OIOIDWS.Net

(Bootstrap Scenario)

Date: 14.07.2017

[1 Introduction 3](#_Toc487801064)

[1.1 Intended audience 3](#_Toc487801065)

[1.2 Prerequisites 3](#_Toc487801066)

[1.3 Disclaimer 3](#_Toc487801067)

[2 How the example code was constructed 4](#_Toc487801068)

[2.1 Modifications to the SAML metadata generation 4](#_Toc487801069)

[2.2 OIOIDWS library reference added 4](#_Toc487801070)

[2.3 Modifications to MyPage.aspx 5](#_Toc487801071)

[2.4 Building your own application 7](#_Toc487801072)

[3 Running the example application 8](#_Toc487801073)

[3.1 DNS name 8](#_Toc487801074)

[3.2 Keystores and certificates 8](#_Toc487801075)

[3.3 Running the WSP 9](#_Toc487801076)

[3.4 Accessing the example application 9](#_Toc487801077)

# Introduction

This document covers the details of the bootstrap-scenario functionality in the OIOIDWS.Net reference implementation, and acts as a companion document to the “Digst.Oioidws.WscBootstrapExample” example source.

The example source shows how to combine the OIOSAML.Net and OIOIDWS.Net libraries to perform the following

* Build a small SAML 2.0 Service Provider (SP) that also acts as a Web Service Consumer (WSC)
* How to request a token from the NemLog-in STS, where the request contains the bootstrap token presented by the end-user logging into the SP
* Use this token to call one of the existing reference WSPs in the OIOIDWS.Net reference implementation

This document does not cover the basics of OIOSAML and OIOIDWS, nor does it cover how to setup a WSC and a SP in the NemLog-in administration module.

## Intended audience

This document is written for developers, and while the example source will build and run without further modifications, the reader is expected to have experience with both .NET development and Windows management, as some steps require starting Visual Studio, configuring the Windows Certificate Store and IIS.

## Prerequisites

The source code uses .NET 4.5, and is configured to run on an IIS application server. The reader should have .NET, Visual Studio and an IIS application server installed.

## Disclaimer

The Danish Agency for Digitisation provides the reference code as is and assumes no responsibility for the code by service providers. Service Providers should understand the limitations of the code and deal with these according to their own needs.

# How the example code was constructed

The example code can be found in the OIOIDWS.Net distribution source, specifically in the folder

\Examples\Digst.Oioidws.WscBootstrapExample

The example source is based on the WebsiteDemo project found in the OIOSAML.Net 1.7.13 distribution. The following modifications where made to the WebsiteDemo project to make it perform the intended OIOIDWS calls.

## Modifications to the SAML metadata generation

We need to make sure the SAML metadata file generated by the application requests the DiscoveryEPR attribute from the NemLog-in Identity Provider, otherwise we will not get the bootstrap token that we need. In Web.config the following line was added to the SAML20Federation section

<SAML20Federation xmlns="urn:dk.nita.saml20.configuration">

.... [ snip ] ...

<RequestedAttributes>

.... [ snip ] ...

<att name="urn:liberty:disco:2006-08:DiscoveryEPR" isRequired="true" />

</RequestedAttributes>

.... [ snip ] ...

</SAML20Federation>

When registering the metadata file at the NemLog-in administration module, remember to double-check that this attribute is part of the requested attributeset, otherwise no bootstrap token will be issued at user-login.

## OIOIDWS library reference added

We need the following OIOIDWS libraries to perform the required calls to the NemLog-in STS and the WSP

* Digst.OioIdws.Common
* Digst.OioIdws.LibBas
* Digst.OioIdws.OioWsTrust
* Digst.OioIdws.Wsc

These where added as project references to the example source to make development and debugging easier, but could be added as NuGet dependencies in a production setting.

## Modifications to MyPage.aspx

In the OIOSAML.Net WebsiteDemo project, there already existed an ASPX page that requires the user has logged on through the NemLog-in Identity Provider.

In the example code, this page was modified to show two additional buttons

* “Call Service with bootstrap token”
* “Call Service without bootstrap token”

With corresponding handlers in the MyPage.aspx.cs file.

