GW Plugin Guide

Revision History

Version	Date	By	Comments
1	06/01/10	Søren Mors	Initial release of document
2	24/08/10	Brian Graversen	Added UserAuthorizationCode as an optional paramter

Introduction

The GW Plugin is an action plugin for the SignOn Library framework, that facilitates issuing IDCards from the STS, through a SOSI Gateway. In order to do this, it requires a PKI-enabled authentication plugin. Read the document “Sign-On projektet: A national architecture for SignOn” for details on entire SignOn Library framework.

Purpose of the plugin

This plugin, together with the SignOn Library and a PKI-enabled authentication plugin, allows client applications to handle logins to the national service infrastructure. The plugin creates an IDCard, submits it to a Sosi gateway and uses a PKI plugin to sign the digest of the IDCard returned from the gateway. It then submits the signed digest back to the gateway, which submits it to the STS. This allows the client application to use the gateway without having to handle the details of PKI usage.

Quick-guide: Using the plugin

We assume that the reader has read the document “Using the SignOn library”, and knows how to configure and call the SignOn method. This section will describe only the settings that are relevant for the GW Plugin, and will show sample code in C#. If the SignOn Library is used from other languages, the language-specific changes should be made to the code below before using it. Note that all the parameters just contain dummy data, which should be replaced with actual production data.

```
SignOnRequest request = new SignOnRequest();

// These parameters are all required for the use of the GW Plugin
request.AddInitialContext(GwCtxKey.CareProviderId, "19343634");
request.AddInitialContext(GwCtxKey.CareProviderName, "TestProvider");
request.AddInitialContext(GwCtxKey.ITSystemName, "SOSITEST");
request.AddInitialContext(GwCtxKey.GWAddress, http://10.11.10.223:8080/sosigw/service/sosigw");
request.AddInitialContext(GwCtxKey.Issuer, "SOSITEST");
request.AddInitialContext(GwCtxKey.MedcomRole, "læge");
request.AddInitialContext(GwCtxKey.IDCardHandle, "MyIdCardHandle");


// this binary runs from the bin\debug folder, and a relative path is used
request.AddPlugin(@"lib\GWPlugin.dll", -1, false);
```

The configuration parameters for the plugin are the following.

Name	Explanation	Mandatory
GwCtxKey.IdCardHandle	The handle used for the generated IDCard. A GUID will be used instead if this is not specified. The value is returned in the SignOn Context regardless.	No
GwCtxKey.Issuer	The Issuer of the generated IDCard	Yes
GwCtxKey.CareProviderId	The CVR number of the Care Provider. Defaults to the CVR number in the certificate used. Since the STS verifies that the cvr number is correct, the default is most likely correct.	No
GwCtxKey.CareProviderName	The name of the care provider	Yes
GwCtxKey.ITSystemName	The name of the IT System used	Yes

Name	Explanation	Mandatory
`GwCtxKey`.GWAddress	The URL of the Sosi GW service	Yes
`GwCtxKey`.UserAuthorizationCode	The users authorization code (if available)	No

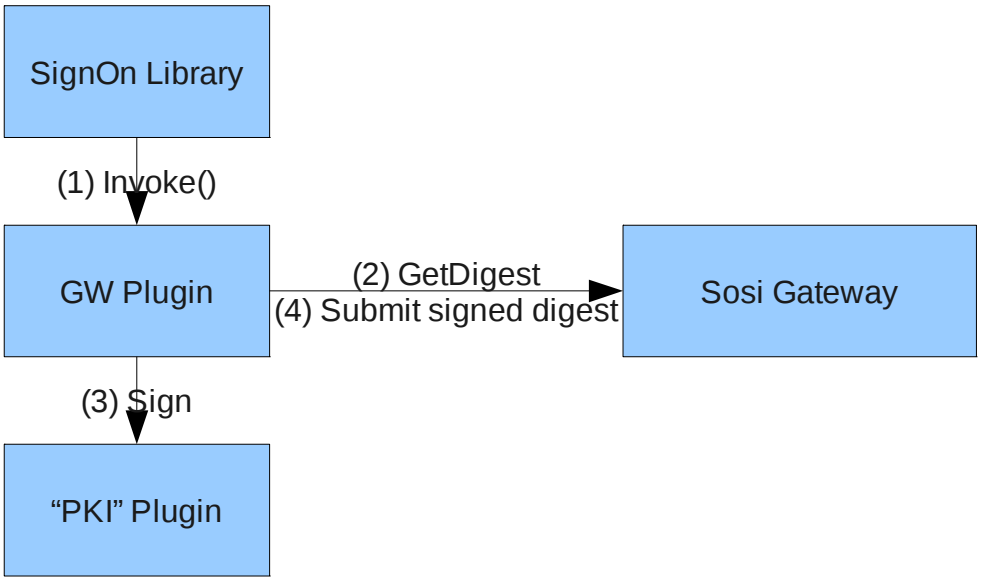
Flow Handling
There are no errors that can occur, which a client application can reasonably be expected to handle, other than telling the user that an error has occured. Therefore all errors are reported as plugin errors.

The generated ID Card
The IDCard generated mostly contains data from the configuration parameters detailed above. In addition to this the name of the user is taken from the Common Name of the certificate used by the PKI plugin.

Dependencies on the PKI-enabled authentication plugin
The GW Plugin cannot function without an initialised PKI-enabled authentication plugin. There must therefore be such a plugin configured to be invoked before the GW Plugin. Since the certificate of the user must be presented to the gateway, the GW plugin requires the authentication plugin to have placed a certificate in the application context.

Detailed description of the plugin
The plugin only exposes the two methods which are required by the signon library, Invoke and Release.

The following diagram shows the components that the GW Plugin communicates with, and the order in which it communicates with them.



As can be seen the plugin gets its Invoke method called by the signon library. It then creates an IDCard which it submits to the gateway, and gets a digest back. The PKI-enabled authentication plugin, which should be in the application context at this time, is used to sign the digest. Finally the signed digest is submitted to the gateway, which handles further communication with the STS.

Since the plugin doesn't have any sensitive information, it doesn't do anything when release is called, except from setting a couple of references to null, so the garbage collector can collect them.