Kombit

Guideline for STS Test Stub

Version 1.0

# Introduction

The following document describes how to configure the STS Test Stub.

After completing this guide, the STS Test Stub will be configured.

Setting up the STS Test Stub in IIS is outside the scope of this document. This is described in the document “All\_guideline\_setup sites IIS.docx”.

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

* C#
* Microsoft.Net framework v4.5
* Microsoft Windows Server Operating System
* Microsoft Internet Information Systems (IIS)
* X509v3 Certificates
* SOAP protocol
* WS-Trust XML protocol

# Table of Contents

[Introduction 2](#_Toc418799841)

[Table of Contents 3](#_Toc418799842)

[Prerequisites 4](#_Toc418799843)

[Configuring The Test Stub 4](#_Toc418799844)

[IIS website 4](#_Toc418799845)

[Configuration 4](#_Toc418799846)

[Using The STS Test Stub 5](#_Toc418799847)

# Prerequisites

This document requires that the following prerequisites are satisfied:

* Setting up the .Net-based samples according to the guide “All\_guideline\_setup sites IIS.docx”
* Logging is done to the folder c:\temp. This folder must exist for logging to work.

# Configuring The Test Stub

The STS test stub is a soap service that simulates processing requests and sending responses for a WS-Trust call which a user system (Anvendersystem) can send.

It will accept a request security token message (RST) and will return with a request security token response message (RSTR).

Specifiic properties about the test stub:

* It uses a X509 signing credential with SHA-1 algorithm
* The RSTR contains a SAML2.0 Assertion element. That is, the RSTR contains an un-encrypted assertion. Assertion encryption is NOT used.
* The service has two endpoints
  + MEX endpoint used to publish metadata
  + Certificate endpoint used for processing RST messages and accepts a certificate credential over the HTTP protocol.

## IIS website

This guideline assumes that the url of the STS Test Stub is:

<https://adgangsstyringeksempler.projekt-stoettesystemerne.dk/STS/>

## Configuration

Some changes to the properties in the configuration file STS\web.config may be required:

* **BaseAddress**: the production address which deploy the STS service.
* **StsServiceCertificateThumbprint**: thumbprint of the certificate used by this STS Test stub as the service certificate for the certificate-endpoint.
  + In this sample, the supplied certificate “KombitTestSigningCertificate.p12” is used.
* **StsSigningCertificateThumbprint**: thumbprint of the certificate used by this STS Test Stub to sign the issued RSTR and Assertion.
  + In this sample, the supplied certificate “KombitTestSigningCertificate.p12” is used.
* **SigningAlgorithm**: the signing algorithm used by this STS Test Stub to sign the issued RSTR and Assertion
* **HeaderSigningAlgorithm**: the signing algorithm used by this STS Test Stub to sign the header of the soap message in which the RSTR is transported.
* **AValidClientCertificateThumbprint**: the thumbprint of a certificate which is used to imitate a valid client credential which will be accepted to receive a security token (may send an RST).
  + Any valid X509 certificate may be used, e.g. the following supplied certificate “CertificateAnvendersystem.p12”
* **MaximumTokenLifetime**: the maximum token life time of issued tokens in minutes.
* **BppValue**: a base-64 encoded value of a basic privilege profile (BPP) xml value. A sample of BPP can be found at [website folder]\Resources\bpp.xml. This xml value is included in an attribute in the issue Assertion.
* **SoapMessageLogLocation**: a folder to store the entire received request to this service and its response to client.
* **serilog:minimum-level**: specify the level of logging. Log files are stored in the Logs\ folder.

# Using The STS Test Stub

Browsing the following URL will present a greeting page, that shows how to invoke the service.

<https://adgangsstyringeksempler.projekt-stoettesystemerne.dk/STS/>

Sample code which also demonstrates can be found in the class:

Kombit.Samples.Consumer.Consumer

The following test case demonstrates how to call the STS and then use the issued token to call a service:

SendRstAndThenExecuteServiceServiceSuccessfully