Diploma Project

Manage Telecommunication equipment using Web Services

Web Service

Version V 1.1 (WS)

Professeurs:
Philippe Joye
François Buntschu

Mandatory:
Daniel Gachet

Expert:
Nicolas Mayencourt

Students:

Thierry Kiki
David Schneider

Table of Contents

1 Definitions

SOA:	Service Oriented Architecture
Web Services:	W3C recommendation
XML:	Extensible Markup Language
WSDL:	Web Service Description Language (W3C recommendation)
SOAP:	Simple Object Access Protocol (W3C recommendation)
UDDI:	Universal Description, Discovery and Integration (directory service)
W3C:	World Wide Web Consortium
OASIS:	Organization for the Advancement of Structured Information Standards
IETF:	Internet Engineering Task Force
WS-I:	Web Services Interoperability Organization

2 Initiation

2.1 Traditional web interaction

USER (Browser) ↔ Web Application
HTML-file

2.2 Web Service interaction

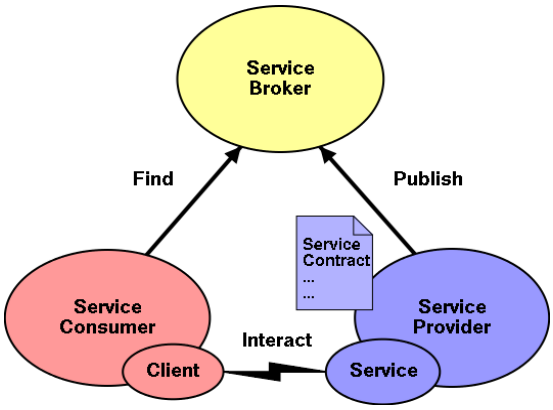
Application ↔ Web Service
XML

3 Service Oriented Architecture

2 words:

World Wide Web (WWW) + Service Oriented Architecture (SOA)
= Web Service Architecture

3.1 Service Oriented Architecture



Source: www.w3.org/2003/Talks/0521-hh-wsa/soa.png

3.1.1 Service Broker (optional)

...

3.1.2 Service Consumer

...

3.1.3 Service Provider

...

3.2 Basic characteristics of a SOA

Loose coupling:

Services are discovered automatically, found and binded by the application.

…...

4 Web Service Architecture

4.1 Definition

The definition of W3 published in the Web Services Architecture Requirements:[1]

« A Web service is a software system identified by a URI [RFC 2396], whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols. »

4.2 Basic Concept

The basic components of a SOA are:

- Communication
- Service Description
- Directory Service

The W3 recommends for the communication of Web Services the use of SOAP, its specification defines the XML-based message format and how it is embedded into a transport protocol. SOAP is mostly transported over HTTP but is not at all dependent on this transport protocol.

WSDL, also XML-based, is used to describe the Web Service.

The directory service specifies a standardized structure for the management of Web Service metadata. This service is optional.

4.3 Standardization

W3C: ...

OASIS: ...

IETF: ...

WS-I: ...

1 Source: http://www.w3.org/TR/wsa-reqs/

5 Web Service Description

In the following paragraphs we will introduce the Web Service Description Language WSDL and link it the interface of the MileGate we want to describe.

5.1 Introduction

WSDL is an XML language for describing Web Service interfaces. The language is standardized by the W3C.
The specification of the version 1.1 exists since 2001. The follower version (2.0) reached the status of a 'W3C Recommendation' in march 2006 but most of the current Web Services still use the previous version.

The description of the service is spitted into two parties, we have an abstract and a concrete description. The abstract view focuses on the functionality and the concrete enters more into the technical description. Like this we have a separation between the technical details and the manner our service is offered.

The components of the description are:

Abstract: Operation
Messages Exchang Pattern
Interface

Concrete: Binding
Endpoint
Service

The main difference of WSDL according to other description languages for interfaces (e.g. IDL, Interface Description Language) is that everything is concentrated in one file. We are able to communicate with the service just on the base of the WSDL file. Off course we have also the possibility to write the description modular (include, import) to provide better legibility and maintainability.

