Widget Signing
Input Paper for W3C Web Application Formats WG
Olli Immonen, Nokia
Updated version: 27.4.2007

Introduction

Widgets 1.0 requirements document [Widget Reqs] states that widgets packaging format should allow authors (and other publishers) digitally sign their widgets so that a user agent can verify the authenticity and the integrity of the package. The current draft Widgets 1.0 [Widgets 1.0] states that the packaging format would be a ZIP archive.

In this paper, we propose a method and format to apply a digital signature in the widget package. The basis for this desing (apart from the basic purpose of digital signing) is following

1. Signatures should be easily applied for the ZIP archive. The input of the signing process and prerably also the output should be ZIP archive. I.e. the signed and the unsiged package should essentially have the same structure. Somebody who has no use of the signature should easily be able to omit that.
2. Signature should be applicable for both textual content (HTML, XML, js, etc) and binary (GIF etc).
3. The basic need would be to sign and verify the entire package. However, it may also be envisioned that only part of the package would be signed and other parts would remain unsigned.
4. Again, the basic need would be to sign the package by a single signer. However, it could be envisioned that there could be multiple signers (e.g. first signed by a developer, then by a distributor). Different signers could even sign different parts of the package. Semantics of these cases are outside the scope of this paper.
5. Developers should be easily able to generate signatures. Tools to do that should be easily made and used.
6. The user agent should be able to verify the signature during download on the fly, i.e. without first having to save it locally and then verifying it separately. (Of course separate verification should also be possible).
7. The user agent should be able to remove the signature after verification, or save it for the records.
8. A standard digital signature format should be used, in practise either CMS or XML Signature.

CMS (Cryptographic Message Syntax http://en.wikipedia.org/wiki/Cryptographic_Message_Syntax) is based on the syntax of PKCS#7, which in turn is based on the Privacy-Enhanced Mail standard. It uses ASN.1 as a presentation format and binary encoding. The newest version of CMS (as of 2004) is specified in RFC 3852. CMS/PKCS#7 is used e.g. in S/MIME and in jar signing (http://java.sun.com/j2se/1.3/docs/guide/jar/jar.html). In addition to a file containing this binary CMS structure, jar signing uses a set of textual non-XML files (Manifest file and so called singature definition files).

XML Signature is a W3C recommendation [XML DSig], (http://en.wikipedia.org/wiki/XML_Signature) that defines an XML syntax for digital signatures. Functionally, it has much in common with PKCS#7 but is more extensible and geared towards signing XML documents. It is used by various Web technologies such as SOAP, SAML, and others.

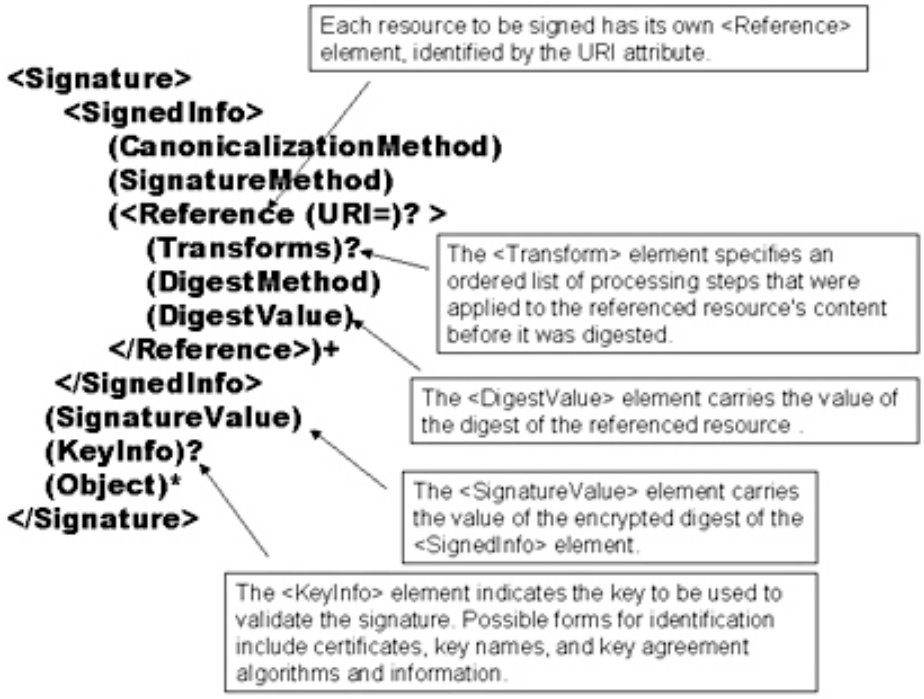
MIDP 2.0 defines a new way to sign a ZIP (jar) archive. The signature and necessary certifiate information is in a file (JAD) distributed separately from the ZIP file.

