from other Use Case 2. Prepare requested derived context

	3. Return
Alternative Flows:	
Exceptions:	
Includes:	
Priority:	
Frequency of Use:	
Business Rules:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.4.7. Provide context to consumer

Use Case ID:			
Use Case Name:	Provide context to consumer		
Created By:	PO	Last Updated By:	PO
Date Created:	17.02.2005	Date Last Updated:	09.04.2005

Actors:	
Description:	Responsible for sending requested context to consumer
Preconditions:	1. Requested context exists
Postconditions:	
Normal Flow:	1. Receive request from other Use Case to provide context 2. Retrieve requested context 3. Send context to user. Resend or eventually resign if problems occur 4. Notify requesting Use Case (step 1) of the outcome of send operation
Alternative Flows:	
Exceptions:	
Includes:	

Priority:	
Frequency of Use:	
Business Rules:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.5. Subsystem interfaces

4.5.1. Context manager – data sources

The context manager and different data sources must use well-define API(s) to interact. The context manager will gather context data from multiple heterogenous data sources. There will therefore be a need for several well-defined interfaces to communicate with different context sources.

- Sub-interface1: Context manager gathers user context data. Part of the information is collected by SIP SIMPLE. This interface may be extended to get user context from other sources that SIMPLE.
  Used protocol: SIP (SIMPLE) (+ additional protocol if needed - proprietary – developed in Akogrimo)
- Sub-interface 2 (CM – local service discovery agent): Discovery of local service in the nearby area of a user (not web/grid services). The "local service discovery agent" running on the mobile terminal interacts with local service discovery and report available services to context manager.
- Sub-interface 3 (CM – location service):
  Used protocol: the protocol will dependent on location technology (for RFID a proprietary protocol based on web service will be developed).

4.5.2. Context manager – context consumers

External context consumers need interface(s) to interact with the context manager. There are at least three distinct types of interactions that are allowed across this interface:

- Context queries: this interface allows queries for specified context data
  Used protocol: Semantic language (e.g. RDF, OWL) over web services
- Context subscription: this interface allows context consumer to subscribe to notifications about changes in context info
  Used protocol: Semantic language, e.g. RDF, OWL over WS-Basenotification

- Context extension: this interface allows context consumers to specify domain specific inferred context
  Used protocol: Semantic language (e.g. RDF, OWL) over web services

4.6. Practical realization

This section will suggest how the context manager could be implemented, and also discuss the simplifications that must be made for the first phase (M0-M18) of Akogrimo.

4.6.1. Collecting context data

Figure 17 shows how context data are collected. The Mobile terminal contains a SIP Presence User Agent (PUA), which sends SIP presence data (PUBLISH requests) to the context manager. The context manager in this case acts as a SIP Presence Agent (PA)[1].

A local service discovery agent in the terminal discovers devices in the vicinity using suitable protocols (Bluetooth, SLP,…). The result of this discovery, along with the capabilities of the mobile terminal itself, is then reported to the local service discovery server (which is part of the context manager). The local service discovery server will be implemented as a WEB service, accepting CC/PP documents. The context manager will register with one or more location services, to track the location of people. Location services could be based on many different location technologies. In the example here, the location service uses RFID readers to determine the location of users (which must carry RFID badges). This is a proprietary location system where RFID readers send information to a location server every time a RFID tag is read. The two main advantages of this approach are:

- Easy to prototype/demonstrate services when location changes (One can set up different RFID readers in a limited space and program the reader software to give any wanted location)
- The implementation resources needed for this solution are limited.

[1] The context manager could act as a SIP PA or receive presence updates from a separate SIP PA. This is an implementation issue and will depend on what SIP protocol stack etc. that will be used in Akogrimo.

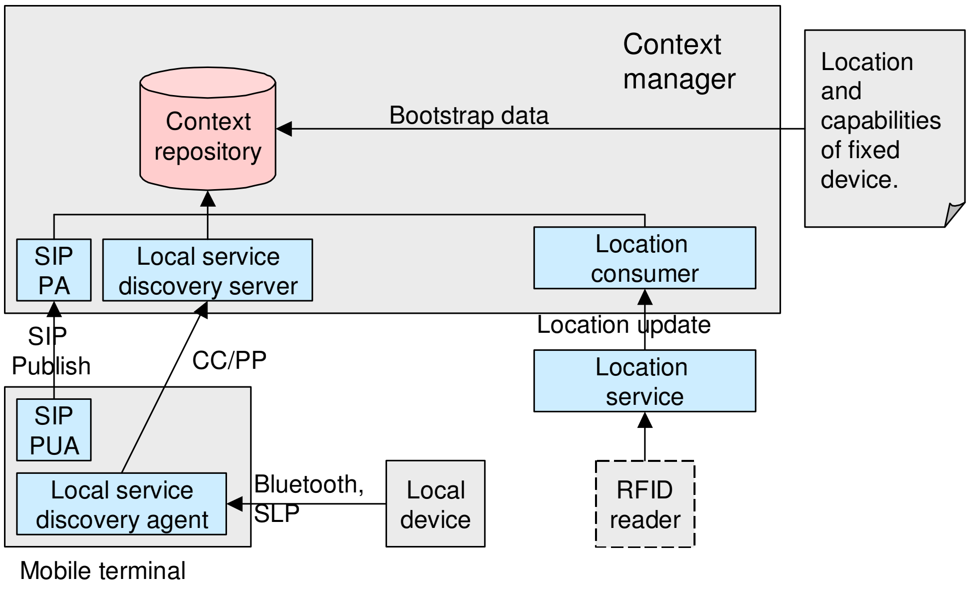


Figure 17, collecting context data.

The context manager will also be bootstrapped with a set of static data, describing location and capabilities of devices not supporting service discovery protocols and/or lacking a mechanism for determining their location.

4.6.2. Using and inferring context data

Figure 18 shows how context information is used by interested parties, and how domain specific context can be inferred. A context consumer may poll for context data using a WEB-service accepting queries in a semantic language, as shown on the left-hand side of the figure. Alternatively, the consumer may subscribe to a given type of context data, and receive notifications when relevant data changes. In this case, WS-baseNotification will be used for the publish/subscribe mechanism, while queries and results will be expressed in a semantic language.

Context inference also utilizes the subscription mechanism. A context inference module will then infer further domain specific context, and submit this to the context manager, through a WEB-service accepting semantic language documents.

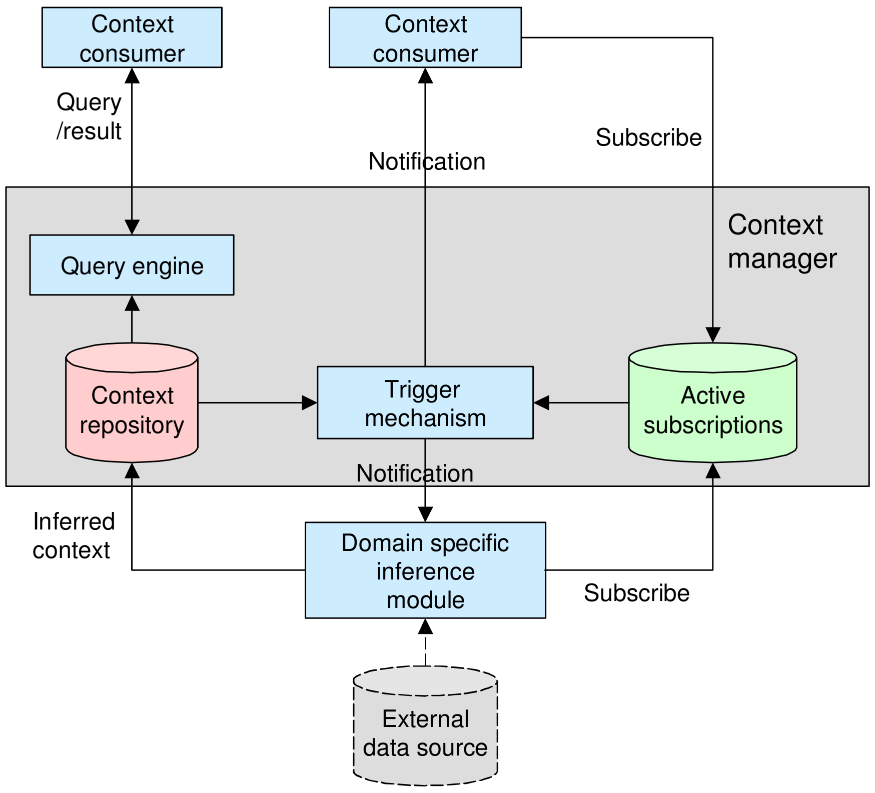


Figure 18, using and inferring context data.

4.6.3. Implementation considerations for phase 1

Limited time and resources are available for phase 1, so the subset of the context manager to be implemented in that phase must be kept simple. The main priority should be to demonstrate the basic functionality of the context manager. Non-functional properties such as scalability, performance, reliability and security will get secondary priority. The following simplifications should be applied:

- A static data model for context data should be used. The context data model can then be hard coded as e.g. SQL tables or Java data types. This also means that context inference is limited to a set of predefined ontologies.
- There should be a fixed set of context data queries, where only attribute values may be changed. This eliminates the need for a true semantic language query engine, and query execution can be hard coded using e.g. SQL or Java.
- Only bluetooth will be used for discovering local devices.
- RFID is the only location service that will be supported.
- A simple callback mechanism may be used instead of WS-baseNotification, depending on availability, maturity and complexity of WS-baseNotification implementations.