5.2 Structure of the description

We want to introduce briefly the elements used to describe the Web Service and show afterwards a few more details using the description of our interface.
If two elements are used to describe one single element, this is due to the different versions of WSDL. The first element belongs to the version 1.1 and the second to the standard 2.0.

definitions / description (root element)
This XMLelement represents the root element of the WSDL file and defines the different name spaces.

```
<definitions name="mob_mainbase"
   xmlns:mob_mainbase_xml="http://www.keymile.com/milegate/ws/mob_mainbase_xml"
   xmlns:mob_mainequipment_xml="http://www.keymile.com/milegate/ws/
mob_mainequipment_xml"
   xmlns:mgws="http://www.keymile.com/milegate/ws/ws-common"

   xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap12/"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsman="http://schemas.xmlsoap.org/ws/2005/06/management"
   targetNamespace="http://www.keymile.com/milegate/ws/mob_mainbase_xml">
```

documentation
The section documentation contains a textual annotation to the service.

```
<!-- === DOCUMENTATION === -->

   <documentation>
      -textual description of Web Service
      -further infos for the use of this service or interface
      -contact person
   </documentation>
```

types
Defines the usable data types.

```
<!-- === TYPES === -->

   <types>
      <xs:schema  xmlns="http://www.keymile.com/milegate/ws/mob_mainbase_xml"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
                  elementFormDefault="qualified"
                  targetNamespace="http://www.keymile.com/milegate/ws/mob_mainbase_xml">

         <xs:element name="Info2" type="Info2__Type"/>
         <xs:complexType name="Info2__Type">
            <xs:sequence>
               <xs:element name="moType">
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:maxLength value="63"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element name="adfReference">
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:maxLength value="63"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element name="addressFragment">
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:maxLength value="80"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element name="moName">

               .
               .
```

XMLSchema base data types

```
        <xs:element name="Label" type="Label__Type"/>
        <xs:complexType name="Label__Type">
           <xs:sequence>
              <xs:element name="user">
                 <xs:simpleType>
                    <xs:restriction base="xs:string">
                       <xs:maxLength value="63"/>
                    </xs:restriction>
                 </xs:simpleType>
              </xs:element>
              <xs:element name="service">
                 <xs:simpleType>
                    <xs:restriction base="xs:string">
                       <xs:maxLength value="63"/>
                    </xs:restriction>
                 </xs:simpleType>
              </xs:element>
              <xs:element name="description">
                 <xs:simpleType>
                    <xs:restriction base="xs:string">
                       <xs:maxLength value="127"/>
                    </xs:restriction>
                 </xs:simpleType>
              </xs:element>
           </xs:sequence>
        </xs:complexType>

        <xs:element name="AlarmSeverity" type="AlarmSeverity__Type"/>
        <xs:complexType name="AlarmSeverity__Type">
           <xs:sequence>
              <xs:element name="maxAlarmSeverity" type="severity__Type"/>
              <xs:element name="maxPropagatedAlarmSeverity" type="severity__Type"/>
           </xs:sequence>
        </xs:complexType>

     </xs:schema>
  </types>
```

message

This element contains the possible messages and the types which are allowed to use.

```
<!-- === MESSAGES === -->

   <!-- WS-Management headers  -->
   <message name="ResourceURIMessage">
      <part name="Header" element="wsman:ResourceURI"/>
   </message>
   <message name="SelectorSetMessage">
      <part name="Header"  element="wsman:SelectorSet"/>
   </message>

   <!-- WS-Addressing headers  -->
   <message name="ToMessage">
      <part name="Header"  element="wsa:To"/>
   </message>
   <message name="ReplyToMessage">
      <part name="Header"  element="wsa:ReplyTo"/>
   </message>
   <message name="ActionMessage">
      <part name="Header"  element="wsa:Action"/>
   </message>
   <message name="MessageIDMessage">
```

```
   <part name="Header"  element="wsa:MessageID"/>
</message>

<!-- bodys  -->
<message name="Info2__Message">
   <part name="Body" element="mob_mainbase_xml:Info2"/>
</message>
<message name="Discover__Message">
   <part name="Body" element="mob_mainbase_xml:Discover"/>
</message>
<message name="Label__Message">
   <part name="Body" element="mob_mainbase_xml:Label"/>
</message>
<message name="AlarmSeverity__Message">
   <part name="Body" element="mob_mainbase_xml:AlarmSeverity"/>
</message>
```