This proposal is based on using XML signature. Some reasons for that choice

- Because of using (textual) XML, it is more developer friendly than ASN.1 based CMS. (The cryptogaphic data is still (Base64 encoded) binary but the "human readable" part is textual.)

- An XML format shoud perhaps be preferred (over binary or non-XML textual formats) for a specification that is mostly using XML
- XML signature syntax has support for covering multiple resources (files or web resources)
- The design makes it possible to have the signature within the ZIP archive (unlike MIDP 2.0 signature).

The attached picture illustrates XML Signature structure. It is borrowed from [XML Dsig Intro]. Anyone not familiar with XML Signature (or digital signatures in general) is engouraged first take a look at this document.



XML signatures can be used to sign data – a resource – of any type, typically XML documents, but anything that is accessible via a URL can be signed. An XML signature used to sign a resource outside its containing XML document is called a detached signature; if it is used to sign some part of its containing document, it is called an enveloped signature; if it contains the signed data within itself it is called an enveloping signature.

This proposal uses the detached signature option of XML Signature. The signature is placed in a separate file, outside of the content that is being signed (i.e. the files in the ZIP). One could also think placing the signature e.g. in the config.xml manifest file, using an enveloped signature. However, this would make the verification procedure more complicated: during the verification, assuming that the config.xml file is also signed, the hash calculation should omit the signature part. This procedure, called enveloped signature transform, even if it is well-defined would be an extra complication.

In order to support signatures made by multiple parties, the proposal uses the Manifest element. This makes it possible to add new signatures without repeating all the references.

Description of the Design

Digital signature covers the necessary (in most cases all) files in in the ZIP. These files are considered as resources for the signing process. A resource has its own <Reference> element, identified by the URI attribute. The file name (or the file path inclding the directory structure) is used as the URI.

Signing Process

1. Identify resources to be signed

Identify the resources through a URI. These would be files in the ZIP archive or on the Web if applicable.

- “config.xml” would reference the manifest file in the root directory
- “index.html” would reference the HTML index file in the root directory
- “pictures/picture1.gif” would reference a GIF image in a subdirectury
- “http://www.abccompany.com/logo.gif" would reference a GIF image on the Web

2. Calculate the digest of each resource

Each referenced resource is specified through a <Reference> element and its digest (calculated on the identified resource and not the <Reference> element itself) is placed in a <DigestValue> child. The <DigestMethod> element identifies the algorithm used to calculate the digest.

```
<Reference URI="config.xml">
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
    <DigestValue>j6...8nk=</DigestValue>
</Reference>
<Reference URI="index.html">
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <DigestValue>lm...34=</DigestValue>
</Reference>
<Reference URI="pictures/picture1.gif">
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <DigestValue>pq...56=</DigestValue>
</Reference>
<Reference URI="http://www.abccompany.com/logo.gif">
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <DigestValue>rs...78=</DigestValue>
</Reference>
```

3. Collect the Reference elements

Collect the <Reference> elements (with their associated digests) within a <Manifest> element.

```
<Manifest Id="References">
    <Reference URI="config.xml">
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
        <DigestValue>j6...8nk=</DigestValue>
    </Reference>
    <Reference URI="index.html">
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>lm...34=</DigestValue>
    </Reference>
    <Reference URI="pictures/picture1.gif">
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>pq...56=</DigestValue>
    </Reference>
    <Reference URI="http://www.abccompany.com/logo.gif">
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>rs...78=</DigestValue>
    </Reference>
</Manifest>
```

3. SignedInfo

Create the <SignedInfo> element.

```
<SignedInfo>
    <CanonicalizationMethod
      Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <SignatureMethod
      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
    <Reference URI="#References">
      Type="http://www.w3.org/2000/09/xmldsig#Manifest">
        <DigestMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>725x3fVasdfvBGFGjhjyDSFvUk=</DigestValue>
```

```
    </Reference>
</SignedInfo>
```

The <CanonicalizationMethod> element indicates the algorithm was used to canonize the <SignedInfo> element. Different data streams with the same XML information set may have different textual representations, e.g. differing as to whitespace. To help prevent inaccurate verification results, XML information sets must first be canonized before extracting their bit representation for signature processing. The <SignatureMethod> element identifies the algorithm used to produce the signature value.

