Nimsoft Monitor

SOAP Web Services
Getting Started Guide

Version 2.0

Legal Notices

Warranty

The material contained in this document is provided "as is," and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Nimsoft LLC disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Nimsoft LLC shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Nimsoft LLC and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

No part of this manual may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Nimsoft LLC as governed by United States and international copyright laws.

Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Nimsoft LLC's standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Trademarks

Nimsoft is a trademark of CA.

Adobe®, Acrobat®, Acrobat Reader®, and Acrobat Exchange® are registered trademarks of Adobe Systems Incorporated.

Intel® and Pentium® are U.S. registered trademarks of Intel Corporation.

Java(TM) is a U.S. trademark of Sun Microsystems, Inc.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

Netscape(TM) is a U.S. trademark of Netscape Communications Corporation.

Oracle® is a U.S. registered trademark of Oracle Corporation, Redwood City, California.

UNIX® is a registered trademark of the Open Group.

ITIL® is a Registered Trade Mark of the Office of Government Commerce in the United Kingdom and other countries.

All other trademarks, trade names, service marks and logos referenced herein belong to their respective companies.

Table of Contents

Prerequisites

The following are required to make use of this quick start guide:

- Java 6 JDK
- Maven 2.2.1 (or later) (http://maven.apache.org/)
- Internet connection for downloading maven libraries.
- Running Nimsoft Monitor software with web services installed.

Basic knowledge of Maven is a plus, but not required for getting started. We recommend that you add Maven bin directory to your system path (this is not required, but it will make running maven calls easier).

Setup

Follow these steps:

1. Create a new project folder.
2. Create the subdirectories within this folder:
   - src/main/java
   - src/test/java
3. Add the following content to a file called **pom.xml**:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.mycompany</groupId>
   <artifactId>nimbus-webservices</artifactId>
   <packaging>jar</packaging>
   <version>1.0-SNAPSHOT</version>

   <dependencies>
      <dependency>
         <groupId>org.apache.cxf</groupId>
         <artifactId>cxf-rt-frontend-jaxws</artifactId>
         <version>${cxf.version}</version>
      </dependency>
      <dependency>
         <groupId>org.apache.cxf</groupId>
         <artifactId>cxf-rt-ws-security</artifactId>
         <version>${cxf.version}</version>
      </dependency>
      <dependency>
         <groupId>org.apache.cxf</groupId>
         <artifactId>cxf-rt-transports-http</artifactId>
         <version>${cxf.version}</version>
      </dependency>

      <dependency>
         <groupId>org.apache.cxf</groupId>
         <artifactId>cxf-rt-frontend-simple</artifactId>
         <version>${cxf.version}</version>
      </dependency>

      <dependency>
```

```
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.7</version>
            <scope>test</scope>
         </dependency>
      </dependencies>

      <build>
         <plugins>

            <!-- enable java 6 -->
            <plugin>
               <groupId>org.apache.maven.plugins</groupId>
               <artifactId>maven-compiler-plugin</artifactId>
               <configuration>
                  <source>1.6</source>
                  <target>1.6</target>
               </configuration>
            </plugin>

            <plugin>
               <groupId>org.apache.cxf</groupId>
               <artifactId>cxf-codegen-plugin</artifactId>
               <version>${cxf.version}</version>
               <executions>
                  <execution>
                     <id>generate-sources</id>
                     <phase>generate-sources</phase>
                     <configuration>

<sourceRoot>${project.build.directory}/generated/cxf</sourceRoot>
                        <wsdlOptions>
                           <wsdlOption>
                              <wsdl>${ws.address}AccountWS?wsdl</wsdl>
                           </wsdlOption>
                           <wsdlOption>
                              <wsdl>${ws.address}AlarmWS?wsdl</wsdl>
                           </wsdlOption>
                           <wsdlOption>
                              <wsdl>${ws.address}DashboardWS?wsdl</wsdl>
                           </wsdlOption>
                           <wsdlOption>
                              <wsdl>${ws.address}NimBUSWS?wsdl</wsdl>
                           </wsdlOption>
                           <wsdlOption>
                              <wsdl>${ws.address}ProbeWS?wsdl</wsdl>
                           </wsdlOption>
                           <wsdlOption>
                              <wsdl>${ws.address}QOSWS?wsdl</wsdl>
```