5. Service Description and Discovery

5.1. Introduction

The design of the service discovery (SD) infrastructure implies the definition and specifications of certain elements such as directory registers, service description, registry procedures, service retrieval or service invocation. All these elements are inter-related in order to build an acceptable infrastructure where services information is easy to access and easy to retrieve with an efficient management and proper maintenance. Allowing thus, adequate matches between the needs of the user and what the correspondent service.

Directory registries are repositories for information about services. Directory services provide a consistent way to name, describe, locate, access, manage and secure information about resources making relevant information accessible. The architecture of directory registers may be centralized of different kinds such as distributed, hierarchical, P2P structured or Ad hoc. Centralized directory registers imply to have all the information stored in one or several places (by means of using replication). This approach has some drawbacks like poor scalability or as less consistency on large registries. On the other hand, a distributed architecture would eliminate these problems. However, it requires an efficient synchronization procedure in order to not suffer of information redundancy or inconsistency. Related to this discussion it exists examples where the architecture registry has been chosen to be P2P []

Service description is another important issue in SD. In order to efficiently access service information in a database, this information needs to be classified and be represented in a flexible format. The more expressive the description is the more accurate our matches will be at the time of looking for a determined service. A classical example would be the use of the Extensible Markup Language (XML). XML is, according the WC3 group, a simple, very flexible text format widely extended that is mainly used in the exchange of data on the Web. Web Services use a XML format called WSDL in order to describe and detail their public interface. WSDL describes the protocol requirements and message formats needed to deals with the listing services for a particular Web Service.

In order to give more information weight to the description of a service we can establish relationships (schemas or ontologies) between different objects and enrich the descriptions with semantics. The use of ontologies provides a very powerful way to describe objects and their relationships to other objects. An ontology has been defined as a formal, explicit representation of a shared conceptualization [Gruber 1991]. In our case, apart from being a common vocabulary of all agents involved, it also introduces the concept of relationship between objects that will be of a very good use for the purpose of Web Service Discovery. An example of a language to describe ontologies would be OWL (Ontology Web Language).

Another issue to take into account in a service discovery system is the way services are announced and discovered. In centralized architectures services register to a server that receives requests from the clients interested in discovering services. In non-centralized architectures services make use of multicast/broadcast methods to communicate their presence, which is not suitable for large networks.

Akogrimo needs to take into account the mobile nature of services and users which must be reflected in the way services are discovered. Traditional more static focuses will have to be reviewed in order to obtain a SD system that is able to handle mobility to some extend. (E.g. due to this mobile nature, the registries will need to be updated more frequently and context information will be used to make searches more accurate.)

5.2. Functions and role

Service discovery has been matter of study in different areas for quite a long time. Therefore, there exist many types of SD systems which are specialized in different fields (ad-hoc SD, infrastructure SD, etc…). However, when thinking in the Akogrimo way it is important to notice that the structure formed by all the services available and the users is dynamic. That means, users and services may appear and disappear from the network, change their location, change their connection characteristics, require the composition of services on the fly, attach to a different administration area in the middle of a session, redirect sessions from one device to another and so forth. For this reason it is necessary to create a SD system that is aware of these kinds of events in order to take advantage of these eventualities.

Due to the changeability of the environment in Akogrimo, where nodes may move and enter other locations without any knowledge as of whether the new environment is hostile or not, it is important to have a strong security system in order to authenticate and authorize each entity and the use made of resources respectively. Therefore a mechanism has to exist to distinguish between authenticated users and services and authorized use of the services so that no unauthorized use of a resource can be performed.

Keeping in mind these issues, the Akogrimo SD system will need to evolve from the current SD technologies which cover small parts of the Akogrimo SD without taking into account the issues of a mobile and changing environment.

Akogrimo SD Requirements:

- The SD must be able to describe functional characteristics of services.
- Must be able to describe physical characteristics of resources as well as persons or entities whose nature is relevant. In addition, characteristics describing how ephemeral a service is (due to mobility) should be specified.
- Should enrich the typical descriptions with semantics and ontologies. Making it easier to automate the use and composition of services.

- Handle services failing or being abruptly disconnected: If a service fails the SD mechanism should be aware of it and update the situation accordingly.
- Services appearing: When a service appears it needs to be registered and users that have been waiting for such a service should be notified of the availability of the service.
- Services changing their context: Bandwidth, location or any kind of context characteristics may change. The SD should to be aware of it.
- The user must be authorized to use a service. This implies the need for a structure in which services are able to register as part of the system as what would be "official" services. That is, any user would need to be authorized to use a certain printer even if the interaction between user and service would be otherwise ad-hoc, without any other external interaction, like in the case of Bluetooth. In order to achieve this services would need a register where the different services are updated and probably also authenticated. In addition, this structure is necessary in order to perform charging and similar issues.
- SD must to be aware of the user's context and preferences in order to perform more accurate queries. E.g. A high-bandwidth demanding video streaming won't be useful for a user connected with a low-bandwidth network, the SD must be aware of the formats supported by the device, choose accordingly the type of service.
- In case the user can't be correctly provided with what he requested, (e.g. a video stream can't be correctly visualized because the screen is too small), a workflow should be launched in order to find a more suitable solution (such as offer to use a bigger screen available in the room).
- SD must efficiently support queries of services by service type and/or service characteristics. (E.g. search for all lpr-type printers that support double sided printing.)

SD does not:

- Automatically perform service composition.
- Interact once the user/workflow has found a suitable service.
- Perform any kind of resource reservation for the services to be used. E.g. does not perform QoS reservation for the services that require it, and so forth.

5.3. SD relations with other modules

This section describes the relationship between service discovery and other Akogrimo subsystems.

5.3.1. WP4.2 Network Middleware Layer

Context-Manager:

- Context Manager -> SD: Mobile terminals have the need of finding services within the Akogrimo system. In order to perform the most suitable match when looking for services, context information may be used. When a query is performed, it can be tuned with information about available bandwidth, size of the screen used, etc…) The communication might be established though an extended SLP-protocol or similar where semantics and ontologies should be included to to enrich the service discovery.
- SD -> Context Manager: The available devices and services that a user has in his local environment are in fact part of its context. Therefore, the SD module in the MT will accordingly feed this information to the Context Manager.

A4C:

- Authorization:

The service Discovery System can be seen as a service itself. For this reason the A4C module would treat Service Discovery as any other service within Akogrimo platform. For instance, some rules relating who can use Service Discovery can be established (Authorization procedure) if required.

As in any service, the user must be authorized to use it. To this end, the A4C server has to validate the cryptographic material provided by the user to the SD mechanism. This could be any a token that designates the user as a member of a certain VO and therefore ascribe him certain rights when looking up services. For example, members of a VO won't be allowed to see services that are restricted to a different VO.

The communication between these two modules could be established through Diameter and SAML over diameter.

- Accounting/Metering:

The user will be accounted for the use made of the SD. Discovering basic services maybe free while the search of more sophisticated services could be charged. There is a need to meter the use made of the SD by each user and communicate it to the A4C server.

This communication could probably be based in Diameter.

5.3.2. WP4.1 Mobile Network Layer

The information required from this layer is fed to the SD by means of the Context module in WP4.2. Network load, Bandwidth, etc…

5.3.3. WP4.3 Grid Infrastructure Layer

EMS

The SDS is contacted by the EMS when the discovery of a service is initiated as a result of a WF initiated in the WP4.4 (application level). The EMS looks for resources to carry out the sequence of jobs that has assigned. The communication could be established through an extended SLP-type protocol (semantics included) with LSDS whereas SOAP could be used for GrSDS system.

5.3.4. WP4.4 Generic Application Services Layer

Workflow Manager

SD is used by WP4.4 when building a composed service.

When the service requested is not prone to be executed with the current device characteristics (screen size, bandwidth, and so forth) the WP4.4 has to create workflow in order to add a new device with sufficient capabilities to the current session or redirect the session to the new device: find bigger screen, etc…

VO Manager

Design rules and rights to access services. This is not necessarily a direct interaction.

5.4. Architecture

Ideally we would like to have a unified service description and service discovery system however it is realised that very diverse services must be handled by the Akogrimo SD. There different types of SD protocols which target different types of services. We may encounter protocols to discover devices, people, places, WSs and so forth. In order to embrace the greatest number of SD types several more or less well established protocols will be used. An API could be implemented in order to hide multiple SD systems from the requestor. It seems most realistic to have two (or more) loosely coupled systems for service discovery; discovery of "local services" (candidate solutions could be e.g. SLP, SDP, UPnP, Jini, etc.) and discovery of web services/grid services (UDDI + semantic – not suited for mobility..).

Within the Akogrimo environment, we call General Service Discovery System (GSDS in short) to the Akogrimo subsystem in charge of supplying a particular service from a particular client request. GSDS have to deal with service of very different nature. On one hand, with the so-called local services/resources such as printers, beamers, virtual hard disks, and on the other hand with the Grid Services that can be defined as stateful web services proper to the semantic Grid.

