WS4D: SOA-Toolkits making embedded systems ready for Web Services

Elmar Zeeb[1], Andreas Bobek[1], Hendrik Bohn[1], Steffen Prüter[1], Andre Pohl[2], Heiko Krumm[2], Ingo Lück[3], Frank Golatowski[1], and Dirk Timmermann[1]

[1] Institute of Applied Microelectronics and Computer Engineering, University of Rostock (elmar.zeeb, andreas.bobek, hendrik.bohn, steffen.prueter, frank.golatowski, dirk.timmermann)@uni-rostock.de
[2] Dpt. CS, Computer Networks and Distributed Systems, University of Dortmund (krumm, pohl)@ls4.cs.uni-dortmund.de
[3] Materna Information & Communications, Dortmund, Germany ingo.lueck@materna.de

Abstract. The usage of the Service Oriented Architecture (SOA) paradigm currently changes the view on many enterprise applications. SOA allows the creation of modular and clearly defined software architectures that ensure a high grade of interoperability and reusability. As even small, resource-constraint networked devices get more and more powerful it is common sense to try to adopt the SOA paradigms to embedded device networks. This idea is substantiated in the specification of the Devices Profile for Web Services (DPWS), a standard that uses the primitives of the Web Services Architecture (WSA) to create a framework for interoperable and standardized communication between embedded devices. This paper introduces the WS4D initiative, a project that tries to provide a common open source platform for using DPWS in different environments. As a basis, we have developed three DPWS stacks that will be released as open source.

Key words: WS4D, SOA, Open Source, Web services, DPWS, Embedded systems

1 Introduction

The increasing complexity of device networks consisting of up to thousands of devices is demanding new technologies for simple device interaction and interoperability. Service-Oriented Architectures (SOA) [DJMZ05] firstly addressed this issue for software components where Web services [Wor04] have achieved the highest market penetration. SOAs describe standards for the description, integration, announcement, discovery and usage of components and their functionality (*services*) in a network.

In 2004 a first proposal for the Devices Profile for Web Services (DPWS) was announced [Mic06]. DPWS enables devices being compatible to Web services

and is part of the current Microsoft Windows platform Vista. The European R&D ITEA project SIRENA (Service Infrastructure for Real-time Embedded Networked Applications) ([SIR06, BBG06]) has developed some of the first DPWS software toolkits for embedded systems worldwide covering a wide range of platforms and application areas.

When the SIRENA project ended in the beginning of 2006 the WS4D initiative was founded by some partners to follow up the development of the different DPWS toolkits, to obtain their interoperability and to bring them to open source.

This paper introduces the Web Services for Devices (WS4D) initiative, their developments and common goals. It is organised as follows: After giving an overview of DPWS in section 2 and looking at some related work in section 3, section 4 presents an introduction to the WS4D initiative. The toolkits mentioned before are described in section 5. The paper ends with some conclusions in section 6.

2 Devices Profile for Web Services

The *Devices Profile for Web Services (DPWS)* was developed to enable secure Web service capabilities on resource-constraint devices [Mic06]. It features secure exchange of messages with Web services, dynamic discovery and description of Web services, and subscribing to, and receiving events from a Web service. DPWS can be used for inter machine communication. However, the latter requires the devices to have an implemented peer functionality, a specific DPWS client implementation, to use a correspondig service hosted on another device.

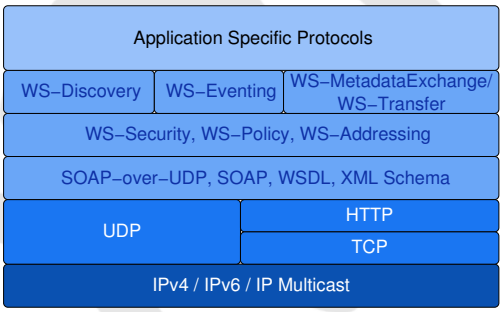

Fig. 1. The Devices Profile for Web Services protocol stack

As shown in Fig. 1 DPWS bases on well known protocols and several Web service specifications. It employs similiar messaging mechanisms as the Web Services Architecture (WSA) with restrictions to complexity and message size ([Mic06, ZBBG07]). On top of the low level communication foundations like IP- Uni- and Multicasting, TCP and HTTP it uses SOAP-over-UDP, SOAP, and

XML Schema for the actual information exchanges. WS-Policy, WS-Addressing and WS-Security are on top of the messaging layer. *WS-Policy* is used to exchange and negotiate policies and parameters required for service usage. *WS-Addressing* separates the SOAP messaging layer from its close binding to HTTP as defined in the SOAP specification. It introduces the concepts of message information headers and endpoint references making service users, providers and transmitted messages uniquely identifiable and addressable. The *WS-Security* specification defines mechanisms for secure communication leveraging standards like XML-Encryption, XML-Signature and Secure Sockets Layer (SSL).