```
                        </wsdlOption>
                        <wsdlOption>
                           <wsdl>${ws.address}SLAWS?wsdl</wsdl>
                        </wsdlOption>
                        <wsdlOption>
                           <wsdl>${ws.address}SLOWS?wsdl</wsdl>
                        </wsdlOption>
                        <wsdlOption>
                           <wsdl>${ws.address}UserWS?wsdl</wsdl>
                        </wsdlOption>
                        <wsdlOption>
                           <wsdl>${ws.address}VariableWS?wsdl</wsdl>
                        </wsdlOption>
                        <wsdlOption>
                           <wsdl>${ws.address}VersionWS?wsdl</wsdl>
                        </wsdlOption>


                     </wsdlOptions>
                  </configuration>
                  <goals>
                     <goal>wsdl2java</goal>
                  </goals>
               </execution>
            </executions>
         </plugin>
      </plugins>
   </build>

   <repositories>
      <repository>
         <id>apache-snapshots</id>
         <name>Apache SNAPSHOT Repository</name>
         <url>http://repository.apache.org/snapshots/</url>
         <snapshots>
            <enabled>true</enabled>
         </snapshots>
      </repository>
   </repositories>

   <properties>
      <cxf.version>2.2.4</cxf.version>
      <ws.address>http://localhost:8084/ws/</ws.address>
   </properties>

</project>
```

4. If your Nimsoft Monitoring software web service is installed on a different host than localhost, change the **properties** entry in the bottom to reflect this.
5. Check that the web services are up and running by accessing the following URL: http://localhsot:8084/ws/
6. From a command shell, run the maven command: **mvn install**.

   This generates service stubs for the web services, and they are ready for use.
7. Create project files for your IDEs.
   - If you are running eclipse, enter: **mvn eclipse:eclipse**
   - If you are running Intellij IDEA, enter: **mvn idea:idea**
   - If you are running Netbeans, enter: **mvn netbeans-freeform:generate-netbeans-project**
8. Import project into your editor.

   In Eclipse, this is done by clicking **File - > Import**, and then selecting **Existing Project into Workspace**.
9. Create a new test class in src/test/java/test/GetAlarmsTest.java with the following content:

```
package test;

import java.util.HashMap;
import java.util.List;
import org.apache.cxf.endpoint.Client;
import org.apache.cxf.frontend.ClientProxy;
import org.apache.cxf.jaxws.JaxWsProxyFactoryBean;
import org.apache.cxf.ws.security.wss4j.WSS4JOutInterceptor;
import org.apache.ws.security.WSConstants;
import org.apache.ws.security.handler.WSHandlerConstants;
import org.junit.Test;
import com.nimsoft.nms.services.Alarm;
import com.nimsoft.nms.services.AlarmWS;

public class GetAlarmsTest {

    static final String PASSWORD = "<password>";
    static final String USER_NAME = "<user>";
    static final String WS_ADDRESS = "http://localhost:8084/ws/";

    @Test
    public void test_getAlarms() {
        JaxWsProxyFactoryBean factory = new JaxWsProxyFactoryBean();

        factory.setServiceClass(AlarmWS.class);
        factory.setAddress(WS_ADDRESS + "AlarmWS");
        prepareService(factory.create());
```

```
        AlarmWS alarmService = (AlarmWS) factory.create();

        List<Alarm> alarms = alarmService.getAlarms();

        System.out.println("Number of alarms:" + alarms.size());
    }

    void prepareService(Object accountService) {
        Client client = ClientProxy.getClient(accountService);

        HashMap<String, Object> outProps = new HashMap<String, Object>();
        outProps.put(WSHandlerConstants.ACTION,
WSHandlerConstants.USERNAME_TOKEN);
        outProps.put(WSHandlerConstants.USER, USER_NAME);
        outProps.put(WSHandlerConstants.PASSWORD_TYPE, WSConstants.PW_TEXT);
        outProps.put(WSHandlerConstants.PW_CALLBACK_CLASS,
ClientPasswordHandler.class.getName());

        WSS4JOutInterceptor wssOut = new WSS4JoutInterceptor(outProps);
        client.getOutInterceptors().add(wssOut);
    }
}
```

10. Add the file ClientPasswordHandler.java in the same package.

    This file is used by CXF to handle the password in ws-security.

```
package test;

import java.io.IOException;
import javax.security.auth.callback.Callback;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.callback.UnsupportedCallbackException;
import org.apache.ws.security.WSPasswordCallback;

public class ClientPasswordHandler implements CallbackHandler {

    public void handle(Callback[] callbacks) throws IOException,
UnsupportedCallbackException {

        WSPasswordCallback pc = (WSPasswordCallback) callbacks[0];

        pc.setPassword(GetAlarmsTest.PASSWORD);
    }
}
```

11. Replace <user> and <password> with a user with access to the service.
12. Test your service by running the class GetAlarmsTest as a Junit test.
13. For consuming other web-services, replace AlarmWS with some of the other provided interfaces.

Invoking Web Services using Microsoft Tools

Use https for the transport. Setting WASP to use https involves a raw configure of the https_port key; set it to default of 443. Without https, .Net WCF will not allow clear username and password.

Using https requires code that deals with the certificate validation. You could consider validating any certificate.

The task is to create a custom binding in APP.CONFIG that works. The <security> section below is crucial, and .Net WCF will not build it for you in Visual Studio.

