# Introduction

Since the web services allow applications running on heterogeneous platforms to communicate with each other. It is important for them to address the nonfunctional requirements like security, asynchronous communication in standard fashion so that we as developers find it easy and forget about coding to address problems like security. Instead, we can configure them simply. All web services standards are implemented by Apache **CXF** engine as same as many other engines that we can found in the industry.

# WS Standards Overview

Since the web services allow applications running on heterogeneous platforms, they need to work with each other in a seamless fashion. That is where the web services standards come into picture. The aimed goal here is to maintain the mostly used standards.

## Why WS Standards?

We are going to take a simple example:

|  |  |
| --- | --- |
|  | |
| **Application** | **Description** |
| **Online Shopping App** | User comes in and buy a few products and the uses his own credit card information to make the purchase. This application have to be connected to a **banks payment gateway** which is typically a web service |
| **Banks Payment Gateway Web service** | Get called by the **Online** **shopping** **App** in order to process the order. This gateway will ask us to authenticate by passing in a username and password to make sure that is really our application and not some hacker who is calling into their banks. |

### Where would the band ask us to put the username and password in the soap message?

|  |  |
| --- | --- |
| Is it in the SOAP body? | What should be the name of the XML elements in the SOAP header?  Just **username** and **password**? |
| In the SOAP header? |

That is where the web services security comes in and tells us where a particular username and password elements to pass in the authentication information. Without that, each bank might ask us to pass in a different set of elements and we have to keep changing our application to work seamlessly with any payment gateway (each time we use a new payment gateway).

## What is the most used WS standard?

The first and mostly used web services standard is the WS security standard. It addresses the three important issues around security:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Authentication** | | | | | Where we can make sur that web services consumer and the web services provider know about each other and they are really the ones who are calling into our web services provider. | | | | | | |
| **User Name Token Profile** | | | | | Used to pass simply the username & password. |
| **X 508 Certificates** | | | |  | | |
| **SAML** | | | Allows us to do a single sign on in web services applications. Once we sign in to one of the web services for an organization, we can use any other web services within that organization. When we use our **Gmail** and login into your **Gmail** account, you will be able to access **YouTube** or any other services like **Google Docs**, **Google Maps** that **Google** offers. | | | |
| **Confidentiality** | | | | | Concerns our messages. We exchange our XML SOAP messages between consumer and provider, you have to make sur that the hacker is not hacking him and reading those messages. | | | | | | |
| **Encrypt** | | | Using what the security provider says we use publicly cryptography | | | |
| **Decrypt** | | |
| **Integrity** | | | Protecting data integrity is making sure that messages are being sent is the consumer receiving the same as that by the provider and vice versa. | | | | | | | | |
| **XML Signature** | | | | Signing our SOAP messages and that signature will be included in the message so when the consumer sends the message along with the signature on the server or the provider side will calculate the signature. By compare both the signatures to make sure that the message was not manipulated on the way. | | | | |
| **MTOM** | | For exchanging files. (Any kind of files like documents Word, Excel, images …) using MTOM application. | | | | | | | | | |
| **WS-Addressing** | | | | | Asynchronous Callbacks. If your consumer do not want to wait (for a long time), they can give you a URL, to which you will call back once the processing is done and as absolute consumer. Another usage is to redirect the web service response to other server instead of the actual consumer who called into the service. | | | | | | |
| **WS-Policy** | | | | Assert and mandate certain rules to consume our web services by making sure that our web services consumers that are following what we are asking them to do. For example asking them to use **HTTPS** (Secured **HTTPS** in order to call your web services and web services engines like **CXF** will ensure that **WS**-**POLICY** is being followed and they will throw an exception back if the consumer is not using **HTTPS**) | | | | | | | |
| **WS-SecureConversation** | | | | | | | Improving performance while encrypting and decrypting by negotiating a key at the beginning instead of negotiating a key for each request which is very problematic and decreases the performance. | | | | |
| **WS-SecurityPolicy** | | | | | | A combination of WS Security and WS Policy that assures that the web service consumer in an easier fashion is following certain WS Security standards. | | | | | |
| **Note** | We do not need a lot of work to use these standards in our applications. (All these standards all implemented by the web services engines like **APACHE CXF**). Therefore, we need simply to configure them as our needs referring to the documentation. | | | | | | | | | | |

# Username Token Authentication Steps

Right now, there is no security implemented for our application. Any client can access our application without providing a username and password (there is no **authentication**). Using the WS Standards and exactly **user name token profile,** we are going to implement an authentication to secure our web service. Therefore, we are going to do that in four simple steps:

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **Adding the Maven CXF security dependency** | | |
| **2** | **Configure the WSS4J Interceptors** | | **WSS4J** stand for **Web Services Security For Java**. |
| **3** | **Create the Password Callback** | A class provides the password information so that these **WSS4J** can intercept the incoming message and validate it against the password that comes in. | |
| **4** | **Run and test the Application** | | |

# Add maven dependency

# Configure the Interceptors

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | |
| **WebServiceConfig** | 1 | configure the interceptor on the endpoint & telling the interceptor what to do by providing a bunch of properties |

# Provide the Interceptor Properties

That action that should be taken is the username token profile authentication. The password type here is a plain text and the password callback that will give us the password information back is the username token or UT password, which we have implemented in this class

# Setup a in memory database

|  |  |
| --- | --- |
| **UTPasswordCallback** | maintain an in-memory database |

# Implement the handle method

|  |  |
| --- | --- |
| **UTPasswordCallback** | Callback handler (has a handle method) which returns a password back. |

# Run the application

When we run our application and after sending the previous request message it will returns an “**A security error was encountered when verifying the message”.**

# Test

Adding a SOAP header element has the complete information to avoid the previous error by implementing the **WS** **Standards**. Therefore, WSS4J interceptors knows what to do with them.

# User Name Token Profile Authentication

# Exchanging files using MTOM

# Section Summary

Web services are loosely coupled use each other (web services consumer’s tries to consume a web service provider) without standards it is too hard to implement security, exchanging files … So **WS** **Standers** provides us the tools to just configure the needed standards in order to use them in our project an facilitate our life as a developer.

Table des matières

[1. Introduction 1](#_Toc94713683)

[2. WS Standards Overview 1](#_Toc94713684)

[2.1. Why WS Standards? 1](#_Toc94713685)

[2.1.1. Where would the band ask us to put the username and password in the soap message? 1](#_Toc94713686)

[2.2. What is the most used WS standard? 1](#_Toc94713687)

[3. Username Token Authentication Steps 2](#_Toc94713688)

[4. Add maven dependency 2](#_Toc94713689)

[5. Configure the Interceptors 2](#_Toc94713690)

[6. Provide the Interceptor Properties 3](#_Toc94713691)

[7. Setup a in memory database 3](#_Toc94713692)

[8. Implement the handle method 3](#_Toc94713693)

[9. Run the application 3](#_Toc94713694)

[10. Test 3](#_Toc94713695)

[11. User Name Token Profile Authentication 3](#_Toc94713696)

[12. Exchanging files using MTOM 3](#_Toc94713697)

[13. Section Summary 3](#_Toc94713698)