KOMBIT

Guideline for Java Sample Service

Version 1.0

# Sample Java Service And Consumer using STS

## Introduction

The following describes how to configure the Java-based samples to request a token from the STS, call a protected service using this token, and how to protect a java service to require a token.

This is a soap service that authenticates the caller with a token issued by an STS compliant with the KOMBIT Støttesystemer specification for STS. The service has a simple ping method, that requires no input and which returns a statically configured text message.

In the following, this service is also referred to as Service.

In the following, a consumer refers to the caller of the service.

After completing this guide, the java-based sampleservice will be configured.

It is assumed that the reader is a Java developer knowledgeable in the following technologies used to develop this based sample. This includes:

* Java
* NetBeans Development Environment (IDE)
* Glassfish application server with the Java Metro framework
* OIOSamlJava library
* X509v3 Certificates
* Java keystore

## Prerequisites

The following prerequisites must be in place to be able to complete this guide.

* JDK 7 installed
* NetBeans 8, installed with Glassfish (metro framework included)
* Unlimited Strength Cryptography file downloaded from Oracle and installed.

## Overview

This sample is comprised of two main web projects:

* Service
* Consumer

And 3 supporting library open source projects from the OIOSAML.Java library

* [OIOSAML.java](https://svn.softwareborsen.dk/oiosaml.java/) (revision 21290)
* [OpenSaml](http://shibboleth.net/downloads/java-opensaml/) v2.5.1
* [XmlTooling](http://opensaml.org/) v1.4.4

Service is a soap service that authenticates the caller with a token issued by an STS compliant with the KOMBIT Støttesystemer specification for STS. The service has a simple ping method, that requires no input and which returns a statically configured text message.

Consumer is a caller of the soap service exposed by Service. This project contains servlets for the test cases below:

* ServiceConsumer: call Service
* IssueToken: issue token successfully directly from STS not through Service.
* IssueTokenByRstWithWrongFormat: try sending an empty soap message to STS and handle error returned from it.
* IssueTokenThrowError: try to issue token from STS which returns specific error and handle it.
* ServiceConsumerWithNormalOBO: call Service webservice with an issued token from STS as an OnBehalfOf token.
* ServiceConsumerWithProxyOBO: call Service webservice with a certificate token as an OnBehalfOf token.

## Import Certificates To Keystore

The first step is to import the relevant certificates with private keys into the relevant java keystore. The default password for java keystore in glassfish is changeit and keystore path of glassfish (on Microsoft Windows) is:

c:\Program Files\glassfish-4.1\glassfish\domains\domain1\config\keystore.jks

This is done as follows:

1. Import “CertificateAnvendersystem.p12” into glassfish keystore with the following command:

keytool -v -importkeystore -srckeystore {certpath}\CertificateAnvendersystem.p12-srcstoretype PKCS12 -destkeystore keystore.jks -deststoretype JKS

1. Change password of the certificate to be the same as the keystore password:

keytool -keypasswd  -alias {46bf4ba8-c898-45b1-b79f-0d7fb690adcf}-new changeit -storepass changeit -keystore keystore.jks

Alias is the output of the first command when the certificate is imported.

## Import Certificate To Trust Store

The next step is to import the public certificates into the cacerts keystore of glassfish.

The default password for trust store in glassfish is changeit and trust store of Glassfish (on Microsoft Windows) is located at:

c:\Program Files\glassfish-4.1\glassfish\domains\domain1\config\cacerts.jks

This is done as follows:

1. Import the public part of CertificateIdp.p12 to trust store, along with all root and intermediate certificates in the certificate chain.
2. Import the SSL certificate of the STS to the trust store, along with all root and intermediate certificates in the certificate chain.

