Introduction to creating WS-Management Resources

By William Reichardt

Introduction

This document is intended for someone just starting out in the area of web services manageability. Familiarity with the Java programming language (JDK5.0) and Jakarta Tomcat or any other J2EE Servlet engine is assumed. This tutorial will cover the basics of exposing an existing resource or object inside of a java VM as a WS-Man compliant resource.

Conceiving a Resource

A resource is the term we will use to describe an object with attributes and operations, which represent the component to be managed and exposed as a web service. Resources are initially modeled using a collection of primitive types such as Strings, longs and the like which are collected into complex structures. These structures are described using a dialect of XML called XML schema.

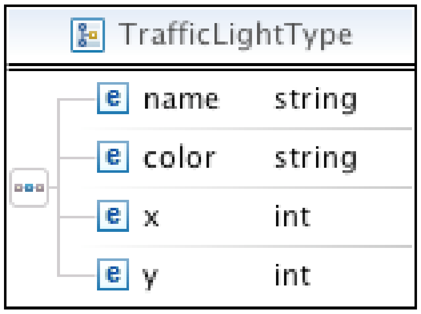
Your resource should be able to be described by and XML Schema document. There are some basic rules to follow when creating a schema document from scratch. You can start out with this template for a simple schema document.

Figure 1. A Simple XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://sun.com/traffic/light/types"
elementFormDefault="qualified" blockDefault="#all"
xmlns:tl="http://sun.com/traffic/light/types"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<xs:complexType name="TrafficLightType">
		<xs:sequence>
			<xs:element name="name" type="xs:string"/>
			<xs:element name="color" type="xs:string"/>
			<xs:element name="x" type="xs:int"/>
			<xs:element name="y" type="xs:int"/>
		</xs:sequence>
	</xs:complexType>
	<xs:element name="trafficlight"
type="tl:TrafficLightType"/>
</xs:schema>
```

Here you see an XML document which draws on the schema namespace to describe a complex type called TrafficLightType. TrafficLightType defines an template for what this XML resource should look like. In this case we are describing a resources that manages a traffic light. It has four attributes or properties. These are name ,color, x and y. X and Y are the map coordinates this light has been placed at. The complex type would look like this:

Figure 2. UML of Traffic Light Resource



By know you probably see how you would construct a schema of your own but the structure of this document bares some explanation.

Figure 3. Structure of the schema XML document

```
schema
        complexType
                sequence
                        element
                        element
                        element
                        element
        element
```

Note that this XML document defines a schema as having two parts. The complex type which is constructed from basic types defined in the schema specification and an element declaration. The element declaration names the element which wraps this complex type when it is actually used in an XML document. Both parts must be present to completely describe your document.

An example of a document that could be described by this schema might look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<tl:TrafficLightType
xmlns:tl="http://sun.com/traffic/light/types">
        <tl:name>Light1</tl:name>
        <tl:color>red</tl:color >
        <tl:x>100</tl:x>
        <tl:y>210</tl:y>
</tl:TrafficLightType >
```

Here we see a sample traffic light resource. Notice that its elements are all in the namespace `xmlns:tl="http://sun.com/traffic/light/types”.` You must choose a namespace for your resource as well. Usually you will find two namespaces in a schema (XSD) document. The first is the Schema Specification namespace, `xmlns:xs="http://www.w3.org/2001/XMLSchema"` which contains the elements you will use to describe your complex type and the other will be the namespace you invent for the parts of your complex structure. Also note that schemas should also declare a target namespace. This is the namespace that all elements without an explicit namespace should fall into. In this case it should be your resources namespace as well. Above it is `targetNamespace="http://sun.com/traffic/light/types"` .

When building your own resources you might want to consider using a XML schema editor such as XML-Spy or Eclipse WTP. These editors will guide you and make sure your schema is valid. They will also show you all the possible primitive types offered in XML schema. Here is a short list if you intend on building your schema by hand.

http://www.w3.org/TR/xmlschema-2/#built-in-datatypes

Another alternative to hand coding your resource’s schema is to generate it using tooling. Wiseman is in the process of developing tooling to generate schema from a CIM MOF. If you are planning on exporting your CIM model you may be interested in the Wiseman Mapping tooling which is part of the Wiseman project.

What is a WSDL document and Why Do We Need One?

Now that we have described our resource you may have some questions such as: How do I declare my own operations? That is where WSDL comes in. There is far more to WSDL than you will need to deal with in this tutorial. WSDL or Web Service Description Language is used to describe your web service to the world. If someone downloads your web service’s WSDL they will then know what kinds of XML documents you require and return (Via your included XSD file) and all of the operations you support and what parameters the require. The problem is that it is not a very human friendly XML document to work with. We will be generating our own WSDL document from the information you provided in your resources XSD document.

Once we have a WSDL document you will be able to use it for two things. These are:

1. A WSDL document can be published in a web service registry such as UDDI so that other’s can know how to make calls on your resource.
2. The WSDL document will be used to auto generate a Java Web Application which will implement your web service in any J2EE application server.

