# LIGHT
## (xmL-Innovative Generation for Home networking Technologies)

---

# Deliverable D21: Requirements analysis of the WSED architecture from the SOA's perspective

---

| | |
|---|---|
| Editor(s): | *Vittorio Miori, Luca Tarrini* |
| Document Name: | *Deliverable D21: Requirements analysis of the WSED architecture from the SOA's perspective* |
| Date: | May 23, 2005 |

# Scope

The LIGHT project aims to develop a lightweight middleware for wired or wireless devices computing through the Web Services Architecture. The innovative idea is to bring the paradigm of Web Services Architecture into home environment allowing the definition of an open networking architecture whereby intelligent devices and applications can realize the "Device Co-ordination" concept.

WSED will provide to a device the ability to announce its presence to the network, an automatic discovery of devices-services, the ability to describe its capabilities as well as query-understand of other devices basing solely on XML-based wire protocol. This document attempts to provide some information helping the consortium to choose the technology relevant to LIGHT.

# Executive Summary

This deliverable carries out a significant number of technical studies on existing home networking technologies evaluating the main features. The main purpose of this document is to summarize the results of the evaluation of existing technologies that could be of interest to WSED (Web Services for Embedded Devices) architecture.

The objective of this study has been to identify the focus and capabilities of these technologies in order to decide whether the project should use some of their ideas and solutions for inspiration. Examples of these technologies that could provide the main foundation are UPnP, Web Services, and Jini.

# 1.0 List of evaluated technologies

## *1.1 Universal Plug and Play*

UPnP [1] can be considered a valuable foundation technology for the LIGHT platform. It provides an open, loosely-coupled, service-oriented architecture, and its communication model is based on message-based exchanges without a priori reliance on synchronous communications.

UPnP is designed to support "zero-configuration", "invisible" networking, and automatic discovery for a breadth of device categories from a wide range of vendors. The technologies leveraged in the UPnP architecture include Internet protocols such as IP, TCP, UDP, HTTP, and XML. In the same way in the Internet, service contracts are based on wire protocols that are expressed in XML and communicated via HTTP.

It favours peer-to-peer communications between intelligent devices, but does not preclude client-server communication structures. It only defines the format of information exchanges, it does not define any programming interfaces; consequently, it is open to any host operating system and to any programming language.

It allows for automatic device and service discovery, requiring no human intervention. It provides generic, network-independent device description and service description schemas, from which domain-specific device and service templates can be derived; versioning is designed into these schemas. The figure shows the stack of the UPnP architecture.

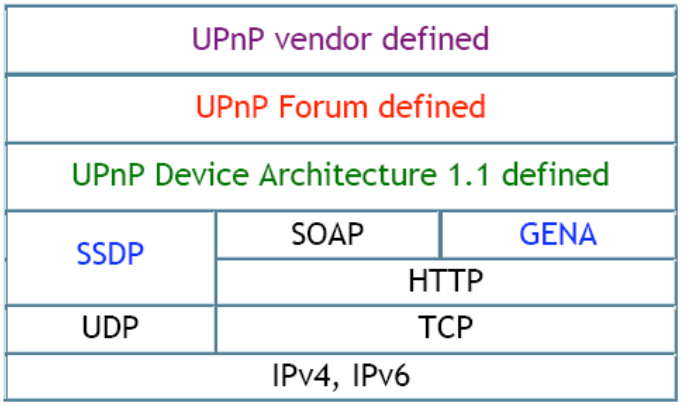


*Figure 1. UPnP v1.1 Architecture*

## 1.2 UPnP and WSED

Given the aims of LIGHT, UPnP is near with the general objectives and can be considered a valuable foundation technology for the WSED architecture. The main are:

- It provides an open, loosely-coupled, service-oriented architecture.
- The UPnP communication is based on message-based exchanges without a priori reliance on synchronous structures.
- It only defines the format of information exchanges, it does not define any programming interfaces; , it is open to any host operating system and to any programming language.
- It allows for automatic device and service discovery, requiring no human intervention.

UPnP provides a device description and service description that it is platform and network independent. Thanks to the extensibility mechanisms of the SOAP protocol, individual higher-layer protocols, whether generic or application-specific, can be improved and versioned in isolation, without affecting the entire stack. This composability property also allows to very easily adapt the protocol stack to a particular application domain. Then UPnP uses IP protocol family, and it is independent of physical networks, whether wired or wireless.

## 1.3 Web Services

What are Web Services?

> "*A Web service is a software application identified by a URI whose interfaces and binding are capable of being defined, described, and discovered by XML artifacts, and supports direct interactions with other software applications using XML-based messages via Internet-based protocols.*" - World Wide Web Consortium -

Web Services [2] target the development of applications based on the XML standard. It is an independent software component supplying a specific set of services. Web services can be used internally by a single application or exposed externally over the Internet for use by a number of applications. This technology provides interoperability between software components that can communicate between different companies and can reside on different infrastructures.