For the sake of clarity and according to the different service nature, we assume that the discovery procedure involving both services types will be hardly decoupled (this assumption could be revised if required). Then, from now on, the system in charge of discovering local resources (services) will be called the Local Service Discovery System (LSDS), whereas Grid Service Discovery System (GrSDS) is the name for the module dealing with proper Grid Services.

An outline of possible architecture is shown in Figure 19.

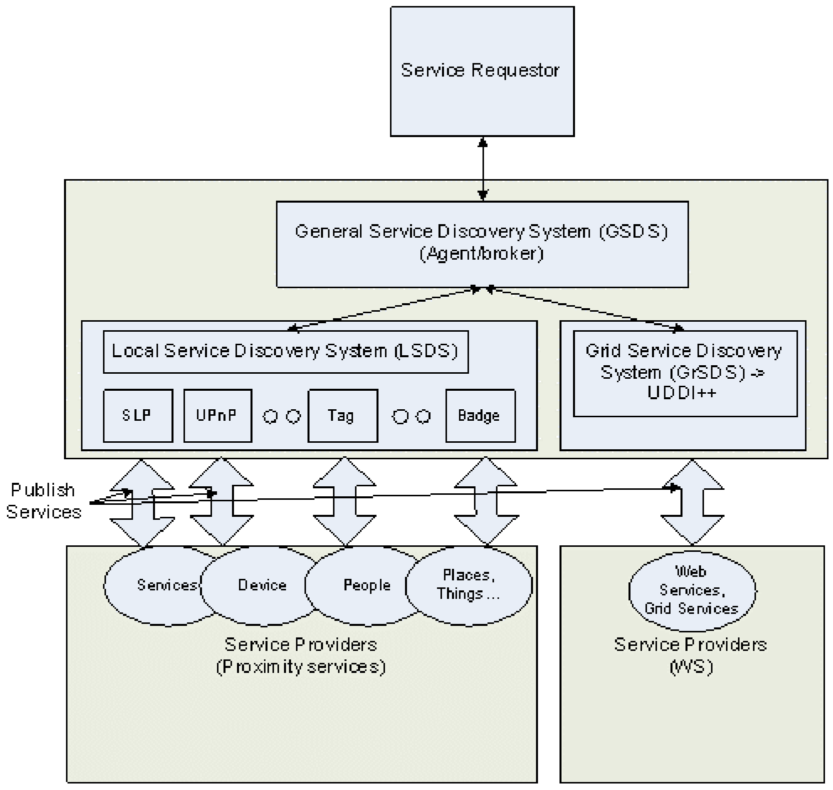

Service Requestor
General Service Discovery System (GSDS) (Agent/broker)
Local Service Discovery System (LSDS)
SLP
UPnP
Tag
Badge
Grid Service Discovery System (GrSDS) -> UDDI++
Publish Services
Services
Device
People
Places, Things...
Service Providers (Proximity services)
Web Services, Grid Services
Service Providers (WS)

Figure 19 Outline of possible architecture

5.4.1. General Service Discovery System Architecture.

The General Service Discovery System is a bridge between the Service Requests and the actual Service Discovery Mechanisms. As part of the SD architecture we distinguish the following entities:

Service Requestor: The entity requests for a certain service. This part of the system sends a service discovery request when required.

- Service Providers: This entity advertises a service. A certain service provider develop a particular service and it decides to make it public (registration/publishing procedure) by notifying to the Service Registry the service and its characteristics or capabilities.
- Service Registry: It contains a list of services together with its characteristics or with a reference to them. This entity is not entirely required. There exist some protocols where the client entity multicast service discovery requests and the proper Service responses to that request.

The corresponding Service Discovery procedures between the entities are basically:

Service Provider:

- Registration/Publishing procedure/request (Announcement). The Service Provider sends a registration/publishing request to the Service Registry to make the service public.
- Deregistration procedure. The Service Provider decides to eliminate a particular service from the Service Registry and sends a deregistration request in order to delete all the references to this services.
- Update registration procedure. Whenever the service information stores in the registry changes, an update registration procedure is sent by the Service Entity in order to update the registry data.

Service Requestor:

- Client Service procedure/request. The Service Requestor sends a service request to the Service Registry. This query can include certain parameters for the search to be more precise. These parameters can be the type of service required together with certain device capabilities.
- Client Service invocation. The Service Requestor invokes the corresponding service, once the Service Discovery System has supplied the necessary service data.

Service Registry:

- Registry Service procedure/response. The Service Registry receives the Client Service Request and a lookup process is initiated to supply a list of services matching the query parameters. Once, the lookup process has finished, the list of services is sent back as response of the Client Service Request.

- Subscribe for Service: If a Service Requestor requests a service that is temporarily not available the client has the option of being notified as soon as the service is offered.
- Check authenticity of user.

This part of the architecture would be used as the only interface to request services independently of their nature, whether local or Grid Services. However, as long as the technologies to do this separately are not consistent two different subsystems will be considered: the Local Service Discovery Subsystem and the Grid Service Discovery Subsystem. The LSDS handles in addition more several different protocols at the same time. In order to create an gateway in charge of interfacing all these different Discovery Services there exists at least one platform that could be of used: the Q3ADE 3G platform. This platform provides a range of gateway functions, which enables translation between a range of data communication technologies popular in the telecom world (e.g. CORBA, XML/SOAP, SNMP, CMIP, TL1).

5.4.2. Service discovery architecture subsystems

In order to ease the development of the Service discovery System its architecture has been separated into two loosely coupled systems. The first one, LSDS, is in charge of providing information about the so-called "local services" available in the system such as printers, DNS, DHCP etc. while the second one, GrSDS, is oriented towards the discovery of Grid services.

5.4.2.1. Local Service Discovery Subsystem (LSDS)

The services that the Local Service Discovery Subsystem should provide are for instance printers, beamers, virtual hard disks, monitors, etc... All these services belong to the Akogrimo system and the access to any of them imply interaction with the A4C and mobile terminals.

The discovery of "local services" may be accomplished by means of protocols like Service Location Protocol, UPnP, Zeroconf and so forth. The LSDS will implement an interface to unite these protocols in order to make them transparent to the user. A protocol based in SLP could be a candidate to be a common protocol to access the LSDS. A different possibility would be the use of the Q3ADE 3G platform in order to mediate among the protocols.

In order to explain the functionality of the LSDS is useful to understand how the Service Location Protocol works since the functionality of the system will be virtualized following this paradigm. The Service Location Protocol (SLP) allows network clients to discover available services on a network. Users needn't know the name of the host which provides a particular service, they only need to know the type of service they want. In addition, SLP can also provide clients with information about the configuration of those services.

The actors involved in this subsystem are Client Agent, Directory Agent and Service Agent:

- The Client Agent (CA) is a process working on the users's behalf to acquire service attributes and configuration [RFC2165].
- The Service Agent (SA) is a process on the behalf of one or more services to advertise service attributes and configuration [RFC2165].
- The Directory Agent (DA) is a process which collects information from Service Agents to provide a single repository of service information in order to centralize it for efficient access by User Agents [RFC2165].

It should be pointed out that there is no restriction for the number of Directory Agents involved within the Local Service Discovery System. The architecture and communication procedures between the Manager Agents will be further studied if needed.

Different architectures can be distinguished depending on depending, among others, the system size:

- An architecture without Manager Agents can be appropriated for a small-size network. In the absence of Manager Agents, the client agent may request in a multicast mode for a particular service. The particular matching service will unicast back to the User agents with the required parameters for the Client agent to use the service. This approach does not seem to be adequate since every Service Agent should implement Service and Manager Agents roles (including matching processes).
- An architecture with an unique Manager Agent. This centralized architecture can be very convenient for a medium size network. In this particular case, the service registration process will be done on this element. All the communication between user and service agent will be established through Manager Agent.
- An architecture with equally-functional Manager Agents. This distributed architecture can be organised regarding different "domain" or "scope" entities. Each of this entities will be organised as a centralized architecture (briefly described previously) without a main manager agent. In this case, there will not be functional distinction between Manager Agents. This lead to a multicast communication between them when synchronization information process required.
- An architecture with several Manager Agents coordinated by a Main Manager Agents. This architecture means that all synchronization procedure is coordinated via the Main Manager Agent. These elements possess information of the rest of Manager Agents.

The last two architectures assure an easier scalability of the system but differ in the way synchronization procedure between Manager Agents will occur.

It is important to notice that the use of a Service Agent is practically mandatory since ad-hoc architectures are more difficult to handle if we take into account those mechanisms to provide the authentication of users and services needs to be provided. Protocols like UPnP will probably need to be adapted to accomplish this requirement.

Regarding register time, the type of information that characterizes the local services (resources) can be classified this way:

- Static-attributes: These attributes are those parameters that do not normally change. They describe services capabilities such as printing speed, printer type (colour, black and white) and sent to the Directory Agent at service register time.
- Dynamic-attributes: The attributes need to be updated when required. Processor load, number of instances running, queued jobs, network connection are examples of dynamic-attributes. Usually the fact of having dynamic attributes means that the register needs to be refreshed more frequently.

Regarding the type of information can be separated in

- Service attributes: Attributes enclosing service capabilities information.
- Context-attributes are defined as any information that can be used to characterize the situation of a service. Since the SD is not 100% context aware most applications that really need to support context awareness will need to make use of the Context Manager. However, in the SD, some not very changing context-information may me stored and a verification of the validity of this data is made by the user after being aware of the existence of the service.