DPWS specifies further mechanisms for ad-hoc device discovery, device and service description and eventing. Ad-hoc device discovery is based on *WS-Discovery*, SOAP-over-UDP and IP-Multicast. Devices can advertise their services to the network and clients can probe a network for specific devices. The devices describe their characteristics and capabilities (in form of services hosted by the device) using the *Web Service Description Language* (WSDL) – as known from the WSA – which can be used by service clients to identify and bind to particular service interfaces. It can also be used to find out where the services actual communication endpoints reside. Finally, the DPWS also contains a publish-subscribe mechanism (*WS-Eventing*) for services acting as an event source and sending events to subscribed clients.

3 Related work

Microsoft as one of the authors of the DPWS specification ships its latest Windows version with an implementation of the DPWS protocol stack, which is called *WSDAPI*. This API is part of the new PNP-X subsystem which allows locally installed devices and such attached through the network (e.g. by UPnP or DPWS) to be accessed and used in a uniform way [Mic07].

Furthermore, Schneider Electric (F) – project coordinator of the SIRENA project – has developed the first implementation of a DPWS stack for embedded devices in the SIRENA project [JMS05]. The WS4D initiative is in contact with Schneider Electric to assure compatibility. The SIRENA follow-up project – Service Oriented Device and Delivery Architecture (SODA) – is currently developing a toolkit around the Schneider stack to improve manageability, orchestration and security [SOD07].

4 The WS4D Initiative

The *Web Services for Devices (WS4D)* [WS407] initiative was established by academic and industrial partners in the middle of 2006 for several reasons. Before WS4D all partners were actively involved in the award-winning European R&D project SIRENA which ended in spring-time 2006. One of the main challenges in SIRENA was establishing a software infrastructure in which embedded, networked devices can be integrated. The devices should be self-contained

and able to communicate with each other. As one result some prototypes implementing earlier versions of DPWS were developed.

The WS4D initiative can be considered as a non-profit follow-up project to preserve and extend the SIRENA results and also to maintain the collaboration between the partners. But the main aspect which is followed by deploying WS4D is building up an open community that actively participates in further development processes:

1. Improving available stacks by providing them as *open source* software on an open platform including bug tracking and documentation issues,
2. Developing *test suites*, and
3. Promoting *standardization* process, e.g. delivery of standardized devices and basic services such as management services.

By putting these items into action we expect interoperable stack solutions.

Currently, the Web services protocol family comprises more than 40 specifications which again make extensive use of the XML protocol family. Next to these main standards there are binding protocols which combine WS protocols with other specifications (e.g. transport) or with themselves. The plethora of Web services protocols, the participation of many different initiatives, the still evolving standardization process, redundancy between and inconsistency within these protocols result in interoperability problems.

Profiles are means for restricting certain specifications and setting up a working subset of given protocol families (q.v. profiles in the Bluetooth protocol family). In this respect DPWS can be considered as a first step in developing an interoperable specification for SOA-enabled devices. However this is a first step and real interoperability can not be achieved until the profile is practically applied.

In this context WS4D considers to be the second step. In a nutshell, the overall objective of WS4D is ensuring interoperability between various implementations of the Devices Profile on different platforms and different programming languages by the help of an active and open community.

5 Toolkits

In the following sections the three WS4D Web services development toolkits are introduced: WS4D-gSOAP, WS4D-Axis2 and WS4D-JavaME. These toolkits will be published under LGPL soon and will be available on the WS4D website [WS407].

The WS4D-gSOAP and WS4D-Axis2 stacks are maintained by the University of Rostock and the WS4D-JavaME stack is maintained by the University of Dortmund and Materna Information & Communications [Mat07].

