WebSigner Plug-in

Introduction

The WebSigner plug-in is used to create digitally signed messages in web browsers.

Plug-in Activation

The following <OBJECT> tags are used to activate the plug-in in a web browser:

ClassID: 6969E7D5-223A-4982-9B79-CC4FAC2D5E5E (**Windows only**)

ProgID: Nexus.SignerCtl (**Windows only**)

Activation MIME type: application/x-personal-signer

Internet Explorer

Note: This section applies to the Windows platform only.

In Internet Explorer, the plug-in is implemented as an ActiveX control. It is activated using the <OBJECT> tag, supplying none, some or all parameters using the <PARAM> tag.

Example of an ActiveX control activation:

```
<OBJECT ID="Signer" CLASSID="CLSID:6969E7D5-223A-4982-9B79-CC4FAC2D5E5E">
<PARAM NAME='CharacterEncoding' VALUE='UTF8'>
<PARAM NAME='DataToBeSigned' VALUE='Sign%20this.'>
<PARAM NAME='PostURL' VALUE='https://server.com'>
</OBJECT>
```

Due to the Eolas Patent, chapter “Appendix A - Eolas Patent” on page 125 supplies more information on how to activate a plug-in. If the WebSigner plug-in is started using direct activation, we recommend solution 2, while solution 1 is preferred if the plug-in is scripted.

At this stage, it is not necessary to set any parameters as they can be set later using the script functions.

It is also possible for the web server to use scripting to silently detect if the plug-in is installed in the client.

Example of a script to activate the plug-in:

```
<SCRIPT language="JavaScript">
try {
  var xObj = new ActiveXObject("Nexus.SignerCtl");
  if(xObj) {
    document.writeln("Object installed.");
  }
} catch (e) {
  document.writeln("Object not installed.");
}
</SCRIPT>
```

It is recommended that the <OBJECT> tag is used to create the object to be used for signing. It is possible to use the plug-in object created using "new ActiveXObject(Nexus.SignerCtl)" but this object will not be initialized correctly by Internet Explorer. In other words, the browser functions needed for the signing operation can not be used i.e., the plug-in will not be able to post the signature by itself, and the DataURL parameter cannot be used. Furthermore, Internet Explorer will not be set as a parent window to the signing window.

Mozilla-Based Browsers

In Mozilla-based browsers, the plug-in is implemented using the NPAPI. It can be activated using the <OBJECT> tag by supplying some or all parameters using the <PARAM> tag. It must be noted that it is done in a different way than for Internet Explorer. The ClassID is not used to identify the plug-in, but rather the activation MIME type as defined above.

Example of how to activate the Mozilla-based browser plug-in using the <OBJECT> tag:

```
<OBJECT id="signer" type="application/x-personal-signer">
<PARAM NAME='CharacterEncoding' VALUE='UTF8'>
<PARAM NAME='DataToBeSigned' VALUE='Sign%20this.'>
<PARAM NAME='PostURL' VALUE='https://server.com'>
</OBJECT>
```

Scripting can be used by the web server to decide whether the plug-in is installed in the browser by checking if the activation MIME type is registered.

Example of a script to activate the plug-in:

```
<SCRIPT language="JavaScript">
if(navigator.plugins) {
  if (navigator.plugins.length > 0) {
    if (navigator.mimeTypes && navigator.mimeTypes["application/x-personal-
signer"]) {
      if (navigator.mimeTypes["application/x-personal-signer"].enabledPlugin)
{
        document.writeln("Plugin installed");
      }
    }
  }
}
</SCRIPT>
```

Parameters

The following parameters are used in the WebSigner interface. They are case sensitive if nothing else is stated explicitly.

Parameter	Explanation
Mime-type	The MIME type of the data to be signed. This type effects which application program to start if the **View** button in the WebSigner window is clicked. MIME types were originally specified in RFC 1341 but improvements have been made in other documents like RFC 1521 and RFC 1522.
CharacterEncoding	Sets the character encoding of the data to be signed, if relevant for the chosen MIME type. The only supported character encodings are "UTF8" and "platform", where "platform" is the platform's default character encoding. The parameter is **optional** and if no character encoding is given, default will be "platform". For Mac OS X, "platform" will be interpreted as ISO 8859-1. "UTF8" is recommended.