In a general view, it can be said that when a Service Requestor request a certain service with certain characteristics, a matching process between requests attributes and services attributes will be done. From such matching process a service list will be supplied to the Service Requestor. A second comparative process will lead to the establishment of preferences or priorities within the elements of the list (the nearest printer or the less loaded printer) due to context-attributes (location, loading)...

Location is considered to be approximated by the region in which the user/service is, that may be defined e.g. by the subnetwork, in the case the entity is dynamic. In the case that the entity is static ("static context") the exact information of its position could be part of the description of the service or the request performed by the user.

5.4.2.1.1. Service Description and Discovery at the Mobile Network layer

In general, one needs mechanisms for determining available network interfaces, their QoS properties, tariffs, end-to-end QoS, etc. Discovery of end-to-end network transport services suitable for higher layer services ultimately depends on the physical location of the end-systems involved. That is, what are the properties of the different access and transport networks that can be used for network transport. Hence, after higher layer services that have specific QoS requirements have been discovered it will be necessary to initiate a negotiation of end-to-end network services. These issues are considered out-of-scope.

At this layer of local service discovery we encounter network related services:

(Refer to the State of the Art document for an extensive explanation of these kinds of discovery protocols)

Discovery of access networks

To discover the connectibility provided by different access networks, e.g. LAN, WLAN, GPRS, and UMTS.

Discovery of basic network connectivity

This refers to services provided to be fully operative in a network. Here we would have protocols like Dynamic Host Configuration Protocol (DHCP), IPv6 autoconfiguration, Domain Name System (DNS) or even the Web Proxy Auto Discover (WPAD)

5.4.2.1.2. Discovery of Devices

In the following several types of discovery protocols will be evaluated taking into account the needs of Akogrimo.

Service Location Protocol (SLP) set forth by Internet Engineering Task Force (IETF) uses multicast and limiting its scalability in larger networks. SLP only addresses service location discovery, leaving other aspects of service discovery to the application. The services are located by means of URL and some additional name-value pairs, called attributes. The definition of services are specified by templates defined in [RFC2609]. SLP is able to:

- Look for all services of a determined type
- Query by attributes by making use of LDAP's query language.
- Request the attributes of a services

In SLP there different entities can be observed:

- User Agents (UA): Are the entities who perform the query.
- Service Agents (SA): Are the entities that announce services
- Directory Agents (DA): Are entities that keep information about the available services in the network.

SLP may work in two different modes depending on the size of the network that is managing. In case of having small networks just UAs and SAs are used and the queries are performed by means of broadcast requests whereas in bigger networks a centralized modality is used in which a DA is introduced which a UA queries in order to look for a service. This approach permits SLP to be suitable for both small and large networks. In addition, SLP creates the notion of scopes in order to group services. SLP would need to be extended in the case of the centralized mode in order to fully support mobility of services.

Regarding security, SLP includes Public key based mechanisms in order to sign service announcements.

UPnP, Microsofts service discovery technology is based on the Simple Service Discovery Protocol (SSDP) and makes extensive HTTP protocol and depends heavily on multicast. Once the service is discovered by means of SSDP, a full description is provided to the client in order to use the service. UPnP uses XML in communication and service descriptions. These characteristics make UPnP inefficient or sometimes unsuitable for smaller devices and low-bandwidth networks. In terms of mobility UPnP has a good support since it assures that no unwanted state is left behind even in case abrupt disconnection. In Akogrimo there would be a need to develop a method through which the services can be proofed to be part of the Akogrimo structure.

Jini is a Suns Java-based service discovery protocol. It requires Java Virtual Machine (JVM) in a client and that consumes processing power and memory. In addition, the JVM is not so portable as is intended to be and some small mobile devices may have problems to use a JVM which permits the use of IPv6 (E.g. in Pocket PC 2003). In the Jini architecture, service providers supply Java objects that are used by the clients by means of methods such as RMI, CORBA or SOAP. The use of multicast makes it difficult for Jini to scale up for Internet.

Zeroconf is a set of technologies that allow the autoconfiguration of two or more communicating entities. Zeroconf aims to provide: automatic interface configuration, automatic multicast address allocation, Name-to-address translation and, service discovery and delivery. To accomplish this zeroconf relies on Multicast DNS. The drawback of zeroconf is that service queries by characteristic are not possible. Instead, the service's characteristics must be fetched service by service. Mobility is well supported since no centralized architecture is used. Authentication methods would need to be developed.

Bluetooth SDP is a service discovery protocol which is part of the Bluetooth protocol stack. It performs only service discovery and it is limited to single hop networks only.

The Salutation Consortium consists of several peripheral device manufactures such as HP, IBM and Cannon. The Salutation Manager Protocol (SMP) is used for Salutation Manager communication between other Managers, clients and servers. It is based on Remote Procedure Call (RPC), which is so resource demanding that it does not scale well to small devices and low bandwidth networks used in mobile pervasive environments. Salutation Lite is reduced implementation of Salutation.

These protocols are candidates to be integrated into the LSDS, however, since they are utilized to cover similar kinds of discovery of services it is probably easier to start out with one or two of them, thus providing the necessary functionality and afterwards cover compatibility issues and extensions by adding new protocols.

Technical challenges for these current service discovery protocols are devices with limited computer power, restricted memory, varying network bandwidth and time to time loss of traffic. Another difficulty is that they don't scale up to larger mobile and wireless networks in terms of number of devices and network size in hops. They are more suitable for environments, where there are low mobility of client devices and stationary devices serving these clients. Access points are used to connect client devices to the network, where devices are homogeneous consisting mostly of portable personal computers, portable desktop assistants (PDAs).

5.4.2.1.3. Discovery of people (and other real world objects; people, things and places)

A part of the LSDS is the discovery of real entities like people, places or things, which are may not be directly conceived as a service but they may be indirectly. The Context Manager also provides the capability of localizing people, however the difference here is that the entities to be localized are not an active part of the system, that is, they may not even have computing power by themselves. Examples of these entities could books in a book store, or people that is not actively involved in the system. This can be done by using technologies like tag, active badge, rfid etc. which allows a person (or other physical object) to be discovered. The physical object has a virtual counterpart(s) (e.g. web service) that represents the physical object [RFC2165], [SG].

5.4.2.2. Grid Service Discovery Subsystem

Grid Services means Stateful Web Services. The General Service Discovery Subsystem should provide Grid services agents in a dynamic way to Service Requestors. This becomes basic when Grid Service composition should be accomplished automatically.

A discovery of proper grid services is also supplied by the GSDS. The GrSDS is the module in charge of providing this capability.

The Grid Services are aligned with Web Service Technology since they may allow dynamic discovery and composition of services. However, up to now Web Service Technology is not able to provide such characteristics since it requires the “manual” human intervention in order to discover suitable web services or to build a composed Web Services out of other more simplistic web services. The area of Semantic Web Services handles with this problems. The approach consists of the definition of a common language together with their relationship in order to build a framework where dynamic discovery and composition might be able.

Currently, there exist several research lines to provide the above characteristics to the Web Service Technology. Examples of that are the Web Service Modeling Ontology (WSMO) and the Web Service Modeling Framework (WSMF).

- The WSMF and WSMO approaches share a common architecture. Both frameworks are made up of four different elements, *ontologies*, providing the terminology to be used, *goal repositories*, specifying objectives to be fulfilled by the client invoking the web service, *web service descriptions*, being the definition of the web services and *mediators* or adapters that bypass interoperability problems. In fact, WSMO working group claims to be a refinement and extension of WSMF. The WSMO framework has the advantage of fully supporting Semantic Web Services as well as non-semantic web services. This choice could be of good use in case that Non-semantic web services are employed.

Other approaches have been also developed. Following more closely the line of the architecture presented in previous sections, there exists a framework where the philosophy of storing the web services within a UDDI registry is complemented by the adaptation of web services description in a semantic way.

Basically, there are two processes involved: registration process and discovery process. The process of registration is carried out in a similar way to the standard Web Services. From the java code it is build the Web Service Description (WSDL). However, since we are interested in introducing semantics, there should be a converter from WSDL to a semantic language. This framework provide a tool to so. Once we have the "Semantic web service description", the real registration procedure begins. The registry used is UDDI and several apis are defined to introduce in a proper way the semantic web service description within the UDDI format. This approach uses OWL-S as ontology languages based on web Services.

On the other hand, clients should be able to discover in an easy way web services described by means of an ontology language. For this reason a matchmaker software is developed in order to carry out such task [SWST].

5.4.3. Akogrimo service description and ontology

The way services are described represents an important issue in Service Discovery process. The efficiency of the retrieval information process is very related to the service description adopted [KLE04]. It can be distinguished four different types of service retrieval depending on the chosen service description:

- Keyword-based retrieval. The searching procedure is based on keywords given supplied in the Client service request. This method does not considered at all the semantics of the keywords, so synonyms of homonyms can not be distinguished leading to a low retrieval precision.
- Table-Based retrieval. The service description and Client service requests are made out of attribute-value pairs. Even when more precise retrieval procedure is obtained, it keeps with the synonym and homonym problems (SLP is an example).

- Concept-based retrieval. The service description implies the use of ontologies in order to classify services. In that case the retrieval precision is higher than for the other two retrievals. The difficulties arise in the ontology definitions.
- Deductive retrieval. Logic is used when expressing service semantics.