```
<customBinding>
   <binding name="MyBinding">
      <security authenticationMode="UserNameOverTransport"
              messageSecurityVersion="WSSecurity11WSTrustFebruary2005WSSecureConv
ersationFebruary2005WSSecurityPolicy11"
              securityHeaderLayout="Strict"
              includeTimestamp="false"
              requireDerivedKeys="true">
      </security>
      <textMessageEncoding messageVersion="Soap11" />
      <httpsTransport authenticationScheme="Negotiate" requireClientCertificate
="false" realm =""/>
   </binding>
</customBinding>
```

IMPORTANT: Set the messageVersion to Soap11 not the default of Soap12.

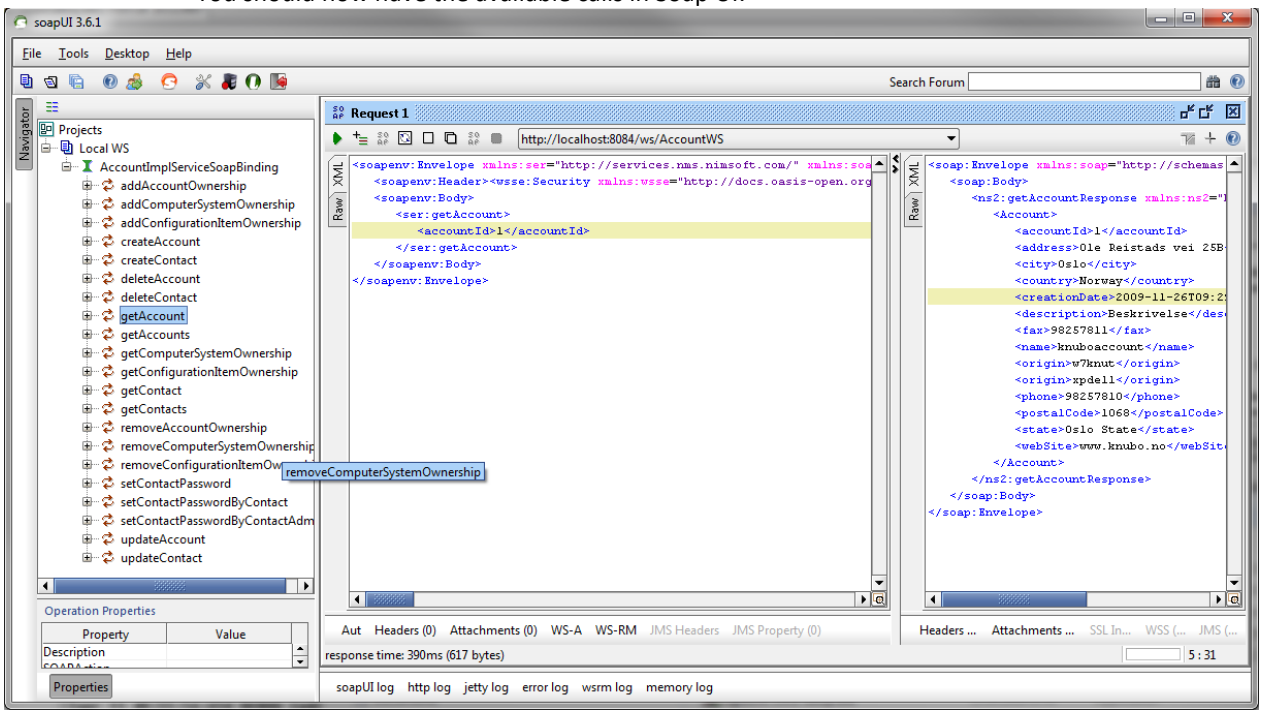
The following example below should now work:

```
Imports System.ServiceModel
Imports System.ServiceModel.Channels
Imports System.Net
Imports System.Net.Security
Imports System.Security.Cryptography.X509Certificates
Imports System.ServiceModel.Dispatcher
Imports System.ServiceModel.Description
Imports System.ServiceModel.Configuration ' for BehaviorExtensionElement


Public Class TestUtils

    Public Shared Sub OverrideCertificateValidation()
        ServicePointManager.ServerCertificateValidationCallback = New
RemoteCertificateValidationCallback(AddressOf RemoteCertValidate)
    End Sub