The WS-Man specification defines and requires some standard operations as well. These are what you might expect. Some examples are get, put and create which are used to access your resources. These will be imported into your WSDL from the specification

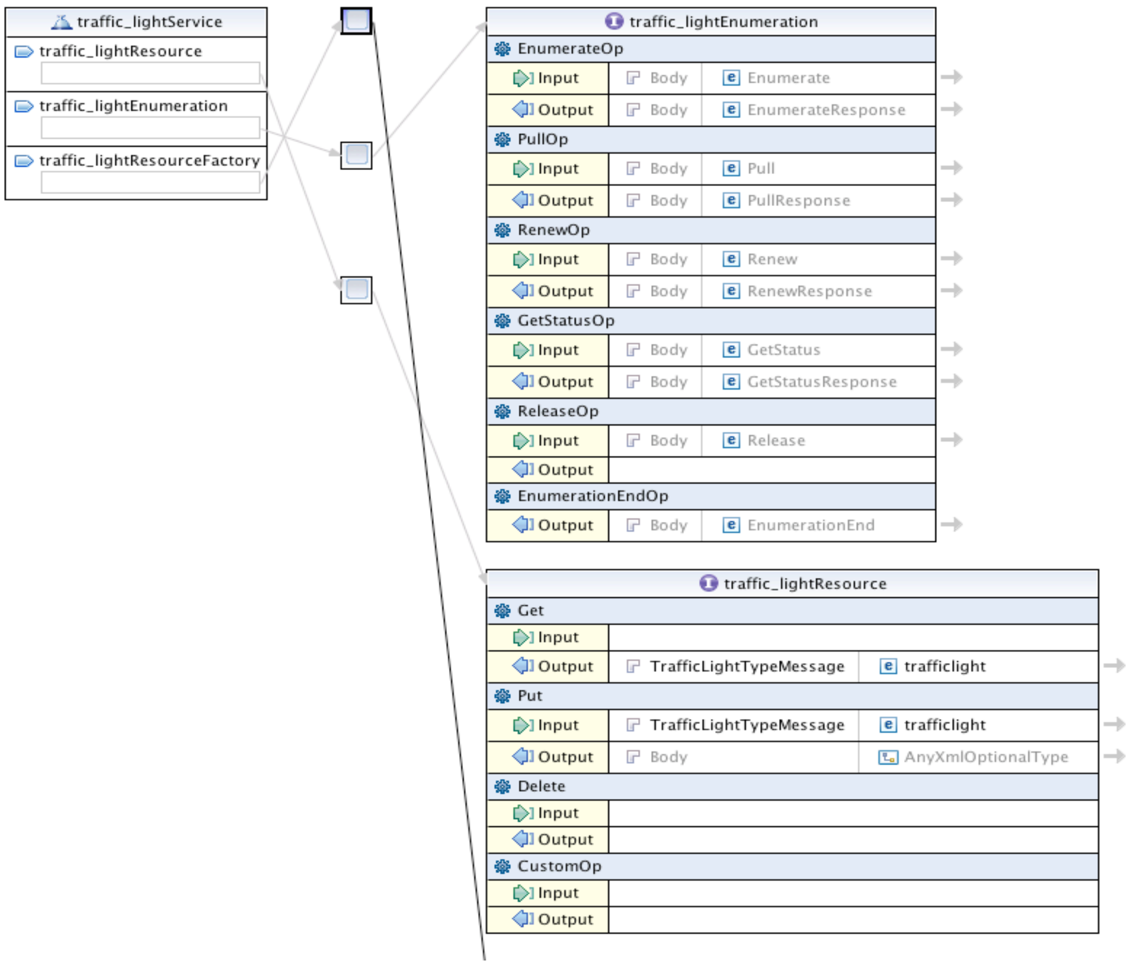
WSDL so that in the end, your WSDL will contain only information specific to your resource.

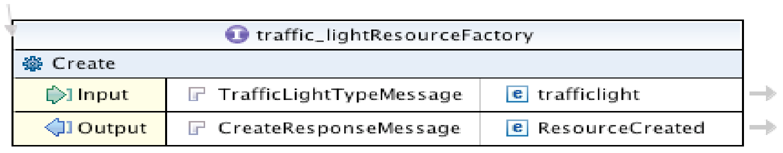
Creating your WSDL Document.

In the samples directory of your Wiseman install you will find a project called trafficlight _server. It contains an Apache ANT build script which you may want to re-use for your own projects. It also contains a copy of the traffic_light.xsd schema document you will be working with. Note that is you want to change the name of the schema file you are using to generate you must change it in the project.properties file which is in the same directory. To generate your WSDL document, enter the following command in the samples directory:

```
ant genwsdl
```

Note: You may have to copy the wiseman_tools.jar to your $ANT_HOME/lib directory for this task to work. This will invoke a custom ant task, which will generate wsdl/traffic_light.wsdl. It is an XML file and you should open it in a test editor and take a look at it. Below is a graphic representation of it, which should help to explain better.





As you can see, the WSDL generation process produced three interfaces or “ports” in web service speak. These ports are not important other than they organize the specification operations into categories. There is a factory to create new traffic lights, an interface to get properties and one to enumerate a list of lights if that is what this resource does.

Each spec operation requires one XML document as an input and one as an output. The format of these documents if defined in the specification WSDL, inside its own embedded schema document. You will also notice there is one operation called customop. Customop is a sample custom operation which is generated for you automatically. If you intend to implement your own custom operation you may copy this one as your starting point.

For this tutorial we will proceed with the generated WSDL unmodified.

Generating a Java Web Service Implementation of this WSDL

We would now like to generate a Java Application from this WSDL document. You might wonder why you might not use Apache AXIS or the JWSDP to convert this WSDL into a Java application as these product offer code generation tools. The reason is that the WS-Man specification dictates that XML documents be dispatched (or mapped to) Java classes using WS-Addressing information and not off the first element in the SOAP document body as is common in the WS-I Basic Profile for web services. In many cases WS-Man documents may have no document body at all which is not permitted by WS-I. Because AXIS and JWSDP do not yet support addressing based dispatching of XML documents, Wiseman uses its own code generation framework to create web services.

To generate Java source for this resource use the following command to invoke ANT:

```
ant generate
```

Note that this command will require access to the Internet to download specification WSDLs as part of the generation process.