Parameter	Explanation
Format	Defines the format of the output data. Currently, only the PKCS #7 signed-data content type format is supported. After creation, the signature will be URL-encoded for sending as a web form element. The value "PKCS7SIGNED" specifies this format. **Note:** For full backwards compatibility with versions of Personal prior to 3.0, the seconds may be removed by adding "_NoSeconds" to the Format parameter: "PKCS7SIGNED_NoSeconds", "PKCS7SIGNED_Attached_NoSeconds", or "PKCS7SIGNED_Detached_NoSeconds". We also recommend that you use "PKCS7", however, "PKCS#7" can still be used for backwards compatibility. As an option, the signed data may be either included in the resulting signature, by specifying "PKCS7SIGNED_Attached", or excluded, by specifying "PKCS7SIGNED_Detached". The parameter is optional and if no format is given, default will be "PKCS7SIGNED_Attached".
HashAlg	Optional parameter specifying which hash algorithms to use in signatures. Possible values are "MD5", "SHA1", "SHA224", "SHA256", "SHA384", "SHA512", "RIPEMD128", and "RIPEMD160". Default is "SHA1".
Filename	An optional file name to be used as default in the Save dialog when the user presses the **Save...** button in the WebSigner dialog box.
WindowName	If present, this parameter specifies the name of the window or frame used to display the server response to the HTML post from WebSigner. WindowName is optional. If omitted, it defaults to the window (or frame), that WebSigner was activated in, i.e., "_self".
DataToBeSigned	Used as the data to be signed when the data is embedded into the HTML page. The data to be signed must be sent URL-encoded. WebSigner will decode the message, present it in the signature dialog box and sign the decoded message. **Note:** Either DataURL or DataToBeSigned is required, to make a signature. However, they do not have to be present when creating the plug-in object, but can be set later using script methods. DataToBeSigned overrides DataURL if both parameters are present.
DataURL	Defines the URL to the data to be signed. Either DataURL or DataToBeSigned is required, to make a signature. However, they do not have to be present when creating the plug-in object, but can be set later using script methods. If both are present DataToBeSigned overrides DataURL. **Note:** To avoid caching problems, this data should always be sent with a "no-cache" pragma from the server.

Parameter	Explanation
PostURL	Defines the URL to which WebSigner will post the signed data. If this parameter is not defined, WebSigner will sign the data and make it available for later retrieval using the GetSignature script function, but it will not post the signature.
PostParams	If PostURL is set then the URL-encoded string with parameters is posted back.
SignReturnName	Defines the name of the form field to contain the signature posted by WebSigner. SignReturnName is optional and if not present the default is "SignedData".
DataReturnName	Defines the name of the form field to contain the unsigned data posted by WebSigner. The original data will be returned. DataReturnName is optional and if not present, WebSigner will not post the unsigned data.
VersionReturnName	Returns the version of Personal when posting the signature. See DataReturnName and SignReturnName.
Issuers	Defines the filter criteria based on Issuers used to reduce the user's certificate choices when signing. Specific certificate attribute search strings can be specified, separated by "," or ";" where comma is interpreted as logical AND and semicolon as logical OR. The following X.500 attribute abbreviations are available: "cn", "g", "s", "t", "ou", "o", "email", "i", "sn", "street", "l", "st", "c", "d", and "dc" . In addition, OIDs can be used. Example 1: The search string "cn=Our CA" will display all certificates issued by CAs with the common name "Our CA". Example 2: 2.5.4.6=SE will filter out all Swedish certificates.
Subjects	Defines the filter criteria based on Subjects used to reduce the user's certificate choices when signing. See *Issuers.*
ViewData	Switches between two default signature dialog boxes handled by WebSigner. The value "false" causes WebSigner to activate a small dialog box where the data to be signed is optionally viewed in a separate viewer application, which depends on the MIME type parameter. The value "true" causes WebSigner to activate a larger dialog box, which, in addition to the possibility to view the data separately, also displays the data in a text area within the dialog box. **Note:** The data may be incorrectly displayed in the text area if it contains certain control characters such as, e.g., the Null character. The default value is "true".
Base64	If the signature is to be Base64-encoded before it is URL-encoded, this parameter must be set to "true". *Base64* is optional and case insensitive. If omitted or set to "false", no Base64-encoding will be performed.

Parameter	Explanation
IncludeCaCert	Possible values are "true" and "false" (default is false). Will include the CA certificate chain (except the rootCA) in the signature, if available. *IncludeCaCert* is case insensitive.
IncludeRootCaCert	Possible values are "true" and "false" (default is false). Will include the Root CA certificate of the certificate chain in the signature, if available. *IncludeRootCaCert* is case insensitive.
UseBranding	An optional parameter that specifies if WebSigner should be branded or not. If the parameter is set to true, WebSigner GUI will be branded if Personal installation is branded. If the parameter is false, WebSigner GUI will not be branded, even if the installation is branded. If this parameter is not present, it defaults to true.

Scripting

The WebSigner profile may also be scripted using JavaScript or VB Script. The following functions are available in Personal:

- int Set<parameter name>(String value)
- int Sign()
- String GetVersion()
- String GetSignature()
- String GetErrorString()

int Set<parameter name>(String value)

Parameters (see page 19) are set using the following:

```
int Set<parameter name>(String value)
```

The only exception is MIME type, which uses SetMimeType. The return value will always be 0.

int Sign()

int Sign() pops up the signature dialog box and signs the data buffer. If *int Sign()* is successful, it returns 0 otherwise -1 is returned.

If the PostURL parameter is set, the plug-in will post the signature by itself. The signature can also be retrieved with *GetSignature()*, independent of it being posted. The retrieved signature can be posted using script.

Note: If Mozilla-based broswers are used and the DataURL parameter is specified, the *Sign()* method immediately returns 0. However, the data to be signed must be downloaded before the signing can take place. This means that the signature is not available using *GetSignature()* as the data has not been downloaded and signed.