The service description includes the definition of certain attributes. Regarding their changing nature, the following classification can be established:

- Static-attributes: These attributes are those parameters that do not normally change. They describe services capabilities such as printing speed, printer type (colour, black and white). They are kept in the service description registered in the Service Registry.
- Dynamic-attributes: The attributes need to be updated when required. Processor load, number of instances running, queued jobs, network connection are examples of dynamic-attributes. They are managed by the Context-Manager.

The use of ontologies guarantees that the service discovery process will be carried out more precisely. A non-ontologic approach will imply the system not to recognise synonyms (concepts syntactically different but semantically equal) and homonyms (concepts semantically different but syntactical equal). In addition, the definition of an ontology will also guarantee the use of the same language for all actors playing roles within the Service Discovery process [BRO04].

An ontology can be defined as a formal, explicit representation of a shared conceptualisation.

6. Relationship to other layers of Akogrimo

This section shows some high-level sequence diagrams, to illustrate how the components of the network middleware layer (corresponding to Akogrimo Work Package 4.2) interacts with the other layers of the Akogrimo architecture. Throughout this section, *4.1* refers to the network layer, 4.3 to the Grid infrastructure layer, and 4.4 to the Grid application support layer.

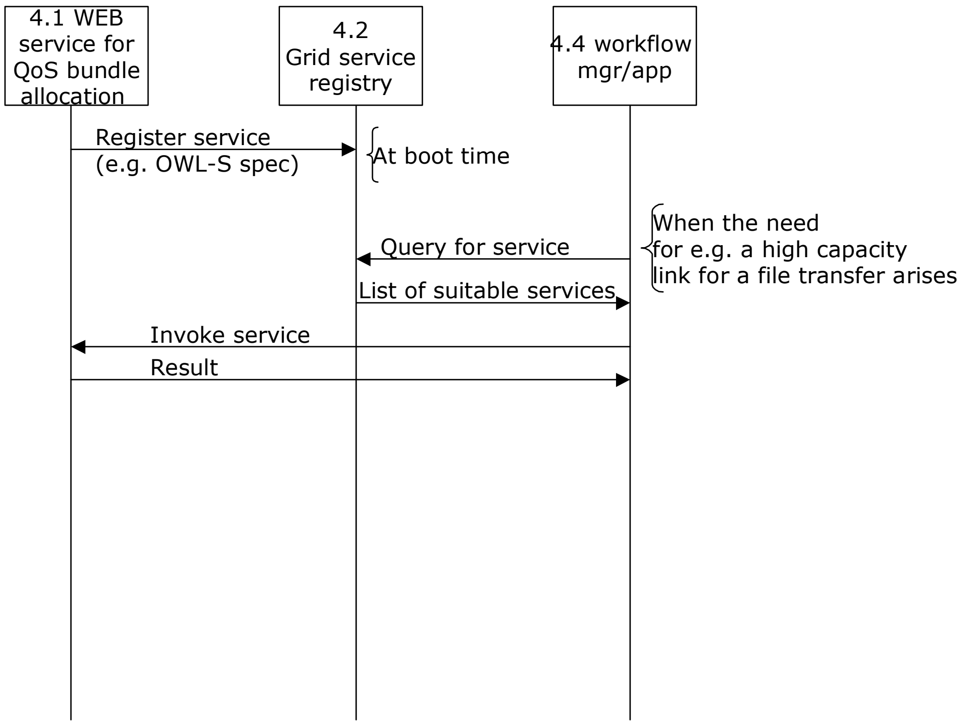

4.1 WEB service for QoS bundle allocation
4.2 Grid service registry
4.4 workflow mgr/app
Register service (e.g. OWL-S spec)
At boot time
When the need for e.g. a high capacity link for a file transfer arises
Query for service
List of suitable services
Invoke service
Result

Figure 20, discovering and using the QoS allocation service.

Figure 20 shows how an application can invoke am enhanced network service, e.g. to allocate a high-capacity link between two network nodes[2]. The network layer implements a WEB service which virtualises this allocation service. The details of how that service was implemented is thus hidden from the other layers. At boot time ()and at regular intervals), the network layer will register this WEB service with the GrSDS (Grid Service Discovery Sub-system). The registration will be done using an ontology language (OWL-S), and will contain:

- The endpoint (address) where the service can be found.

[2] This diagram is simplified, and does not show interactions with the A4C subsystem.

- A machine-readable description of the functionality the service offers.
- Formal interface definitions (WSDL).
- Other meta-data, e.g. concerning tariffs.

When an application needs such a service, it will query the GrSDS for a list of relevant services, and pick one of these. The application will then invoke the chosen service.

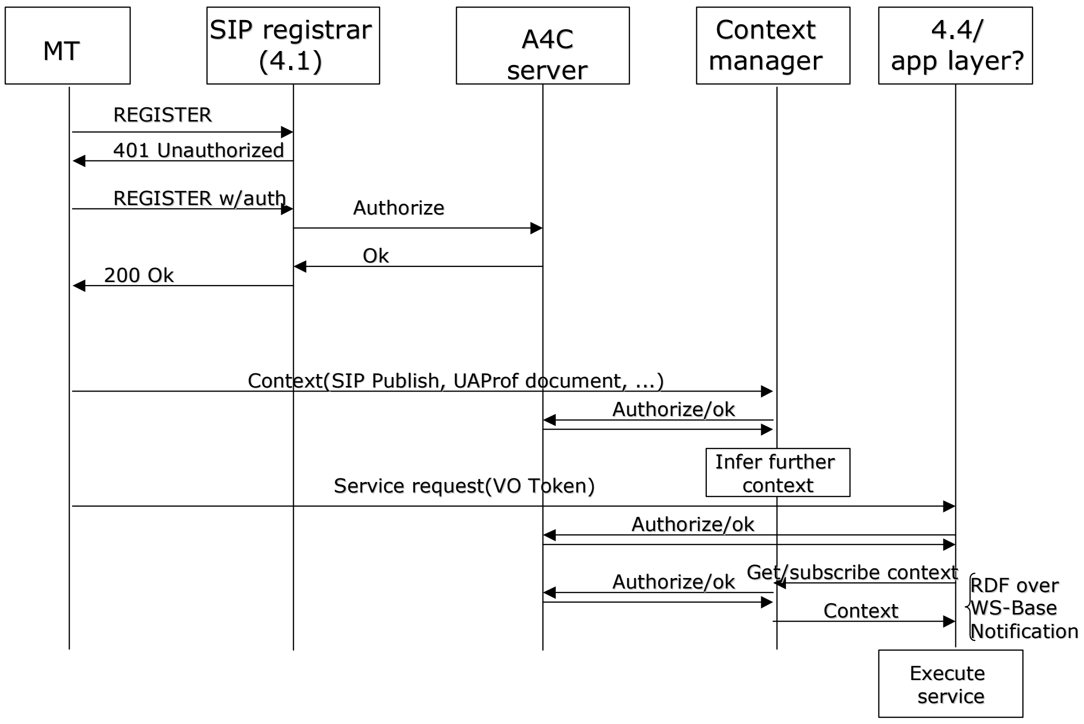


Figure 21, logging on to a virtual organization, and invoking a service.

Figure 21 shows how a user may log on to a virtual organization, and then invoke a service. The user logs on to a mobile terminal, which will then do a SIP registration with a SIP registrar (part of the 4.1/network layer). The registrar will use the A4C server for verifying the user's credentials. The mobile terminal will then send context data to the context manager, in the form of:

- A SIP publish message, with SIP/SIMPLE presence information[3].
- A UAProf document, describing terminal capabilities and preferences (for media formates, etc.)

[3] Here we assume that the context manager also is a SIP PA (Presence Agent).

At some point the, the user chooses to invoke a service. The service then subscribes to context updates for the user, to be able to adapt itself to any context changes.

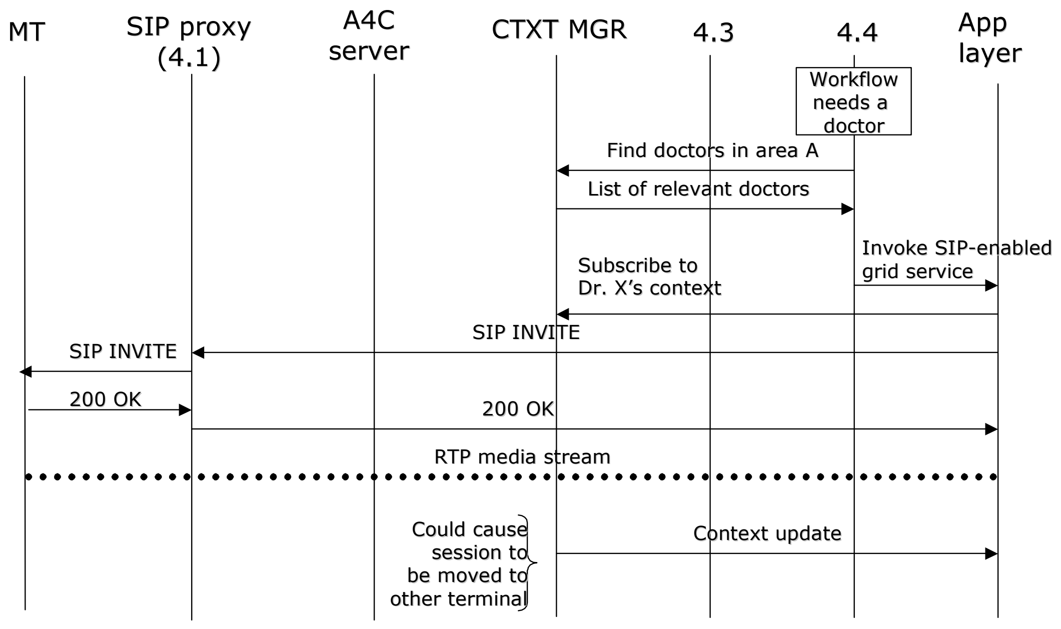


Figure 22, session setup, initiated by the virtual organization.

