Build Notes

Revision History

Version	Date	By	Comments
1	21/01/10	Brian Graversen	Initial release of document

Introduction

Building the entire SignOn Library application requires several steps, since the product contains both C# assemblies, a Java application, JNI code, and most of the compiled binaries needs to be signed as part of the build procedure.

1. **Locate the keystore needed to sign the binaries, and extract the certificate**

To secure the trust chain between all the components of the library, a digital signature is used on all components. For this purpose a keystore containing keys/certificates is used (likely a PFX/PKCS#12 file). For testing purposes, the pluginsigner.pfx file found in the Binaries folder of the source can be used.

The certificate (the public part of the keystore) should be extracted, since it should be added to the sourcecode before compiling it. Either use something like keytool to do this, or simply import the PFX file into Windows, and then extract the certificate using Internet Explorer (Tools -> Internet Options -> Content -> Certificates). This will result in a .cer file. You will need both the .cer file and the keystore file (a .pfx file).

2. **Compile the SignOn Library project using Visual Studio 2008+**

The easiest way to compile everything is to open the solution file (SignOnLibrary.sln) and simply perform a “rebuild solution” with the “Release” target. Before compiling everything, open PluginLoader.cs and replace the hardcoded TEST certificate with the contents of the .cer file (base64 encoded the content).

3. **Compile any 3rd party plugins**

The SignOn Libary solution file only compiles the SignOn Library itself, and the GW/CAPI plugins. Any other plugins that should be bundled with the final product should also be compiled at this step.

4. **Sign the C# assemblies**

Use the keystore to digitally sign the DLL output from the previous steps.

5. **Compile the JNI code**

Open the SignOnJNI project found under the Java Wrapper folder (c-code) and compile the project. Note that this project may not compile out-of-the-box, since it requires Java JDK to be installed on the machine, and that the headers from the Java JDK is referenced in the project (you may need to modify the include path).

6. **Compute the SHA-1 digest of the JNI code**

Compute a sha-1 digest on the JNI dll, we will need to add this to the Java Wrapper before compiling it. The Java Wrapper has a small program to do this if you do not have an easy way to do this yourself (found in dk.sosi.signOnLibrary.DigestCalculator.java in the Java Wrapper project).

7. **Compile the Java wrapper**

Before compiling the code, open SignOnFacade.java source file and replace the digest value with the one found in the previous step. To compile the project, use the Ant build.xml script. Note that the build.xml script will use the keystore file to sign the JAR file, so modify the build.xml file first,

so it points to the correct keystore (and change the password).

8. **Bundle the binaries and documentation**

Finally grab all the documentation, and bundle the binaries and the documention in a zip-file. The release should be added to SVN and given a version.