IT PRACTICE

OpenSign Extensions

For Mac OS X and non-javascript integration

IT Practice
Kronprinsessegade 54
DK-1306 København K

T: +45 33 36 46 60
F: +45 33 36 46 61
E: itp@it-practice.dk

Author: Jesper Nielsen
Date: 30. November, 2003

CONTENT

INTRODUCTION

This document describes how to use the IT-Practice extensions to the OpenSign applets in order to facilitate Mac OS X and non-javascript integration on any platform available.

REQUIREMENTS

The extension code was successfully tested on the following platforms and versions.

OpenSign version	Operating system	Browser	Java version
v1.0.2	Linux	Galeon v1.3.5	Sun java-plugin v1.4.2_b28
v1.0.2	Linux (Debian)	Galeon v1.3.10	Blackdown java-plugin v1.4.1_01
v1.0.2	MacOS X 10.2.8	Mozilla 1.5	java v1.3.1
v1.0.2	MacOS X 10.2.8	Mozilla 1.6	java v1.3.1
v1.0.2	MacOS X 10.2.8	Mozilla Firebird 0.7	java v1.3.1
v1.0.2	MacOS X 10.2.8	Mozilla Firebird 0.8b	java v1.3.1
v1.0.2	MacOS X 10.2.8	Netscape 7.1	java v1.3.1
v1.0.2	Microsoft Windows XP (Prof Ed.)	Internet Explorer v6.0	Microsoft Native JVM
v1.0.2	Microsoft Windows XP (Prof Ed.)	Internet Explorer v6.0	Sun java-plugin v1.4.2_01

- The extension code should work on any existing OpenSign supported platform.

USING THE CODE

The code is an extension of the existing two applets Logon and Sign applets from the `org.openoces.opensign.client.applet` package. Two applets are extended from those. They are names Logon and Sign and located in the `dk.itp.openoces.opensign.client.applet` package . In order to use the extension applets simply modify the existing applet presentation HTML to point to those two applets instead of the original OpenSign ones.

Simply using the applets will not change the functionality of the applets. In order to activate the new functionality the new applets supports a set of new applet parameters that needs to be supplied to them. These new applet parameters can co-exist with the existing parameters and the original OpenSign applets can run with these parameters without any problem.

New applet parameters

The new applet parameters used to activate the new non-javascript functionality are these:

Parameter	Description
opensign.doappletrequest	When set to "true" it tells the extension applets to use the new sign/logon procedure
opensign.doappletrequestonmac	When set to "true" it tells the extension applets to use the new sign/logon procedure automatically when running on a mac
opensign.verifieruri	The URI to the verifier that handles the logon/sign request
opensign.doappletrequestonmac	When set to "true" it tells the extension applets to use the new sign/logon procedure automatically when running on a mac
opensign.verifieruri	The URI to the verifier that handles the logon/sign request (also used for posting error and alert messages from the applet which can be stored in the users session and presented later)
opensign.canceluri	URI to the service called when the user cancels the sign/logon request
opensign.erroruri	Called if an error occurs during certificate handling
opensign.alerturi	If an alert occurs (like invalid password) this URI is called
opensign.verifiedokuri	If a successsfull call to the "opensign.verifieruri" was made this URI is the one forwarded to
opensign.verifiederroruri	If the verifier does not return an OK message the user is forwarded to this URI
opensign.message.name	The formfield name used to post the logon/sign message string
opensign.result.name	The formfield name used to post the result message from the applet (ok/cancel/error/alert)
opensign.verifieruri	The URI to the verifier that handles the logon/sign request
opensign.cookiecount	The number of cookies contained in the applet parameters.
opensign.cookie.X.name	The name and value of cookie number X (the index X must start with the number 1 and be

opensign.cookie.X.value	continous from there)
opensign.formdata.count	The number of formfields contained in the applet parameters (used if it is required to send on other information to the verifier other than the standard logon information).
opensign.formdata.X.name opensign.formdata.X.value	The name and value of formfield number X (the index X must start with the number 1 and be continous from there)