The latter is added just to show the difference between the bootstrap-scenario and the ordinary signature-scenario. When clicking any of the buttons, the handler will call the NemLog-in STS to fetch a token, and then use it to call the Web Service Provider.

When clicking the “Call service with bootstrap token” button, the bootstrap token is pulled from the users’ attribute-set, and added as part of the call to the STS

// pull the token from the current logged in users token

string rawToken = Saml20Identity.Current["urn:liberty:disco:2006-08:DiscoveryEPR"][0].AttributeValue[0];

// convert to SecurityToken

byte[] raw = Convert.FromBase64String(rawToken);

var reader = new XmlTextReader(new StreamReader(new MemoryStream(raw)));

SecurityTokenHandler handler = new Saml2SecurityTokenHandler();

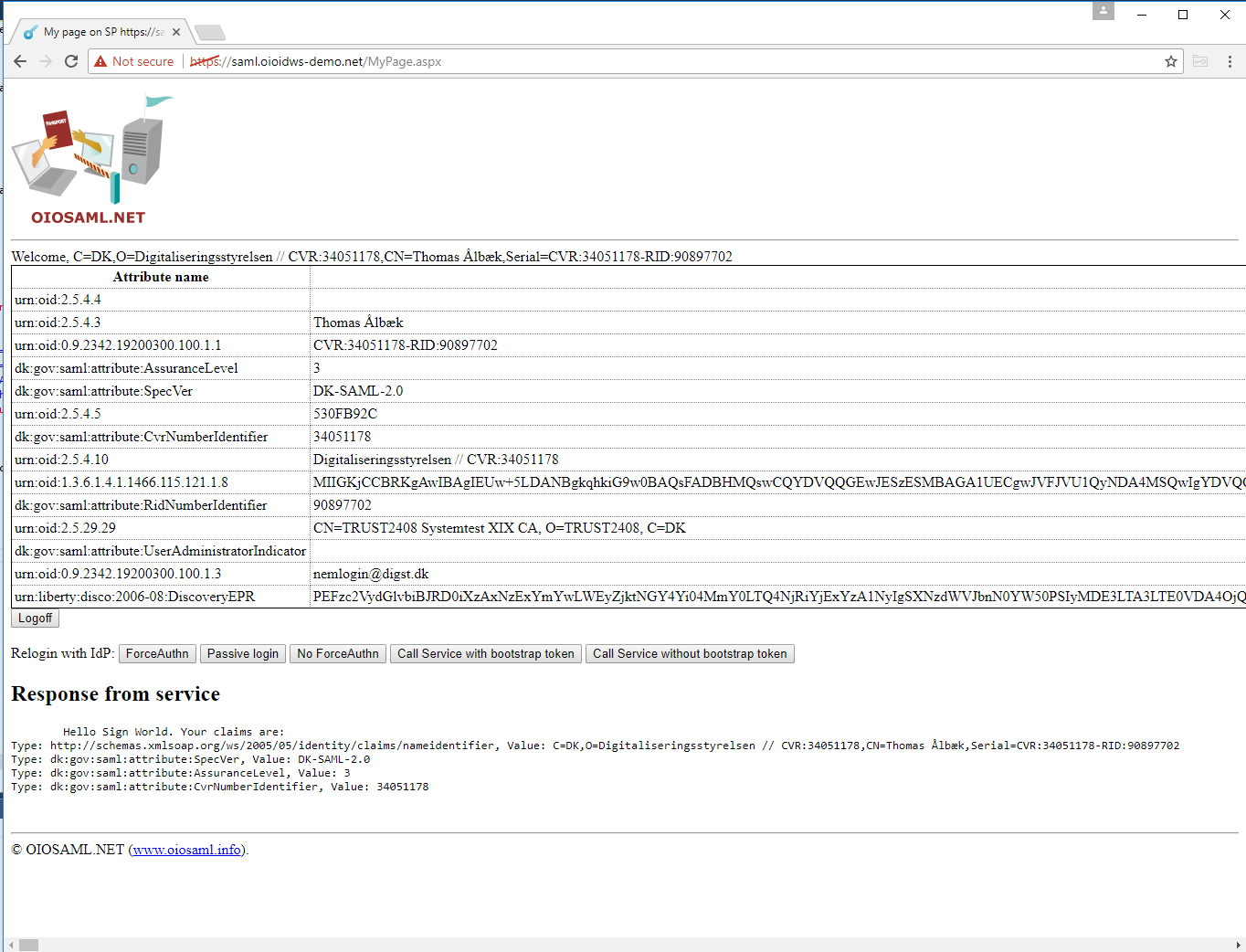
handler.Configuration = new SecurityTokenHandlerConfiguration();