Reference to data type (TYPES)

port type / interface

Describes the interfaces and the provided operations on this interface. For each operation the corresponding input and output messages are listed.

```
<!-- === PORT TYPES === -->

   <portType name="main_base__PortType">
      <operation name="GetLabel__Operation">
         <input message="mgws:EmptyMessage"/>
         <output message="mob_mainbase_xml:Label__Message"/>
      </operation>
      <operation name="SetLabel__Operation">
         <input message="mob_mainbase_xml:Label__Message"/>
         <output message="mob_mainbase_xml:Label__Message"/>
      </operation>
      <operation name="GetAlarmSeverity__Operation">
         <input message="mgws:EmptyMessage"/>
         <output message="mob_mainbase_xml:AlarmSeverity__Message"/>
      </operation>
      <operation name="GetInfo2__Operation">
         <input message="mgws:EmptyMessage"/>
         <output message="mob_mainbase_xml:Info2__Message"/>
      </operation>
      <operation name="GetDiscover__Operation">
         <input message="mgws:EmptyMessage"/>
         <output message="mob_mainbase_xml:Discover__Message"/>
      </operation>
   </portType>
```

The input message (view of Service) doesn't have any body!

binding

With the element binding we declare which transport protocol is used for which interface. For inputs or outputs of operations we need to assign the messages to the elements of the transport protocol (for the example SOAP, this will be SOAP:body or SOAP:header)

```
<!-- === BINDING === -->

   <binding name="main_base__Interface" type="mob_mainbase_xml:main_base__PortType">
      <soapbind:binding style="document" transport="http://schemas.xmlsoap.org/soap/
http"/>

      <operation name="GetLabel__Operation">
         <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/
Get"/>
         <input>
            <soapbind:header message="tns:ResourceURIMessage" part="Header"
use="literal" />
            <soapbind:header message="tns:SelectorSetMessage" part="Header"
use="literal" />
            <soapbind:header message="tns:ToMessage" part="Header" use="literal" />
            <soapbind:header message="tns:ReplyToMessage" part="Header" use="literal" />
            <soapbind:header message="tns:ActionMessage" part="Header" use="literal" />
            <soapbind:header message="tns:MessageIDMessage" part="Header"
```

Definition of SOAP transport protocol and style

```
use="literal" />
            <soapbind:body use="literal"/>
         </input>
         <output>
            <soapbind:body use="literal"/>
         </output>
      </operation>

      <operation name="SetLabel__Operation">
         <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/
Set"/>
         <input>
            <soapbind:header message="tns:ResourceURIMessage" part="Header"
use="literal" />
            <soapbind:header message="tns:SelectorSetMessage" part="Header"
use="literal" />
            <soapbind:header message="tns:ToMessage" part="Header" use="literal" />
            <soapbind:header message="tns:ReplyToMessage" part="Header" use="literal" />
            <soapbind:header message="tns:ActionMessage" part="Header" use="literal" />
            <soapbind:header message="tns:MessageIDMessage" part="Header"
use="literal" />
            <soapbind:body use="literal"/>
         </input>
         <output>
            <soapbind:body use="literal"/>
         </output>
      </operation>
      <operation name="GetDiscover__Operation">
         <soapbind:operation soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/
Get"/>
         <input>
            <soapbind:header message="tns:ResourceURIMessage" part="Header"
use="literal" />
            <soapbind:header message="tns:SelectorSetMessage" part="Header"
use="literal" />
            <soapbind:header message="tns:ToMessage" part="Header" use="literal" />
            <soapbind:header message="tns:ReplyToMessage" part="Header" use="literal" />
            <soapbind:header message="tns:ActionMessage" part="Header" use="literal" />
            <soapbind:header message="tns:MessageIDMessage" part="Header"
use="literal" />
            <soapbind:body use="literal"/>
         </input>
         <output>
            <soapbind:body use="literal"/>
         </output>
      </operation>
   </binding>
```