Applying cookies to the applet
In order to apply cookies to the code the applet content page must be dynamic (a servlet, a JSP or a CGI program or similar). If for example a servlet or a JSP is used to generate the applet presentation HTML the following code is an example on how to generate the applet parameters.

```
…
…
  Cookie cookies[] = request.getCookies();
  if( cookies != null ) {
    out.println(getParamPair("opensign.cookiecount",
                             ""+cookies.length));
    for( int i=0; i<cookies.length; i++ ) {
      out.println(getParamPair("opensign.cookie."+(i+1)+".name",
                  cookies[i].getName()) );
      out.println(getParamPair("opensign.cookie."+(i+1)+".value",
                  cookies[i].getValue()) );
    }
  }
  else {
      out.println(getParamPair("opensign.cookiecount", "0"));
  }
…
…
private String getParamPair(String name, String value) {
  return "<param name=\""+name+"\" value=\""+value+"\">";
}
```

THE NEW REQUEST FLOW

Using the applet extensions puts a new flow on the system. Instead of the verifier service actually being able to redirect the user to the first page of the application (or actually be the first page of the application) it is now a “simple” verifier service that must return HTTP responsecode 200/OK when the verification is ok and something else like for example 501/SERVICE UNAVAILABLE when the verification fails. After that the applet will redirect the user to either the URI contained in the opensign.verifiedokuri or the opensign.verifiederroruri parameters. If an error occurs the server application should by it self be able to present the correct error message to the user (by using his/hers session).

If the user clicks cancel the applet will immediately forward the user to the URI contained in the opensign.canceluri parameter.

If an error (or alert) occurs during the signing process in the applet (wrong password, invalid certificate or similar) the applet will post the error message to verifier URI contained in the opensign.verifieruri parameters and then redirect the user to either the URI contained in the opensign.erroruri or the opensign.alerturi parameters. The verifier service on the server can tell the difference between a logon/sign request and an error/alert request by checking on the “result” message in the incoming data which will contain the values “ok”, “error” or “alert”.

Sample applet parameters

This is a sample of how the new applet parameters could be set up.

```
applet name="signing_applet"
code="dk.itp.openoces.opensign.client.applet.Logon" width="500"
height="200" archive="/demo/opensign.jar" mayscript>

<param name="locale" value="en,US">
<param name="cabbase" value="/demo/OpenSign.cab">
<param name="key.store.directory" value="null">
<param name="background" value="255,255,255">
<param name="socialsecuritynumber" value="yes">
<param name="optionalid" value="no">
<param name="opensign.doappletrequestonmac" value="true">
<param name="opensign.verifieruri" value="/demo/servlet/Verifier">
<param name="opensign.canceluri" value="/demo/index.html ">
<param name="opensign.erroruri" value="/demo/servlet/showlogon">
<param name="opensign.alerturi" value="/demo/servlet/showlogon">
<param name="opensign.verifiedokuri"
       value="/demo/servlet/Verifier">
<param name="opensign.verifiederroruri"
       value="/demo/servlet/showlogon">
<param name="opensign.message.name" value="message">
<param name="opensign.result.name" value="result">
<param name="opensign.cookie.1.name" value="JSESSIONID">
<param name="opensign.cookie.1.value"
       value="24778657HJG723B27FQCEGWHRAZXYUREURTDW46SDUI4TNX35YS">
<param name="opensign.formdata.count" value="2">
<param name="opensign.formdata.1.name" value="samplename1">
<param name="opensign.formdata.1.value" value="samplevalue1">
<param name="opensign.formdata.2.name" value="formfield2name">
<param name="opensign.formdata.2.value" value="formfield2value">
</applet>
```

The output is generated by a showlogon servlet which will also be redirected to when an error or alerts occurs. The showlogon servlet then presents the error/alert above the applet (the message is received through the verifier servlet and placed in the session).

With these parameters the the applet will automically use the new logon procedure on a Mac client. But on other clients (Windows, Linux, etc.) the javascript logon procedure will be used since the “opensign.doappletrequest” is not set. Setting this would allow the applet to use the non-javascript logon procedure on all platforms (givin a uniform way of handling logn/sign).k