SecurityToken bootstrapToken = handler.ReadToken(reader);

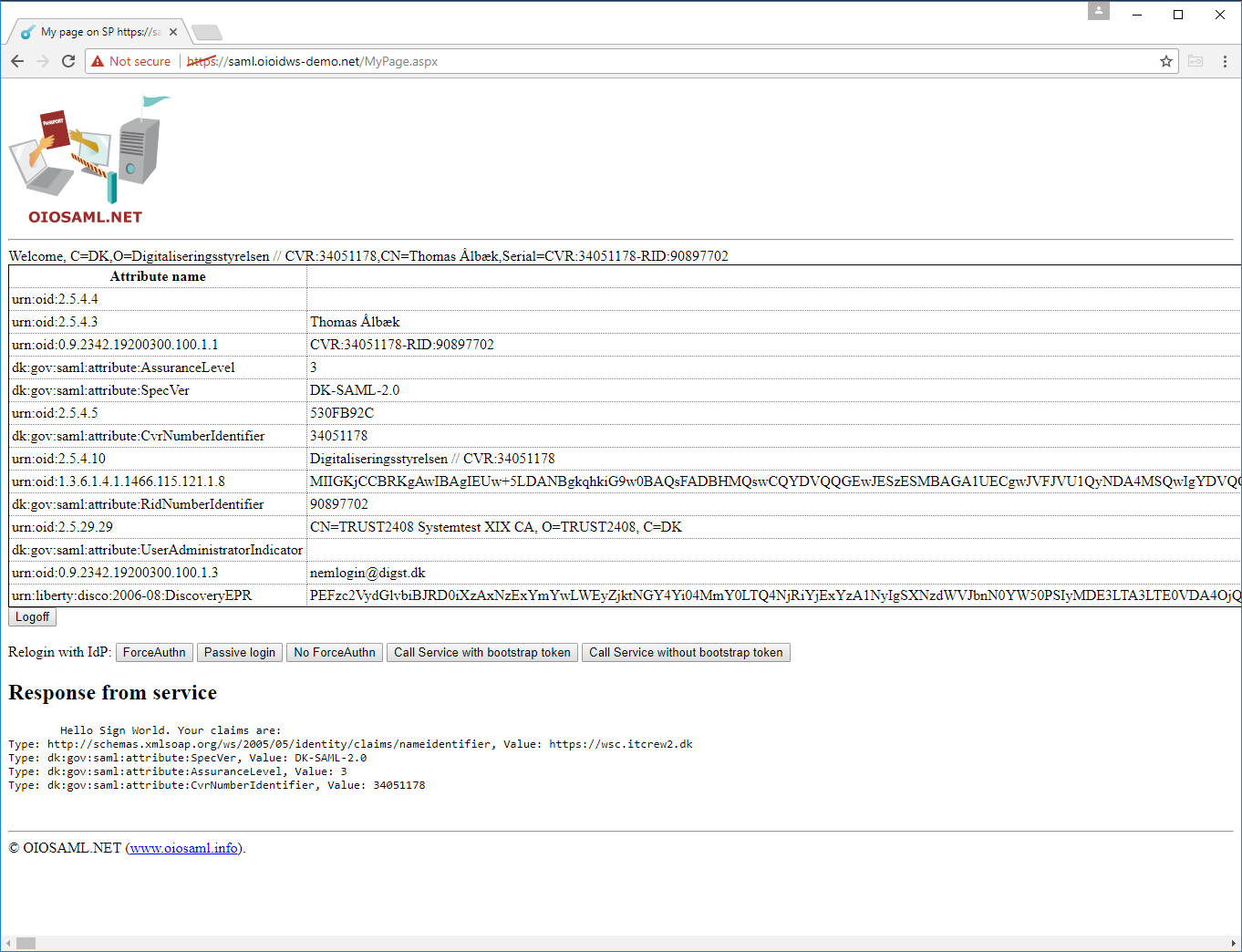
The OIOIDWS library exposes two methods for fetching a token, GetToken() and GetActAsToken(), depending on what case we are in, we will either use the GetToken() method (signature scenario without a bootstrap token), or the GetActAsToken() method call (bootstrap scenario).