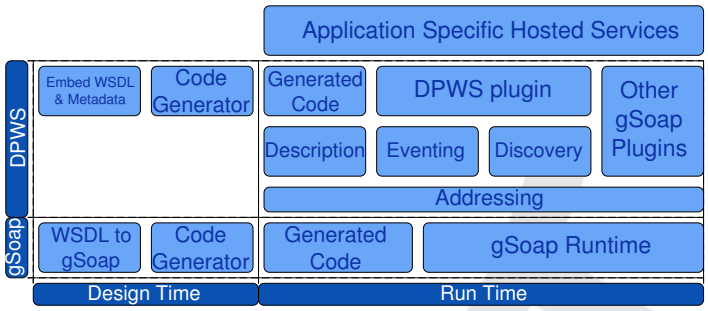

Fig. 2. WS4D-gSOAP toolkit

5.1 WS4D-gSOAP

WS4D-gSOAP is an extension of the well known gSOAP Web services toolkit, a toolkit for building SOAP-based Web services with C/C++ developed by Robert A. van Engelen [vE07]. It is designed to develop small footprint and high throughput Web services. The toolkit consists of a development and a runtime environment.

gSOAP offers code generation tools for implementing Web services. gSOAP has defined its own service description language that is based on C syntax. This description is saved in special gSOAP files that are similar to C header files with annotations. To complete the Web services design flow the toolkit also includes a tool to translate WSDL files into gSOAP files.

The second part of the development environment is the gSOAP code generator. It generates XML schema to C data binding as well as stub and skeleton code for a specific gSOAP service description. The XML schema to C data binding creates a mapping from every type of the used XML schema definitions to a C type structure and generates functions for the marshalling and demarshalling. The skeleton and stub code generator finally maps WSDL operations to C functions.

The runtime part of gSOAP consists of the generated code and the gSOAP runtime. The gSOAP runtime consists of functions for the service developer and functions used in the generated code.

WS4D-gSOAP uses a similar workflow as gSOAP (see Fig. 3). To create a DPWS device a developer has to specify a WSDL description of the services on a device and the device's metadata. The WSDL files are used for code generation in gSOAP's typical way as described in the last paragraph. The device metadata is used to generate code for service setup and assignment of model metadata and device characteristics. With the resulting code a developer can concentrate just on the implementation of the functionality of the services hosted by a device.

As shown in Fig. 2 the WS4D-gSOAP toolkit uses gSOAP's plug-in mechanism to implement WS-Addressing, WS-Discovery, WS-MetadataExchange / WS-Transfer and WS-Eventing on top of gSOAP. WS4D-gSOAP supports three different roles for an endpoint implementation that can be switched at compile

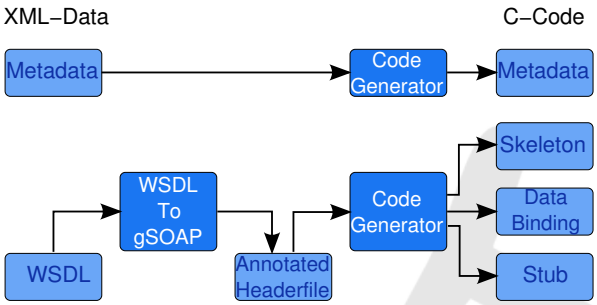

Fig. 3. WS4D-gSOAP code generation

time: device, client and peer. With the device role an endpoint implements the device side of the specification. The client role is used to create code for a Web service client, respectively. The peer role has to be used when both client and device are about to be integrated in one application.

WS4D-gSOAP offers multi-platform support such as the Linux i386, Windows-native, Windows-cygwin and embedded Linux (FOX Board [Acm07] and Nokia Maemo [Mae07]) platforms. To develop devices a typical GNU software development toolchain can be used. Developers preferring integrated development environments can use Visual Studio 8.0 on Windows or Eclipse on other platforms.

5.2 WS4D-Axis2

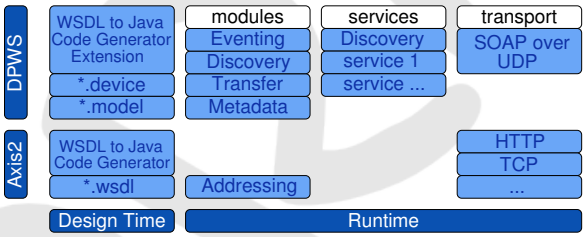

Fig. 4. Axis2 Architecture and DPWS Extensions

The second DPWS stack is based on Axis2 being the successor of Axis – a Java-based SOAP processor for Web services [Apa07]. As with all projects of the Apache group Axis2 is also available as an open source software with an Apache licence.