Figure 22 shows how a session may be initiated by the virtual organization. In this example, a workflow has reached a point where there is a need for a doctor to have a look at an ultrasound-video. The workflow manager queries the context manager for a list of relevant doctors, i.e. doctors who are on-line, has a terminal with video capabilities, and the right qualifications. An application layer service is then invoked, which subscribes to context updates for the chosen doctor, and sets up a SIP session to him/her. If the doctor e.g. walks into a room where a wall-mounted screen is available, the service will be notified, and may choose to move the session (video stream) to this screen.

7. References

[BaDu04] M. Baldauf & S. Dustdar, *A Survey on Context-aware systems*, technical report, Distributed Systems Group, Technical University of Vienna, 2004

[BaNo04] M. Baker, M. Nottingham: *The "application/soap+xml" media type,* RFC3902, September 2004

[BeSchu03] S. Berger, H. Schulzrinne, S. Sidiroglou, X. Wu: *Ubiquios Computing Using SIP*, NOSSDAV 03, May 2003

[BRO04] T. Broens*, Context-aware, Ontology based, Semantic Service Discovery*, Thesis for a Master of Science degree in Telematics from the University of Twente, Enschede, The Netherlands

[CLGZA03] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko. *Diameter Base Protocol*, RFC 3588, September 2003.

[D2.3.1] Akogrimo Consortium: *D2.3.1 Testbed Description Report*, 2005.

[De00] N. Deason: *SIP and SOAP*, draft-deason-sip-soap-00, June 2000

[De01] N. Deason: *SIP and SOAP White Paper*, Ubiquity Software Corporation, May 2001

[Dey01] A. K. Dey, *Understanding and Using context*, Personal and Ubiquitous Computing Journal, Volume 5 (1), 2001, pp. 4-7.

[FoKe] I. Foster, C. Kesselmann, J. M. Nick, S. Tuecke: *The Physiology of the GRID*, Draft, Work in progress

[FOPTY05] D. Forsberg, Y. Ohba, B. Patil, H. Tschofenig, A. Yegin. *Protocol for Carrying Authentication for Network Access (PANA)*, Intenet draft, expires June 2005

[GGF05] GGF Usage Record Working Group, http://www.psc.edu/~lfm/Grid/UR-WG/ , Last visit: March 2005

[Gruber 1991] T.R. Gruber, *The role of common ontology in achieving sharable, reusable knowledge bases,* Principles of knowledge representation and reasoning, proceedings from the second international conference, 1991.

[HIR02] K. Henricksen, J. Indulska, and A. Rakotonirainy, *Modeling Context Information in Pervasive Computing Systems*, Proceedings of the First International Conference on Pervasive Computing, 2002.

[HLRZ04] Herzog, R., Lausen, H., Roman, D., Zugmann, P. (eds.): *WSMO Registry*, WSMO working draft, available at http://www.wsmo.org/2004/d10/v0.1/ , 2004

[KLE04] M. Klien et al., *Towards High-Precision Service Retrieval*, IEEE Internet Computing, February 2004

[LGGVS00] C. de Laat, G. Gross, L. Gommans, J. Vollbrecht, D. Spence: *Generic AAA Architecture,* RFC 2903, August 2000

[MCKP03] E. Maler, S. Cantor, J. Kemp, R. Philpott: *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML).* OASIS, September 2003. Document ID oasis-sstc-saml-core-1.1 http://www.oasis-open.org/committees/security/

[MLP] Open Mobile Alliance, *Mobile Location Protocol*, 3.2Draft Version 2004-11-02, http://member.openmobilealliance.org/ftp/public_documents/loc/Permanent_documents/OMA-MLP-Spec-V3_2-20041102-D.zip

[Moby02] Moby Dick Consortium: *AAAC Design*,Moby Dick Deliverable D0401, January 2002

[Most03] S. Kouadri Mostéfaoui, G. Kouadri Mostéfaoui, *Towards A Contextualisation of Service Discovery and Composition for Pervasive Environments*, University of Fribourg, 2003.

[MPP] Ericsson, http://www.ericsson.com/mobilityworld/sub/open/technologies/mobile_positioning/index.html

[PKPS02] Paolucci, M., Kawamura, T., Payne, T. R., Sycara, K.: *Importing the Semantic Web in UDDI*, Proceedings of Web Services, E-business and Semantic Web Workshop, 2002.

[RFC2543] M. Handley, H. Schulzrinne: *SIP: Session Initiation Protocol*, RFC2543, IETF, March 1999

[RFC2616] R. Fielding, J. Gettys, J. Mogul, H. Rystyk, L. Masinter, P. Leach, T. Berners-Lee: *Hypertext transfer protocol – HTTP/1.1*, RFC2616, IETF, June 1999

[RFC3265] RFC 3265, *Session Initiation Protocol (SIP)-Specific Event Notification*, http://www.ietf.org/rfc/rfc3265.txt?number=3265

[RFC3856] RFC 3856, *A Presence Event Package for the Session Initiation Protocol (SIP)*, http://www.ietf.org/rfc/rfc3856.txt?number=3856

[RFC3863] RFC 3863, *Presence Information Data Format (PIDF*). http://www.ietf.org/rfc/rfc3863.txt?number=3863

[RFC3903] RFC 3903, *Session Initiation Protocol (SIP) Extension for Event State Publication*, http://www.ietf.org/rfc/rfc3903.txt?number=3903

[RHKS02] Christoph Rensing, Hasan Hasan, Martin Karsten, Burkhard Stiller: *AAA: A Survey and Policy-based Architecture and Framework*. IEEE Network Magazine, IEEE, December, 2002.

[RoSchu00] J. Rosenberg, H. Schulzrinne: *Third party call control in SIP*, Internet Draft, IETF, March 2000, Work in Progress

[RoSchu99] J. Rosenberg, H. Schulzrinne: *Internet telephony: Architecture and protocols*, IEEE Network, Vol. 13, pp. 18-23, May/June 1999

[RPID] IETF Internet Draft, *Rich Presence Extensions to the Presence Information Data Format (PIDF)*, http://www.ietf.org/internet-drafts/draft-ietf-simple-rpid-05.txt

[SAW94] Schilit B., Adams, N. & Want, R. (1994). *Context-aware computing applications*, Proceedings of the 1st International Workshop on Mobile Computing Systems and Applications, 85-90. Los Alamitos, CA: IEEE.

[SchuRo00] H. Schulzrinne, J. Rosenberg: *The session initiation protocol: Internet-centric signaling*, IEEE Communications Magazine, Vol. 38, Oct. 2000

[SchuRo99] H. Schulzrinne, J. Rosenberg: *Internet telephony: Architecture and protocols – an IETF perspective*, Computer Networks and ISDN Systems, Vol. 31, pp. 237-255, Feb. 1999

[SchuWe00] H. Schulzrinne, E. Wedlund: *Application Layer Mobility Using SIP*, ACM SIGMOBILE Mobile Computing and Communications Review, Vol. 4, Issue 3, pp. 47-57, July 2000

[SIMPLDM] IETF Internet draft, *A Data Model for Presence*, http://www.ietf.org/internet-drafts/draft-ietf-simple-presence-data-model-02.txt

[SIMPLE] *SIP for Instant Messaging and Presence Leveraging Extensions (simple)*, http://www.ietf.org/html.charters/simple-charter.html

[Sk03] A. Skonnard: *Understanding SOAP*, MSDN (http://msdn.microsoft.com/webservices/understanding/webservicebasics/default.aspx?pull=/library/en-us//dnsoap/html/understandsoap.asp), March 2003

[Tu03] S. Tuecke, et al.: *Open GRID Services Intrastructure*, draft-ggf-ogsi-gridservice-33, June 2003

[UAProf] Wireless Application Forum, *WAP-248-UAPROF-20011020-a*, Version 20 October 2001, http://www.openmobilealliance.org/tech/affiliates/wap/wap-248-uaprof-20011020-a.pdf

[UDDI0] *UDDI Technical Whitepaper*, http://uddi.org/pubs/uddi-tech-wp.pdf, October 2004

[UDDI1] *UDDI Specs*, http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm

[UPnP] UPnP Forum, *UPnP Device Architecture 1.0*, Dec. 2003, http://www.upnp.org/resources/documents/CleanUPnPDA101-20031202s.pdf

[VCZZ01] J. Vollbrecht, G. Carle, S. Zander, T. Zseby: *Session ID,* Interbet-Draft, February 2001

[W3C00] W3C: *SOAP 1.1 Note*, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/, May 2000

[W3C02-2] W3C: *SOAP Version 1.2 Part 2: Adjuncts* (W3C Recommendation 24 June 2003), http://www.w3.org/TR/soap12-part2/, June 2003

[W3C03] W3C: *SOAP Version 1.2 Part 1: Messaging Framework* (W3C Recommendation 24 June 2003), http://www.w3.org/TR/soap12-part1/, June 2003

[Weis92] M. Weiser, *The Computer for the Twenty-First Century*, Scientific American, pp. 94-10, September 1991.