Each operation has its transfer function (soapAction)

Each operation has its SOAP header and body

No need for interpretation. Passes to application as a full XML

service

Describes where the service is located. 'Services' can be subdivided into 'ports' with different addressing parameters

```
<!-- === SERVICE === -->

   <service name="MileGateService">

      <port name="LabelPort" binding="main_base__Interface">
         <soapbind:address location="http://localhost:9357/wsman" />
         <wsa:EndpointReference
         xmlns:wsa="http://www.w3.org/2005/08/addressing"
         xmlns:wsaw="http://www.w3.org/2006/02/addressing/wsdl"
         xmlns:wsdli="http://www.w3.org/2006/01/wsdl-instance">
            <wsa:Address>http://localhost:9357/wsman</wsa:Address>
            <wsa:Metadata>
               <wsaw:InterfaceName>main_base__PortType</wsaw:InterfaceName>
            </wsa:Metadata>
            <wsa:ReferenceParameters>

<wsman:SelectorSet>
                  <wsman:Selector Name="mf">main</
wsman:Selector>
                  <wsman:Selector Name="property">www.keymile.com/mg/2008/06/MoInfo/
Lable</wsman:Selector>
               </wsman:SelectorSet>
               <wsman:ResourceURI>/unit-11</wsman:ResourceURI>
            </wsa:ReferenceParameters>
         </wsa:EndpointReference>
      </port>
```

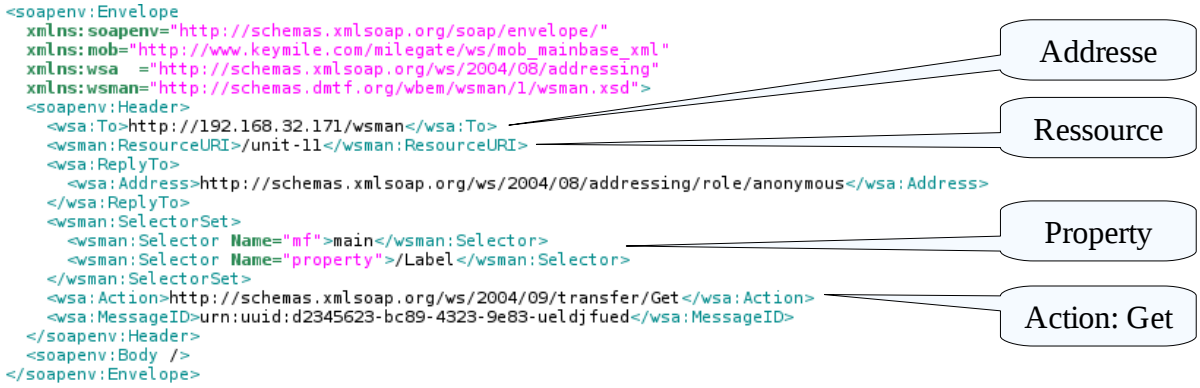
Definition of Service Endpoint with the addressing parameters

.

5.3 SOAP Message

The SOAP message defined in the Web Service Description File helps a lot to understand the descrtiption.
We are going to represent the SOAP messages for the Set- and GetLabel operation.

GetLabel SOAP message:

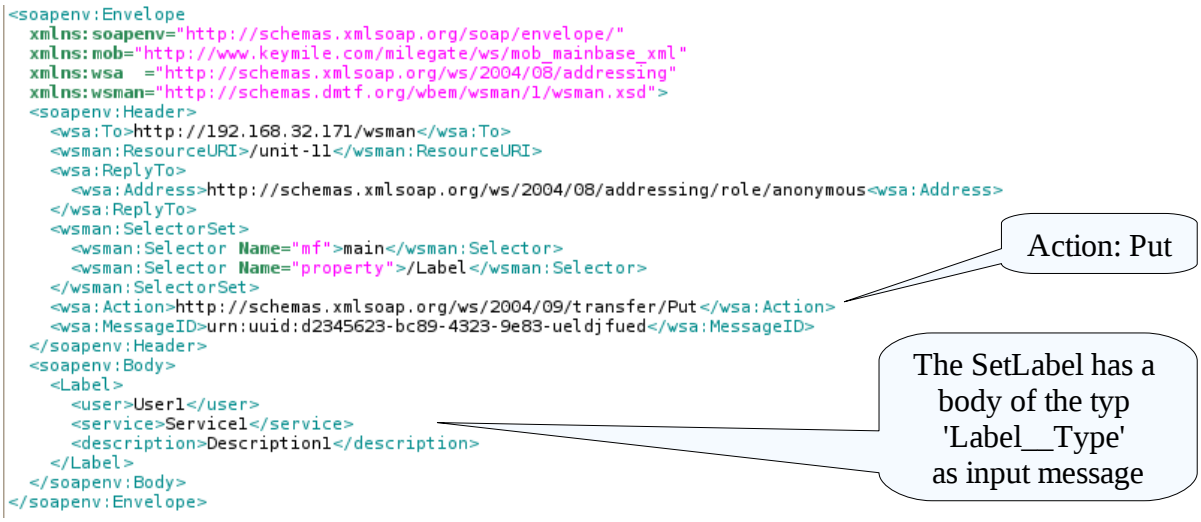



```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:mob="http://www.keymile.com/milegate/ws/mob_mainbase_xml"
  xmlns:wsa  ="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
  <soapenv:Header>
    <wsa:To>http://192.168.32.171/wsman</wsa:To>
    <wsman:ResourceURI>/unit-11</wsman:ResourceURI>
    <wsa:ReplyTo>
      <wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:Address>
    </wsa:ReplyTo>
    <wsman:SelectorSet>
      <wsman:Selector Name="mf">main</wsman:Selector>
      <wsman:Selector Name="property">/Label</wsman:Selector>
    </wsman:SelectorSet>
    <wsa:Action>http://schemas.xmlsoap.org/ws/2004/09/transfer/Get</wsa:Action>
    <wsa:MessageID>urn:uuid:d2345623-bc89-4323-9e83-ueldjfued</wsa:MessageID>
  </soapenv:Header>
  <soapenv:Body />
</soapenv:Envelope>
```

The input type (view of service) of the GetLabel message (defined in PortTypes):

```
<input message="mgws:EmptyMessage"/>      <!-- NO BODY -->
```

SetLabel SOAP message:




```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:mob="http://www.keymile.com/milegate/ws/mob_mainbase_xml"
  xmlns:wsa  ="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd">
  <soapenv:Header>
    <wsa:To>http://192.168.32.171/wsman</wsa:To>
    <wsman:ResourceURI>/unit-11</wsman:ResourceURI>
    <wsa:ReplyTo>
      <wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous<wsa:Address>
    </wsa:ReplyTo>
    <wsman:SelectorSet>
      <wsman:Selector Name="mf">main</wsman:Selector>
      <wsman:Selector Name="property">/Label</wsman:Selector>
    </wsman:SelectorSet>
    <wsa:Action>http://schemas.xmlsoap.org/ws/2004/09/transfer/Put</wsa:Action>
    <wsa:MessageID>urn:uuid:d2345623-bc89-4323-9e83-ueldjfued</wsa:MessageID>
  </soapenv:Header>
  <soapenv:Body>
    <Label>
      <user>User1</user>
      <service>Service1</service>
      <description>Description1</description>
    </Label>
  </soapenv:Body>
</soapenv:Envelope>
```

The input type (view of service) of the SetLabel message (defined in PortTypes):

```
<input message="mob_mainbase_xml:Label__Message"/>
```

6 Web Service Concepts

6.1 Addressing

…

6.1.1 WS-Addressing

…

- How to address the MileGate
- Binding of the transport protocol
- SOAP Fault Binding
- Metadata, WS-Policy, WDSL Binding (indicate support of WS-Addressing)

6.1.1.1 Endpoint Reference

- Concept of Endpoint Reference.
- Send/Receive EPR
  http://svn.apache.org/repos/asf/cxf/trunk/testutils/src/main/resources/wsdl/locator.wsdl

6.1.2 WS-Management

…

- Addressing the resource
- Addressing the management function

6.1.3 WS-Transfer

…

- Specification of the actions (get, set, put, delete, ..)