Axis2 supports SOAP 1.1 as well as SOAP 1.2, offers several transport connectors such as HTTP, TCP, JMS and SMTP and comes with a WS-Addressing enabling module. So by default asynchronous message exchange is supported.

Axis2 is a modular built SOAP stack in which implementations of additional Web services specifications can be easily plugged in via modules.

By using Axis2 we can bridge the two worlds of embedded devices and application development at the enterprise level without abandoning existing solutions and customs: Axis2 runs on J2SE, an Eclipse plug-in is available for code creation, and further lots of plug-ins implementing other WS specifications such as WS-Security, WS-ReliableMessaging, WS-Coordination and WS-AtomicTransaction are available or on the way.

Since the engine is Java-based several platforms (Windows, Linux etc.) are targeted. Services are deployable in standalone mode or under servlet container (default) such as Tomcat.

WS4D-Axis2 is a stack solution on top of Axis2 as depicted in Fig. 4 for writing primarily clients for controlling DPWS enabled devices, e.g. for management tasks. WS4D-Axis2 comes with several modules (plug-ins) which are independent of each other: SOAP over UDP, Discovery, Eventing and DPWS. It offers three main APIs for searching for, subscribing to and controlling devices.

Since WS4D-Axis2 runs on J2SE it is not targeted for embedded devices, but it can be used for device management, discovery proxies and other rich client implementations.

5.3 WS4D-JavaME

The Java 2 Micro Edition is targeted at small and resource constraint devices such as mobile phones and PDAs (e.g. running the *Mobile Information Device Profile (MIDP)*) [Sun07]. Additionally, it defines an environment (*Connected Device Configuration*) for more capable devices like set-top boxes or other high-end embedded devices. The *WS4D-JavaME* stack is based on the *Connected Limited Device Configuration*, the smallest subset of JavaME configurations and can thus be used on all JavaME platforms and even on Java 2 Standard editions using platform dependent toolkits.

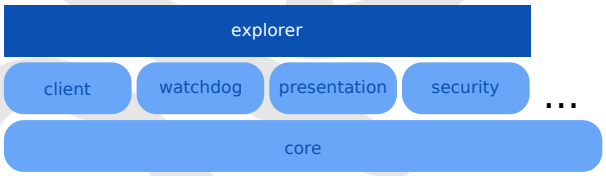

Fig. 5. Modules of the WS4D-JavaME stack

The main packaging of the stack is depicted in Fig. 5. The `core` package contains all code that is necessary to create a DPWS device with one or more services. This device is discoverable via WS-Discovery[Mic05] mechanisms and can be used as an event-source. All packages on top of the core package add additional functionality, which is not mandatory for normal, self-contained DPWS

devices. The `client` package adds functionality to find remote devices and services in the network and to invoke the operations defined in the WSDL. Adding this module permits a service to act as a client and to use remote services. The `watchdog` module contains a software watchdog that can be used to check for dead threads and service processings. The `presentation` package subsumes the classes that are used to create a rich featured presentation URL for the device which can be controlled with any web-browser like Firefox or Opera and features the complete usage of deployed devices and services as looking at the device's metadata or invoking service operations. The last module worth to be mentioned in this article is the `security` module. This module implements the subset of security features of WS-Security specified in the DPWS specification. It uses the cryptographic API of the Java 2 Standard Edition. Thus security is currently not available for Java Micro Edition Devices.

On top of this framework we have implemented a testing and debugging tool called *DPWS-Explorer* which can be used with any DPWS featured device. The DPWS-Explorer is able to listen for devices' discovery messages (`Hello`, `Bye`) and is able to actively search for devices. If a device is discovered, the tool gets the metadata information from the device and creates a user interface for accessing and using the hosted services' operations. The usage of the eventing mechanisms is also supported. Additionally one can log all network conversation for later analysis.