Currently, UDDI, WSDL, and SOAP represents a set of baseline specifications that provide the foundation for application integration. These specifications are the following:

- *UDDI* (Universal Description, Discovery and Lookup) is a specification of a registry for dynamically locating and advertising Web services. It defines white pages (i.e., general information about services), yellow pages (i.e., categorization of services according to standardized taxonomies) and green pages (i.e., technical detail on how to access services such as references to the services' specifications).
- *WSDL* (Web Service Description Language) is a language based on XML that is proposed by the W3C for describing the interfaces of Web services. A WSDL interface then describes how to encode requests and replies for the given Web service.
- *SOAP* (Simple Object Access Protocol) defines a lightweight protocol for information exchange. It sets the rules of how to encode data in XML. It also includes conventions for partly prescribing the invocation semantics (either synchronous or asynchronous) as well as the SOAP mapping to HTTP.

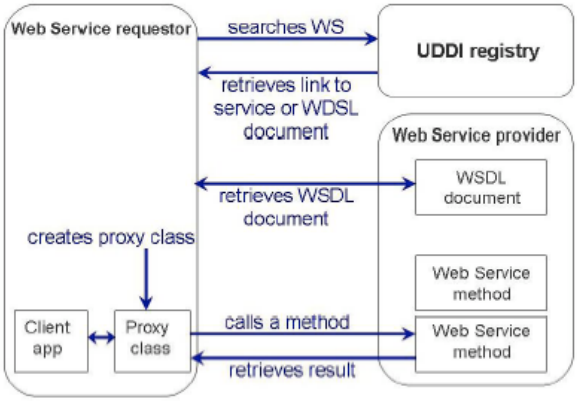


*Figure 3 . Web Service Architecture*

Figure 3 shows an overview of the Web Services Architecture. Based on the SOA-model (Service Oriented Architecture), it consists of three blocks, the Web Service requestor, the Web Service Provider and the UDDI registry.

There already exist platforms that are compliant with the Web Services standards. Even though Web Services is quite recent and not fully mature, it is anticipated that it will play a prominent role in the development of next generation distributed systems.

While the core Web Services technologies (XML, SOAP, WSDL) are ratified standard, the additional Web Services technologies (WS-Addressing, WS-Discovery, WS-Eventing, and so on) are emerging technologies. In fact the baseline specifications require high-level functionality such as security, addressing, reliable messaging, and so on. Web service architecture involves many layered and interrelated technologies. In the figure we provide one illustration of these technologies.

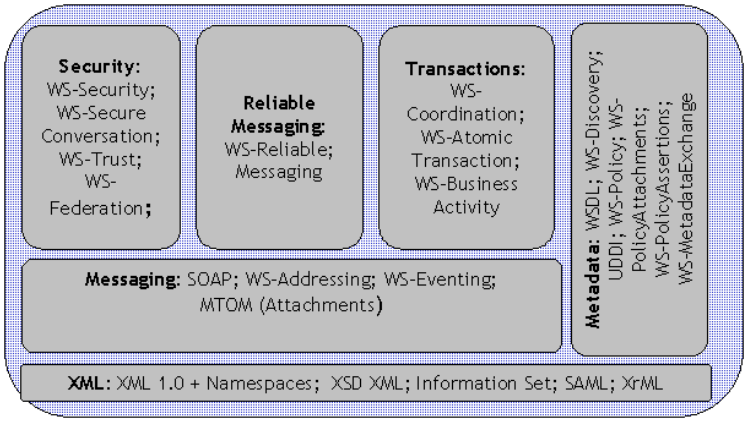


*Figure . Web Services technologies layered view*

The Web Service Architecture provides a set of modular protocol building blocks that can be composed in varying ways to create protocols specific to particular applications. Therefore the idea of *composability* resides in the possibility to combine different specification to provide more powerful capabilities. Composability has and continues to be one of the key design goals for Web services.

## 1.4 Web Services and WSED

It seems that Web Services technologies could be good candidates for the final WSED architecture.

- Web services are based on open standards and leverage Internet protocols and infrastructure.
- Web services are media and network independent.
- Web services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks. They can be developed using any language, and they can be deployed on any platform from the tiniest device to the largest super computer.
- Web Services communicate by passing messages to each other. The Web Service interface adds a layer of abstraction to the environment that makes the connections flexible and adaptable. The Web service implementation is totally hidden from its interface. Hence, when a service implementation is changed at run-time, other services using that service are unaffected.

Web services provide a set of specifications for the addressing, discovery, description, event-oriented messaging and security purposes.