6.2 Resource

6.2.1 WS-Discovery

WS-Discovery has the state of an OASIS Committee Specification 01 since 14 May 2009.[2]

It defines a discovery protocol to locate services. It is often used to discover structures like LDAP (Lightweight Directory Access Protocol) or similar directories.

As our system contains one single service per MileGate, we have no need of a discovery at this level. Discovery could be used to figure out the complete infrastructure (ensemble of MileGates). Actually, this function is not needed because the system architecture and its addressing is designed in advance and won't change over the time.

CAN'T BE USED TO DISCOVER THE MANAGED OBJECTS (RESOURCE) OF THE MILEGATE!

6.2.2 WS-Resource

WS-Resource became a OASIS Standard the 1 April 2006.

The goal of WS-Resource is to standardize the terminology and concepts needed to express the relationship between Web services and resources.[3]

A resource is represented by an endpoint reference (EPR) and addressed using the WS-Addressing concept:

```
<wsa:EndpointReference>
   <wsa:Address>http://192.168.0.1?res=RessourceName</wsa:Address>
   …
</wsa:EndpointReference>
```

The SOAP binding would look as followed:

```
<wsa:To>http://192.168.0.1?res=RessourceName</wsa:To>
```

6.2.2.1 WS-Resource Properties[4]

WS-Resource Properties also became a OASIS Standard the 1 April 2006.

2 http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.pdf
3 http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf (1.1 Goals and Requirements)

The goal of WS-ResourceProperties is to standardize the terminology, concepts, operations, WSDL and XML needed to express the resource properties projection, its association with the Web service interface, and the messages defining the query and update capability against the properties of a WS-Resource.

Resource Property:
A resource property is a piece of information defined as part of the state model of a WS196 Resource.

Resource Properties Document:
The XML document representing a logical composition of resource property elements. The resource properties document defines a particular view or projection of the state data implemented by the WS-Resource.

6.2.2.2 Comment

This concepts offer another concept for addressing the ???Types??? (e.g. Lable ?) and its properties (parameters, e.g. Lable1).

- With GetMultipleResourceProperties we can get a selection of Resource Properties. This mechanism offers the possibility of a customized requests according to the preferences of the client. The advantage is that we do not have to request multiple operations and filter the content afterwards.
- With QueryResourceProperties we are able to query a Resource Properties document of a WS-Resource using a query expression such as XPath.
- The manageability of the system could be improved due to the dynamic add/delete of Resource Properties into the Resource Property document. (InsertResourceProperties, UpdateResourceProperties, DeleteResourceProperties)

DOES NOT HELP TO FIGURE OUT WHICH ENDPOINT IS SUPPORTED BY WHICH OPERATION!

6.2.3 WS-Notification

WS-Notification contains the standars WS-Base Notification, WS-Brockered Notification and WS-Topics.

6.2.3.1 WS-Base Notification

WS-Base Notification became a OASIS Standard the 1 October 2006.

The goal of WS-BaseNotification is to standardize the terminology, concepts, operations, WSDL and XML needed to express the basic roles involved in Web services publish and subscribe for notification message exchange.[5]

A notify message containing one or more notifications should look as followed:[6]

```
…
<wsnt:Notify>
   <wsnt:NotificationMessage>
      <wsnt:SubscriptionReference>
         wsa:EndpointReferenceType
      </wsnt:SubscriptionReference> ?
      <wsnt:Topic Dialect="xsd:anyURI">
         {any} ?
      </wsnt:Topic>?
      <wsnt:ProducerReference>
         wsa:EndpointReferenceType
      </wsnt:ProducerReference> ?
      <wsnt:Message>
         {any}
      </wsnt:Message>
   </wsnt:NotificationMessage> +
      {any} *
</wsnt:Notify>
…
```

The notify message just before is transported as content of the SOAP body. Addressing for the notification (in SOAP header) by definition is following WS-Addressing action.

```
 <wsa:Action>
   http://docs.oasis-open.org/wsn/bw-2/NotificationConsumer/Notify
</wsa:Action>
```

The concept for the management of the subscription is also defined in WS-Base Notification.