The following command is a sample command for importing a certificate to the cacerts.jks keystore file:

keytool -import -alias kombit -file {certpath}\kombit.cer -keystore cacerts.jks

The supplied keystore files in the “certifcatestore” folder of project can be used, and may be copied to your glassfish config install folder:

c:\Program Files\glassfish-4.1\glassfish\domains\domain1\config

## Setup Consumer (Anvendersystem) With NetBeans

This section describes how to setup Consumer and build it for deployment:

1. Right click at Consumer project, click on Properties > Build > Packing > Compress WAR file to force the project to build the war file in the dist folder.
2. Open Constant.java in package consumer.util, and change the following configurations, if you’re not using the STS in the project environment:
   1. StsEndpointAddress The address where this STS service is deployed.
   2. StsMexEndpointAddress The mex endpoint address of STS
   3. UserContextEndpointAddress The address where Service is deployed.
   4. AppliesToEndpointAddress The applied to endpoint address.
3. Change log message path (optional)
   1. Open Constant.java in package consumer.util, change value of LogFilePath property (default value isC:\temp\).
4. Right Click at project click Clean and Build
5. Deploy the war file in dist folder to Flassfish then you can run the test servlets (see section [Test](https://github.com/Safewhere/kombit-service-java#testing)).

## Setup a Service With NetBeans

This section describes how to setup Service and build it for deployment:

1. Right click project, click on Properties > Build > Packing > Compress WAR file to force the project build war file in the dist folder.
2. Right click a web service, and select “Edit Web Service Attributes”
   1. Select Secure Service with option STS Issued Token with Service Certificate
   2. Click configure to edit key type and key size. Public key must be used.
   3. Do not input value for issuer address or meta data, since we will set it in the code to easily change the URL for deployment purposes.
   4. All the URL values are stored in the Constant.java file.
   5. Select the keystore and config with glass fish keystore and alias we import for CertificateAnvendersystem.p12 as above step
3. It is not recommended to edit setting on “security mechanism”. Otherwise, the generated file Web pages\WEB-INF\wsit-service.service.Service.xml will be re-generated so it must be updated manually as follows:
   1. On element AsymmetricBinding\Policy\InitiatorToken\Policy\IssueToken
      1. Replace value of IncludeToken with http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never
   2. On element AsymmetricBinding\Policy\InitiatorToken\Policy\IssueToken\RequestSecurityTokenTemplate\TokenType
      1. Replace value of TokenType with http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0
   3. Add element <sp:ProtectTokens/> below element AsymmetricBinding\Policy\IncludeTimestamp
      1. Remove element <sp:EncryptedParts>
      2. Replace policy wss11 by wss10
4. <sp:Wss10 xmlns:sp=“http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702”>
5. <wsp:Policy>
6. <sp:MustSupportRefKeyIdentifier />
7. <sp:MustSupportRefIssuerSerial />
8. </wsp:Policy>
9. </sp:Wss10>
10. Save XML file click Clean and Build, then deploy to Glassfish
11. Then we can access to that service WSDL URL: <https://adgangsstyringeksempler.projekt-stoettesystemerne.dk:8181/Service/Service>

## Testing

The deployed Service can be tested with the deployed Anvendersystem (consumer) by visiting these URLs:

<https://adgangsstyringeksempler.projekt-stoettesystemerne.dk:8181/Consumer/IssueToken>

<https://adgangsstyringeksempler.projekt-stoettesystemerne.dk:8181/Consumer/IssueTokenByRstWithWrongFormat>

<https://adgangsstyringeksempler.projekt-stoettesystemerne.dk:8181/Consumer/IssueTokenThrowsError>

<https://adgangsstyringeksempler.projekt-stoettesystemerne.dk:8181/Consumer/ServiceConsumer>

<https://adgangsstyringeksempler.projekt-stoettesystemerne.dk:8181/Consumer/ServiceConsumerWithNormalOBO>

<https://adgangsstyringeksempler.projekt-stoettesystemerne.dk:8181/Consumer/ServiceConsumerWithProxyOBO>

In addition, the service WSDL can be downloaded here:

<https://adgangsstyringeksempler.projekt-stoettesystemerne.dk:8181/Service/Service>