[WS04] *Introduction to Web Services Technologies: SOA, SOAP, WSDL and UDDI*, http://www.informit.com/articles/article.asp?p=336265, September 2004

Annex A. SIP presence and context information

A presence document is a well formed XML file, containing an encoding declaration (UTF-8) and using the content type "application/pidf+xml" defined in the RFC3863 and enhanced by the presence data model and PIDF internet drafts. This information in included in the body field of the PUBLISH SIP message to upload presence information.

A.1. PIDF XML format description

Presence Information Data Format document [RFC3863] defines the format of presence data for a presentity, through the definition of a new media type ″application/pidf+xml″ to represent the XML MIME entity. The most important element in this XML document is the "presence tuple" element. Each tuple describes a "communication capability" of the presentity, associated to a specific service available. To specify the availability of that service, each tuple groups other child elements which provide the information itself.

A PIDF document can contain several tuples, and it is possible to provide several PIDF files associated to the same presentity. This mechanism provides a way of segmenting presence information. Protocols or applications may choose to segment the presence information associated with certain presentity for any number of reasons, for example, because components of the full presence information for a presentity have come from distinct devices or different applications on the same device, or have been generated at different times.

The following table summarizes the possible content of a PIDF tuple, their meaning and values.

Table 2 PIDF tuple child elements

Element	Description	Values
status	Status values of the tuple	
basic	Availability (open) or not (closed) for any communication means	open/close
contact	URL of the contact address to reach the presentity and communicate to it depending on the tuple basic status	
note	A string value used for a human readable comment	
timestamp	The date and time of the status change of the tuple	

The basic rules to build a well conformed application/xml-pidf document are the following:

- The root element of the PIDF XML document is a <presence> element that MUST have an ‘entity’ attribute that is the ‘pre’ URL of the presentity publishing this presence document and must contain a namespace declaration (urn:ietf:params:xml:ns:pidf:).
- The <presence> element can contain any number (including 0) of <tuple> elements followed by any number (including 0) of <note> elements, followed by any number of OPTIONAL extension elements from other namespaces.
- The <tuple> element carries a presence tuple, consisting of a mandatory <status> element, followed by any number of optional extension elements (possibly from other namespaces), followed by an optional <contact> element, followed by any number of optional <note> elements, followed by an optional <timestamp> element. The <tuple> element MUST contain an ‘id’ attribute to distinguish this tuple from other tuples in the same presentity and must have an unique value for the same presentity.
- The <status> element contains one OPTIONAL <basic> element, followed by any number of optional extension elements (possibly from other namespaces), under the restriction that at least one child element appears in the <status> element (this status value might not be the <basic> element).
- The <contact> element contains an URL of the contact address. It optionally has a ’priority’ attribute, whose value means a relative priority of this contact address over the others. The value of the attribute MUST be a decimal number between 0 and 1 inclusive with at most 3 digits after the decimal point. Higher values indicate higher priority. If the ’priority’ attribute is omitted or out of the range, applications MUST assign the contact address the lowest priority.
- The <note> element contains a string value, which is usually used for a human readable comment. The <note> element SHOULD have a special attribute ’xml:lang’ to specify the language used in the contents of this element.
- The <timestamp> element contains a string indicating the date and time of the status change of this tuple. It SHOULD be included in all tuples unless the exact time of the status change cannot be determined.
- A processor of presence information MUST ignore any XML element with an unrecognized name including all of the element content, even if it appears to contain elements with recognized names. In order to understand a complex extension, nested elements within an extension element might need to be marked as mandatory. In such cases, the element name is qualified with a mustUnderstand=’true’ or mustUnderstand=’1’ attribute, but this attribute MUST be used only within optional elements nested in a <status> element.
- The following example shows presence information of the presentity someone@example.com. Usually, this presentity can be accessed via phone or email, however, at this moment only call phone is possible.

```
<?xml version="1.0" encoding="UTF-8"?>
   <presence xmlns="urn:ietf:params:xml:ns:pidf"
```

```
        entity="pres:someone@example.com">
      <tuple id="sg89ae">
        <status>
          <basic>open</basic>
        </status>
        <contact priority="0.8">tel:+09012345678</contact>
        <note>You can contact me at phone number +09012345678</contact>
      </tuple>
      <tuple id="sg34tr">
        <status>
          <basic>closed</basic>
        </status>
        <contact priority="0.1">mailto:someone@example.com</contact>
        <note>Please, do not send me an email, I do not have access to the
email</contact>
      </tuple>
    </presence>
```

A.2. Data Model for Presence format description

The SIMPLE Presence Data Model [SIMPLEDM] describes how to take real world systems and map them into presence document, providing characteristics and status info about a presentity.

There are three main elements defined in the data model:

- **Person**: that represents the human user and their states that are relevant to presence systems.
- **Tuple**: it represents a service to communicate with the user (instant messaging, VoIP via SIP, videoconference via SIP, etc.), that is a point of reachability for communications that can be used to interact with the user.
- **Device**: the physical operating environment in which services execute.

The tables showed below list the XML elements defined in the presence data model to describe person and device respectively, since services are represented by <tuple> PIDF XML element (see Table 2)

Table 3 Data model elements to describe persons

Element	Description
status	Status values of the person
personStatus	Status values of the person included in the status element
personCharacteristics	Characteristics values of the person included in the status element
note	A string value used for a human readable comment about person

Element	Description
timestamp	The date and time of the status change of the person

Table 4 Data model elements to describe devices

Element	Description
status	Status values of the device
device-id	URN that identifies a device
deviceStatus	Status values of the device included in the status element
deviceCharacteristics	Characteristics values of the device included in the status element
note	A string value used for a human readable comment about the device
timestamp	The date and time of the status change of the device

The <presence> element is the root element in the XML doc and there can be 0 o more instances of <person>/<device>/<tuple> elements underneath <presence> element and each one have a mandatory 'id' attribute which contains the instance identifier for the person/device/tuple.

A.3. RPID XML format description

Rich Presence Information Data Format [RPID] defines additional presence attributes to describe person, service and device data elements. These attributes are specified by XML elements which extend the PIDF <tuple>, <device> and <person> elements defined in the data model.

An important advantage of RPID is that it is easy to derive info from other information sources, such us calendars, the status of communication devices such as telephones, typing activity, physical presence detectors, etc.

The namespace URIs defined for persons, devices and services are URNs, using the namespace identifier 'ietf'. A new namespace <status> is used for extending the PIDF <status> namespace:

- urn:ietf:params:xml:ns:pidf:status:rpid-status
- urn:ietf:params:xml:ns:pidf:rpid-tuple

- urn:ietf:params:xml:ns:pidf:rpid-person
- urn:ietf:params:xml:ns:pidf:rpid-device

The following tables summarize the possible RPID elements applied to person, device and tuple elements respectively, their meaning and possible values. The elements marked with '1' MAY be qualified with the 'since' and 'until' attributes to describe the absolute time when the element assumed this value and the absolute time until which is element is expected to be valid. The 'since' time must be in the past, the 'until' time in the future relative to the time of publication of the presence information and, if available, the PIDF timestamp element.

Table 5 RPID elements applicable to person element

Element	Description	Values
activities[1]	What the person is currently doing. expressed as an enumeration of activity-describing elements. A person can be involved in multiples activities at the same time.	away
		appointment
		busy
		holiday
		in-transit
		meal
		meeting
		on-the-phone
		performance
		permanent-absence
		sleeping
		steering
		travel
		vacation
mood[1]	Mood of the person	afraid
		amazed
		angry
		annoyed
		anxious
		ashamed
		bored
		etc
place-is[1]	Properties of the place the person is currently at.	bright
		dark
		noisy
		quiet
place-type[1]	Type of place the person is currently at.	aircraft
		airport

Element	Description	Values
		bus
		car
		convention center
		home
		hotel
		industrial
		library
		mall
		office
		outdoors
		public
		public-transport
		restaurant
		school
		ship
		station
		street
		theater
		train
		truck
sphere[1]	Current status and role that the person plays.	work
		home
privacy[1]	Type of communication third parties in the vicinity of the presentity are unlikely to be able to intercept accidentally or intentionally.	audio
		video
		text
time-offset[1]	The time zone the person is in, expressed as the number of minutes away from UTC	
status-icon[1]	URI pointing to an image (icon) representing the current status of the person.	
user-input	It summarizes any user input across all devices and services operated by the presentity.	active
		idle
class	The class of the person.	