The WSP that we call will return some of the information that it reads from the presented token, among these the identity of the caller. This information is printed on the HTML page.

Note that when running the bootstrap scenario, the end-users’ identity is what the WSP sees, whereas in the signature scenario, it is the identity of the WSC that is seen by the WSP.



**Bootstrap scenario**



**Signature scenario**

## Building your own application

To combine the OIOIDWS.Net and OIOSAML.Net libraries, you must perform these three additional steps (besides the steps required to use the OIOIDWS.Net and OIOSAML.Net libraries)

1. Make sure to register your application metadata with the DiscoveryEPR attribute request
2. Use the GetActAsToken() method instead of the GetToken() method, supplying the bootstrap token as argument
3. Make sure that your SP and your WSC registrations in the NemLog-in Administrationmodule are registered with the **SAME** certificate. The NemLog-in STS will only issue a token if the supplied bootstrap token is issued to a SP that uses the same certificate as the WSC requesting the token.

The OIOIDWS.Net library will take care of the rest.

# Running the example application

The example application is based on the WebsiteDemo application from the OIOSAML.Net source distribution, and the steps required to deploy this example application in an IIS application server, are the same as those required to deploy the WebsiteDemo application. The reader should consult the OIOSAML.Net documentation for these details.

## DNS name

The example application has been registered in the NemLog-in Administrationmodule with the following DNS name, meaning that all access to the application must use this domain name

https://saml.oioidws-demo.net

Open the file “C:\Windows\System32\Drivers\etc\hosts”, and add the following record (which must be done with an editor running as Administrator)

127.0.0.1 saml.oioidws-demo.net

This will map the DNS name to localhost.

## Keystores and certificates

Once deployed, the following certificates and keystores must be installed in the Windows Keystore

* STS.cer
* WSC (Java ref. TEST (funktionscertifikat)).p12
* WSP (eID JAVA test (funktionscertifikat)).p12

The passwords for the keystores are “Test1234”, and the certificates should be installed under “Local Computer/Personal”, and suitable access should be granted to the private keys (usually access needs to be granted to the system user running the IIS application).

These certificates and keystores can be found in this folder in the OIOIDWS.Net source distribution

\Misc\Certificates

## Running the WSP

The WSP should be running. The example application is configured to run against the WSP found in this project in the OIOIDWS.Net source distribution

\Examples\Digst.OioIdws.WspExampleNuGet

Running the project from Visual Studio will work – the example application will attempt to call it on 127.0.0.1 port 9091, which is the default setting for the WSP.

Note that some versions of Windows might prohibit binding to this port, unless access has been granted. This can be done using netsh, and the following command (in Powershell as Administrator) will allow anyone to run an application on that port

netsh http add urlacl url=http://+:9091/ user=Everyone

## Accessing the example application

Once the application has been deployed in IIS, and all the certificates are in place, and the WSP is running, open a browser and navigate to

https://saml.oioidws-demo.net

accessing the page should show the default frontpage from the OIOSAML demo website, with the option of accessing the restricted page (MyPage.aspx). Doing that will trigger a NemLog-in redirect, from where a user must login using an OCES certificate from the NemLog-in test environment.

After a successful login, the user ends up on MyPage.aspx, where he has the option of clicking one of the two mentioned buttons, which will trigger a call to the NemLog-in STS and from there, a call to the WSP.