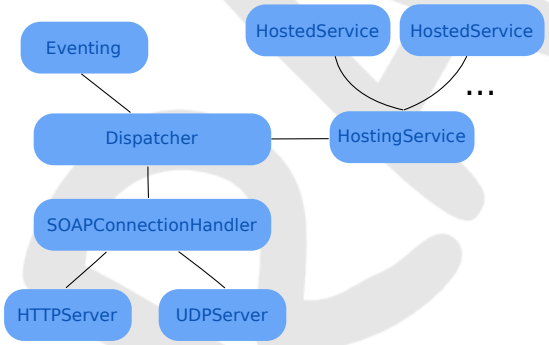

Fig. 6. The WS4D-JavaME architecture

The architecture of the framework is shown in Fig.6. The framework communicates via the `HTTPServer` and `UDPServer` classes with the external network. All incoming traffic is passed to the `SOAPConnectionHandler` which parses and analyses the content of incoming messages. On successful completion the gathered data is forwarded to the `Dispatcher` which in turn dispatches the message to the targeted service or device.

Creating a new device with the WS4D-JavaME framework is a bit different compared to the other implementations. Most frameworks use a code generator to create the code frame for the device implementation that at least requires

the WSDL of the services to be hosted by the device. In the WS4D-JavaME framework one does not have to deal with WSDLs or device descriptions necessarily, as you create your device only by programming in the framework. One can create a new service by extending the `HostedService` class, adding operations with arbitrary parameters to this service and finally add an instance of this service to an extended `HostingService` instance. The services' WSDL and the device metadata are generated on demand, when a client asks for it. The `client` package also includes the code to create proxy objects for remote services from a given WSDL. Thus the framework is very dynamic and DPWS devices and services can be built-up on-the-fly at runtime.

6 Conclusion and future work

In this paper we introduced the WS4D initiative, i.e. its purpose, the tools which are currently in progress and upcoming challenges. The main objective is building up an open community which practically applies the Devices Profile for Web Services to attain interoperability.

We will further foster the standardization process. One of the next tasks will be a proposal for device templates. Such template system could standardize certain device types and provide similar advantages (e.g. easy code generation) like device templates in the UPnP technology.

Acknowledgments

This work has been funded by German Federal Ministry of Education and Research (BMBF) under reference number 01—SF11H.

References

[Acm07] Acme Systems. `http://www.acmesystems.it`, 2007.

[Apa07] Apache Axis2 / Java. `http://ws.apache.org/axis2/`, 2007.

[BBG06] H. Bohn, A. Bobek, and F. Golatowski. SIRENA - Service Infrastructure for Real-time Embedded Networked Devices: A service oriented framework for different domains. In *International Conference on Networking (ICN)*, 2006.

[DJMZ05] W. Dostal, M. Jeckle, I. Melzer, and B. Zengler. *Service-orientierte Architekturen mit Web Services*. Elsevier, 2005.

[JMS05] F. Jammes, A. Mensch, and H. Smit. Service-oriented device communications using the devices profile for web services. In *3rd International Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC05) at the 6th International Middleware Conference*, 2005.

[Mae07] Maemo: Development Platform for Nokia Internet Tablet Products. `http://www.maemo.org`, 2007.

[Mat07] Materna Information & Communications. `http://www.materna.com`, 2007.

[Mic05] Microsoft. *Web Services Dynamic Discovery (WS-Discovery)*, 2005. `http://schemas.xmlsoap.org/ws/2005/04/discovery/`.

[Mic06] Microsoft, Intel, Ricoh, Lexmark. *Devices Profile for Web Services*, 2006. `http://schemas.xmlsoap.org/ws/2006/02/devprof/`.

[Mic07] Microsoft Rally. `http://www.microsoft.com/rally`, 2007.

[SIR06] SIRENA: Service Infrastructure for Real-time Embedded Networked Applications. `http://www.sirena-itea.org`, 2006.

[SOD07] SODA consortium. *SODA - Technical Framework Description*, 2007. `http://www.soda-itea.org/Documents/AllDocuments/`.

[Sun07] Sun Microsystems. *Java 2 Micro Edition*, 2007. `http://java.sun.com/javame/index.jsp`.

[vE07] R. A. van Engelen. *gSOAP*, 2007. `http://www.cs.fsu.edu/~engelen/soap.html`.

[Wor04] World Wide Web Consortium (W3C). *Web Services Architecture*, 2004.

[WS407] WS4D: Web Services for Devices. `http://www.ws4d.org`, 2007.

[ZBBG07] E. Zeeb, A. Bobek, H. Bohn, and F. Golatowski. Service-oriented architectures for embedded systems using devices profile for web services. In *2nd International IEEE Workshop on SOCNE07*, 2007.