The following XML document represents presence information of the presentity someone@example.com. This presence info describes a person who is working at the school library, a quiet place that discourages noise, conversation and other distractions. Moreover, he is happy because he thinks he is doing a good work.

```
<?xml version="1.0" encoding="UTF-8"?>
  <presence xmlns="urn:ietf:params:xml:ns:pidf"
  xmlns:p="urn:ietf:params:xml:ns:pidf:person"
```

```
   xmlns:rp="urn:ietf:params:xml:ns:pidf:rpid-person"
   entity="pres: someone@example.com"
   xsi:schemaLocation="urn:ietf:params:xml:ns:pidf pidf.xsd
   urn:ietf:params:xml:ns:pidf:status:rpid-person rpid-person.xsd
   urn:ietf:params:xml:ns:pidf:status:rpid-status rpid-status.xsd">
     <p:person>
       <p:status>
       <rp:activities>
         <rp:school/>
       </rp:activities>
       <rp:class>composed</rp:class>
       <rp:mood>
         <rp:happy/>
         <rp:text>I think we are doing a good project!</rp:text>
       </rp:mood>
       <rp:place-type until="2005-03-16T17:30:00Z">
         <rp:library/>
       </rp:place-type>
       <rp:place-is until="2005-03-16T17:30:00Z">
         <rp:quiet/>
       </rp:place-is>
       <rp:sphere until="2005-03-16T17:30:00Z">
         <rp:work/>
       </rp:sphere>
     </person>
   </presence>
```

Table 6 RPID elements applicable to device element

Element	Description	Values
user-input	The user-input or usage state of the device, based on human user input, e.g., keyboard, pointing device or voice	active
		idle
privacy[1]	Type of communication third parties in the vicinity of the presentity are unlikely to be able to intercept accidentally or intentionally.	audio
		video
		text
status-icon[1]	URI pointing to an image (icon) representing the current status of the device.	
class	The class of the device.	

The following XML document represents presence information of the presentity someone@example.com who is using a PDA at this moment.

```
<?xml version="1.0" encoding="UTF-8"?>
   <presence xmlns="urn:ietf:params:xml:ns:pidf"
   xmlns:d="urn:ietf:params:xml:ns:pidf:device"
   xmlns:rd="urn:ietf:params:xml:ns:pidf:rpid-device"
   entity="pres: someone@example.com"
   xsi:schemaLocation="urn:ietf:params:xml:ns:pidf pidf.xsd
   urn:ietf:params:xml:ns:pidf:status:rpid-device rpid-device.xsd
   urn:ietf:params:xml:ns:pidf:status:rpid-status rpid-status.xsd">
     <d:device>
       <d:status>
       <rd:device-id>
```

```
          http://mydevice/id
        </rd:device-id>
        <rd:class>PDA</rp:class>
        <rd:user-input>active</rd:user-input>
         <rd:text>At this moment I am using a PDA</rd:text>
      </device>
   </presence>
```

Table 7 RPID elements applicable to tuple element

Element	Description	Values
status-icon[1]	URI pointing to an image (icon) representing the current status of the service.	
privacy[1]	Type of communication third parties in the vicinity of the presentity are unlikely to be able to intercept accidentally or intentionally.	audio
		video
		text
relationship	Type of relationship an alternate contact has with the presentity.	family
		associate
		assistant
		supervisor
service-class	Type of the service offered.	electronic
		delivery
		postal
		in-person
user-input	The user-input or usage state of the service, based on human user input, e.g., keyboard, pointing device or voice	active
		idle
class	The class of the service.	

Next, the following XML example shows the presence information of the presentity someone@example.com who is available at this moment via someone@example.com address email.

```
<?xml version="1.0" encoding="UTF-8"?>
   <presence xmlns="urn:ietf:params:xml:ns:pidf"
   xmlns:rs="urn:ietf:params:xml:ns:pidf:status:rpid-status"
   xmlns:rt="urn:ietf:params:xml:ns:pidf:rpid-tuple"
   entity="pres: someone@example.com"
   xsi:schemaLocation="urn:ietf:params:xml:ns:pidf pidf.xsd
   urn:ietf:params:xml:ns:pidf:status:rpid-tuple rpid-tuple.xsd
   urn:ietf:params:xml:ns:pidf:status:rpid-status rpid-status.xsd">
     <tuple id="fdmy43">
       <status>
         <basic>open</basic>
       </status>
       <et:class>mail</et:class>
       <contact priority="1.0">mailto:someone@example.com</contact>
       <note>I'm in the computer room and you can contact me by
email</note>
```

```
     </tuple>
    </pesence>
```

The following elements can be extended by free-text values or additional IANA-registered values:

- <activities>
- <place-is>
- <place-type>
- <relationship>
- <sphere>

All elements and some attributes are associated with a namespace, which is in turn associated with a globally unique URI. Any developer can introduce their own element names, avoiding conflict by choosing an appropriate namespace.

In case of extending <status> values beyond <basic> it may be sometimes desirable to standardize extension status elements and their semantics. These extensions should be specified under status namespace and must be defined by a standards-track RFC.

A.4. EXAMPLE

The following XML doc represents an integrated example.

The student 'pres:student1@fieldtrip.com' (Field Trip scenario), has a SIP contact address 'sip:student1@archaeology.com', an email address 'mailto:student1@archaeology.com' and an instant messaging address 'im:student1@instantmessag.net.

It has a PDA as device contact with audioconference and speech recognition capabilities.

The student is in the library at school, a quiet place that discourages noise, conversation and other distractions until 5.30 pm GMT. Moreover he is working on the Field Trip project with a mate, sip:student2@archaeology.com, and with the aid of the teacher, 'sip:teacher@archaeology.com', who happen to be available for communications.

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
xmlns:p="urn:ietf:params:xml:ns:pidf:person"
xmlns:d="urn:ietf:params:xml:ns:pidf:device"
xmlns:rs="urn:ietf:params:xml:ns:pidf:status:rpid-status"
xmlns:rt="urn:ietf:params:xml:ns:pidf:rpid-tuple"
xmlns:rp="urn:ietf:params:xml:ns:pidf:rpid-person"
xmlns:rd="urn:ietf:params:xml:ns:pidf:rpid-device"
xmlns:extdcap ="http://mydevide/capabilities/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
entity="pres:student1@fieldtrip.com"
xsi:schemaLocation="urn:ietf:params:xml:ns:pidf pidf.xsd
urn:ietf:params:xml:ns:pidf:status:rpid-device rpid-device.xsd
urn:ietf:params:xml:ns:pidf:status:rpid-tuple rpid-tuple.xsd
```

```
urn:ietf:params:xml:ns:pidf:status:rpid-person rpid-person.xsd
urn:ietf:params:xml:ns:pidf:status:rpid-status rpid-status.xsd
urn:ietf:params:xml:ns:pidf:rpid-tuple rpid-tuple.xsd
http://mydevice/capabilities mydevice-capabilities.xsd">
  <tuple id="t0">
    <status>
      <basic>open</basic>
    </status>
    <et:class>supervisor</et:class>
    <et:relationship>supervisor</et:relationship>
    <contact>sip:teacher@archaeology.com</contact>
    <note>My archaeology teacher</note>
  </tuple>
  <tuple id="t1">
    <status>
      <basic>open</basic>
    </status>
    <et:class>associate</et:class>
    <et:relationship>associate</et:relationship>
    <contact>sip:student2@archaeology.com</contact>
    <note>My mate in Field Trip project</note>
  </tuple>
  <tuple id="t2">
    <status>
      <basic>open</basic>
      <rs:privacy>
        <audio/>
      </rs:privacy>
      <rs:idle since="2005-03-16T10:43:00Z"/>
    </status>
    <et:class>sip</et:class>
    <contact priority="0.8">sip:student1@archaeology.com</contact>
    <timestamp>2004-10-27T16:49:29Z</timestamp>
  </tuple>
  <tuple id="t3">
    <status>
      <basic>open</basic>
      <e:privacy>
        <text/>
      </e:privacy>
      <e:timeoffset>300</e:timeoffset>
    </status>
    <contact priority="0.8">im:student1@instantmessag.net</contact>
    <timestamp>2004-10-27T16:49:29Z</timestamp>
  </tuple>
  <tuple id="t4">
    <status>
      <basic>open</basic>
    </status>
    <et:class>mail</et:class>
    <contact priority="1.0">mailto:student1@archaeology.com</contact>
    <note>I'm in the library working on my Field Trip project</note>
  </tuple>
  <tuple id="t4">
    <status>
      <basic>closed</basic>
    </status>
    <et:service-class>in-person</et:class>
    <note>Closed-door room</note>
```

```
     </tuple>
     <p:person>
       <p:status>
       <rp:activities>
         <rp:school/>
       </rp:activities>
       <rp:class>composed</rp:class>
       <rp:mood>
         <rp:happy/>
         <rp:text>I think we are doing a good project!</rp:text>
       </rp:mood>
       <rp:place-type until="2005-03-16T17:30:00Z">
         <rp:library/>
       </rp:place-type>
       <rp:place-is until="2005-03-16T17:30:00Z">
         <rp:quiet/>
       </rp:place-is>
       <rp:sphere until="2005-03-16T17:30:00Z">
         <rp:work/>
       </rp:sphere>
     </person>
     <d:device>
       <d:status>
       <rd:device-id>
         http://mydevice/id
       </rd:device-id>
       <rd:class>PDA</rp:class>
       <extdcap:input>
         <extdcap:speech/>
       </extdcap:input>
       </extdcap:communication>
         <extdcap:audioconference/>
       </extdcap:communication>
       <rd:text>PDA with speech recognition and audioconference
capabilities</rd:text>
       </rd:extdcap>
     </device>
  </presence>
```