    Private Shared Function RemoteCertValidate(ByVal sender As Object, ByVal
cert As X509Certificate, ByVal chain As X509Chain, ByVal [error] As
System.Net.Security.SslPolicyErrors) As Boolean
        Return True
    End Function
End Class


Module Module1


    Sub Main()

        TestUtils.OverrideCertificateValidation()

        Dim CallWebService As New nimsoftVersion2.VersionWSClient()

        CallWebService.ClientCredentials.UserName.UserName = "administrator"
        CallWebService.ClientCredentials.UserName.Password = "<password>"

        Dim sGetValue As String = "blank"

        sGetValue = CallWebService.getVersion()

        System.Console.WriteLine("Nimsoft Web Services version: " &
sGetValue)
        System.Console.Read()
    End Sub

End Module
```

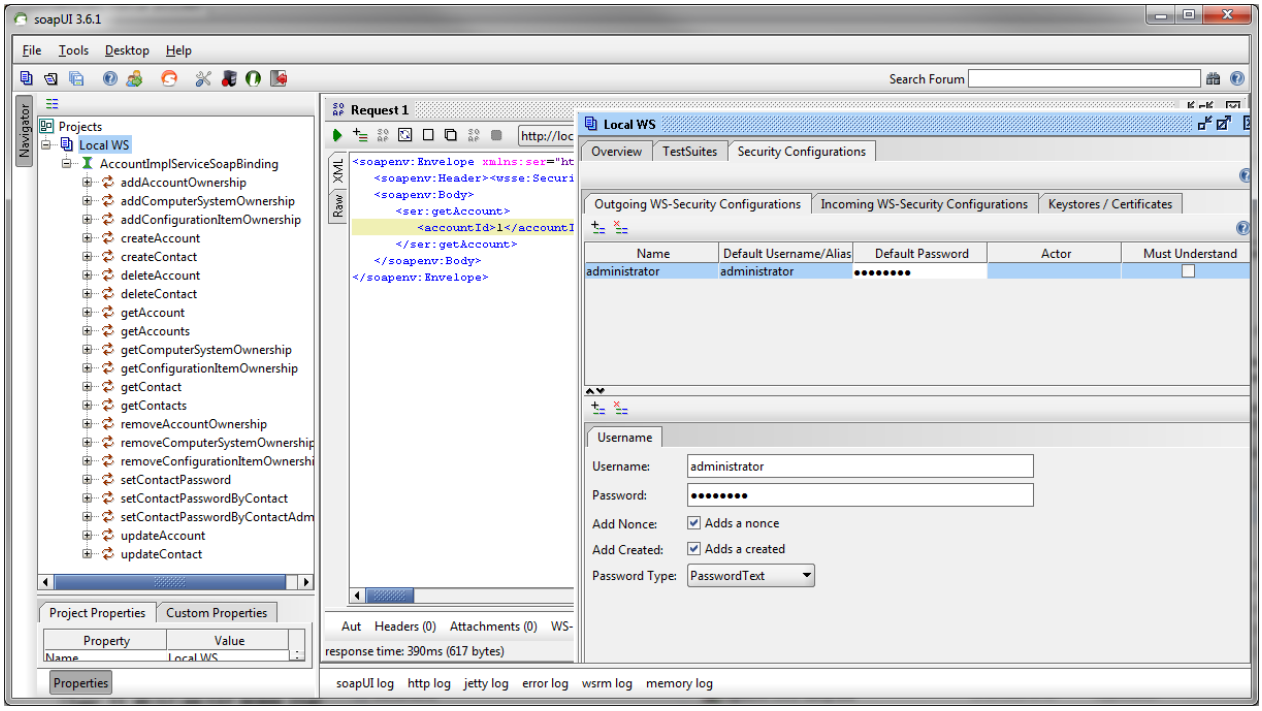
Invoking Web Services using Soap UI

First add the WSDL you want to test to your project. You will find all available WSDL if go to the URL that points to your wasp: http://localhost:8084/ws/. The WSDL to the account service is then: http://localhost:8084/ws/AccountWS?wsdl

You should now have the available calls in Soap UI:



Now you must configure authentication parameters:
Right click your project and open “Show Project View”. Navigate to the Security Configurations. First add an outgoing WS-Security configurations (see the + sign above the first line, and then add a new WSS Entry:



In your request 1 for the service you want to call, right click and select menu point Outgoing WSS and select Apply “administrator” (or whatever your entry was called). You will now get a <wsse:Security token and you should be able to call the wesbservice and get the response back that you want.