- For the addressing, WS-Addressing provides an interoperable, transport independent approach to identifying message senders and receivers. WS-Addressing also provides a finer grain approach to identifying specific elements within a service that send or should receive a message.
- Web services provide registry and discovery mechanisms: UDDI, and WS-Discovery. Using UDDI-based service discovery is optional, as a service provider can send the description directly to service requesters. WS-Discovery is a specification that defines a multicast protocol for dynamic discovery of services on ad-hoc and managed networks. This specification defines a multicast discovery protocol to locate services. The primary mode of discovery is a client searching for one or more target services.
- Web services provide a service description language (WSDL). WSDL defines the interface and mechanics of service interaction. The WSDL document can be complemented by other service

description documents to describe higher-level aspects of the Web service: business context, Quality of service, and service-to-service relationships. WSDL provides a programmatic way to describe what a service does, paving the way for automation. WSDL eases the definition of abstract services.

- Eventing, WS-Eventing describes how to construct an event-oriented message exchange pattern using WS-Addressing concepts, allowing Web services to act as event sources for subscribers. It defines the operations required to manage subscriptions to event sources, as well as how the actual event messages are constructed. It sets out to define how to implement a solicit-response (i.e. event-based) message exchange pattern for Web services. The idea is to implement on top of the same substrate a range of applications, from simple device-oriented eventing, to Enterprise publish-subscribe systems.
- Web services provide several security specifications. WS-Security specifies enhancements to SOAP messaging aimed at protecting the integrity and confidentiality of a message and authenticating the sender. WS-Security also specifies how to associate a security token with a message, without specifying what kind of token is to be used.

The set of Web services technologies provide a valuable foundation to build distributed applications linking together all types of computing devices, leveraging Internet protocols and infrastructure, using open existing standards and bringing system interoperability and language neutrality. The *composability* principle of the Web services technologies allows a modular approach: only use what is needed, thus, adopt only the set of technologies required by LIGHT goals.

However, Web Services technologies themselves are not sufficient to meet WSED requirements, the immaturity of some specifications, the differing interpretation of the standards and how they relate, causes interoperability problems; that's the reason why the WS-I exists, and deliver Web service profiles. This profile is a step towards advancing interoperability. It clears up simple ambiguities in the current standards and enforces a particular transport, type system, and encoding for compliance with the profile.

Another main reason is that Web services do not bring to computing devices concrete solution, concrete behavior guideline and convention in managed or unmanaged network environment. Here, UPnP is the solution. UPnP provides Plug and Play mechanism, supports zero-configuration, invisible networking and automatic discovery for devices. Both UPnP and Web Services share the same underlying basic concepts: standard-based, platform and programming language independence, leverage Internet protocols, loosely coupled and so on.

Web services technologies constitute a valuable foundation technology for LIGHT. In addition to such a base technology, it must be noted that LIGHT addresses mobile and embedded applications. This requirement not only constrains processing and memory resources, but it also creates the need for a time-sensitive protocol on top of the Web services profile.

## 1.5 Jini

Jini [3] uses the term *service* explicitly to refer to some entity on the network that can be used by other Jini participants providing the infrastructure that allows these services to find and use one another on a network. Jini allows services to come and go without requiring any static configuration or administration. This means that you do not have to edit configuration files, configure gateways, or anything else to use a Jini service. In other words, Jini provides Plug and Play capabilities.

Jini services join together in cooperation sets called *communities*. All services in a community are aware of each other and able to use each other. Jini is designed to accommodate varying numbers of services, from the very large to the very small. Jini addresses scalability through *federation.* This is the ability for Jini communities to be linked together into larger groups.

Jini is designed to support a wide variety of entities that can participate in a Jini community. These entities may be devices or software or some combinations of both. Jini is flexible about how much computational power a hardware device needs to have. However, devices need to be capable of running JVMs. Nowadays, this is not an obstacle as Jini can also work with such devices as PDAs and cell phones that may have only limited Java capabilities (Java 2 Micro Edition). Furthermore, Jini doesn't even require that the device or service is written in or understands Java, all that is need is that some Java-speaking device is willing to act as a proxy on behalf of the Java-challenged device or service.

From what we have been discussing so far, we can conclude that Jini provides a very powerful platform for the spontaneous networking of devices and services. Devices and services can go up or down, or change their location in the network without affecting the whole system. There is no need for human administrators. All in all, Jini simplifies the task of building and maintaining a network of devices, software and users.

## 1.6 Jini and WSED

This section discusses whether Jini technology fulfils or not the requirements for the WSED platform. Let's revisit its advantages first:

- Jini has been designed to allow spontaneous networking of devices and services with zero configuration.
- Jini provides plenty of built-in functionality (leasing, event notification, transactions, security, configuration, exporter and more).

However, there are some disadvantages too, or requirements that are not fulfilled.

- Although in theory Jini is only tied to RMI and Jini protocols, in practice we can say that Jini is tied to Java, while LIGHT claims for a language independent solution.

# 2.0 Reference

[1] UPnP. http://upnp.org

[2] Web Services Architecture. http://www.w3.org/2002/ws/

[3] Jini. http://www.jini.org