6.2.3.2 WS-Brokered Notification

WS-Topics became a OASIS Standard the 1 October 2006.

The goal of WS-BrokeredNotification is to standardize message exchanges involved in Web services publish and subscribe of a message broker.[7]

6.2.3.3 WS-Topics

WS-Topics became a OASIS Standard the 1 October 2006.

5 http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf (1.1 Goals and Requirements)
6 http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf (3.2 Notify)
7 http://docs.oasis-open.org/wsn/wsn-ws_brokered_notification-1.3-spec-os.pdf (1.1 Goals and Requirements)

The goal of the WS-Topics specification is to define a mechanism to organize and categorize items of interest for subscription known as “topics”. It defines a set of topic expression dialects that can be used as subscription expressions in subscribe request messages and other parts of the WS-Notification system.[8]

Topic:
A Topic is the concept used to categorize Notifications and their related Notification schemas.

Topic Tree:
A hierarchical grouping of Topics.

6.2.3.4 Comment

The mechanism described in this standards is basically similar to the notification system used in the MileGate. The requirement for the notifications used for the logbook could be fulfilled with this technique without the need for a continuous polling. (Pull-style notifications also possible)
Informations in the logbook subcategories configuration changes, alarm, session login, equipment changes and events can be made accessible in a more particular way for other purposes.
It is recommended to allow authorization policies for topics.

- The hierarchical structure of the topics allows a very targeted subscription for notifications.
- Management of the topics stays handy, also for large topic sets.
- The client can regroup the readout of notification according to his belongings and anywhere in his system.

6.3 Management

6.3.1 WS-Distributed Management

6.3.1.1 Management Using Web Services

...

8 http://docs.oasis-open.org/wsn/wsn-ws_topics-1.3-spec-os.pdf (1.1 Goals and Requirements)

7 Annexes

7.1 Revision history

Revision			Short description of the modification	Prepared by	Checked by	Approved
Doc ID	Version	Date				
WS	1	24/06/09	First Version	DSCHN		
WS	1.1	26/06/09	- WebService Description & SOAP message - WS-Resource and WS-Notification concept - References	DSCHN		

7.2 References

7.2.1 Service Oriented Architecture / Web Service Architecture

• World Wide Web Concortium
http://www.w3.org

• Book: Service-orientierte Architekturen mit Web Services, Konzepte – Standards – Praxis, Ingo Melzer et al., SPEKTRUM Akademischer Verlag

• Book: Web Services. Principles and Technology, Michael P. Papazoglou, PEARSON

7.2.2 Webservice description / concepts

• WS-Addressing, W3C Recommendation
http://www.w3.org/TR/ws-addr-core/

• WS-A: WSDL Binding, W3C Recommendation
http://www.w3.org/TR/ws-addr-wsdl/

• WS-A: SOAP Binding, W3C Recommendation
http://www.w3.org/TR/ws-addr-soap/

• WS-A: Metatdata, W3C Recommendation
http://www.w3.org/TR/ws-addr-metadata/

• WS-Management, Distributed Management Task Force

http://www.dmtf.org/standards/published_documents/DSP0226_1.0.0.pdf

• WS-Transfer, W3C Submission
http://www.w3.org/Submission/WS-Transfer/

• WS-Discovery
http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.pdf

• WS-Base Notification
http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf

• WS-Brokered Notification
http://docs.oasis-open.org/wsn/wsn-ws_brokered_notification-1.3-spec-os.pdf

• WS-Topics
http://docs.oasis-open.org/wsn/wsn-ws_topics-1.3-spec-os.pdf

• WS-Resource ..
http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf

• WS-Resource Properties
http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-os.pdf

• WS-Distributed Management: Management Using Web Services MUWS Part 1, OASIS Standard
http://docs.oasis-open.org/wsdm/wsdm-muws1-1.1-spec-os-01.pdf

• WS-Distributed Management: Management Using Web Services MUWS Part 2, OASIS Standard
http://docs.oasis-open.org/wsdm/wsdm-muws2-1.1-spec-os-01.pdf

• UDDI (Universal Description, Discovery and Integration), OASIS Standard
http://uddi.org/pubs/uddi-v3.0.2-20041019.pdf