4. Signing

Calculate the digest of the <SignedInfo> element, sign that digest and put the signature value in a <SignatureValue> element.

```
<SignatureValue>MC0E…LE=</SignatureValue>
```

5. Add key information

If keying information is to be included, place it in a <KeyInfo> element. Here the keying information contains the X.509 certificate for the sender, which would include the public key needed for signature verification.

```
<KeyInfo>
 <X509Data>
  <X509Certificate>MIID5jCCA0+gA...lVN</X509Certificate>
 </X509Data>
</KeyInfo>
```

6. Enclose in a Signature element

Place the <SignedInfo>, <SignatureValue>, and <KeyInfo> elements into a <Signature> element. The <Signature> element comprises the XML signature.

```
<?xml version="1.0" encoding="UTF-8"?>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

<SignedInfo>
    <CanonicalizationMethod
      Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <SignatureMethod
      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
    <Reference URI="#References">
      Type="http://www.w3.org/2000/09/xmldsig#Manifest">
        <DigestMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>725x3fVasdfvBGFGjhjyDSFvUk=</DigestValue>
    </Reference>
</SignedInfo>

<SignatureValue>MC0E~LE=</SignatureValue>
<KeyInfo>
    <X509Data>
        <X509Certificate>MI...lVN</X509Certificate>
    </X509Data>
</KeyInfo>

<Manifest Id="References">
    <Reference URI="config.xml">
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
        <DigestValue>j6...8nk=</DigestValue>
    </Reference>
    <Reference URI="index.html">
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>lm...34=</DigestValue>
    </Reference>
```

```
    <Reference URI="pictures/picture1.gif">
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>pq...56=</DigestValue>
    </Reference>
    <Reference URI="http://www.abccompany.com/logo.gif">
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>rs...78=</DigestValue>
    </Reference>
</Manifest>

</Signature>
```

9. Store the XML signature in a file

Store the XML signature in the ZIP root directory as “signature.xml”.

Additional Signatures

Additional signatures (by different signers) are placed in files “signature1.xml”, “signature2.xml” and so on. They have indentical structure except that they do not contain the <Manifest> element but refer to the <Manifest> in “signature.xml”.

```
<?xml version="1.0" encoding="UTF-8"?>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

<SignedInfo>
    <CanonicalizationMethod
      Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <SignatureMethod
      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
    <Reference URI="signature.xml#References">
      Type="http://www.w3.org/2000/09/xmldsig#Manifest">
        <DigestMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>725x3fVasdfvBGFGjhjyDSFvUk=</DigestValue>
    </Reference>
</SignedInfo>

<SignatureValue>MC0E~LE=</SignatureValue>

<KeyInfo>
    <X509Data>
        <X509Certificate>MI...lVN</X509Certificate>
    </X509Data>
</KeyInfo>

</Signature>
```

Verifying an XML Signature

1. Verify the signature of the <SignedInfo> element. To do so, recalculate the digest of the <SignedInfo> element (using the digest algorithm specified in the <SignatureMethod> element) and use the public verification key to verify that the value of the <SignatureValue> element is correct for the digest of the <SignedInfo> element.

2. Calculate the digest of the <Manifest> element and check that it is the same as <DigestValue> in <SignedInfo>.

3. Recalculate the digests of the references contained within the <Manifest> element and compare them to the digest values expressed in each <Reference> element's corresponding <DigestValue> element. Check that all files in the ZIP archive are listed in <Manifest>.

Note. The signature file needs to be in the beginning of the ZIP package. This makes it possible for a user agent to verify the signature while receiving the ZIP on-the-fly which involves calculating the digests (as in Step 3 above). Of course it is possible to first store the ZIP locally and after that verify the signature.

References

[Widget Reqs] Widgets 1.0 Requirements http://dev.w3.org/cvsweb/~checkout~/2006/waf/widgets-reqs/Overview.html

[Widgets 1.0] Widgets 1.0 http://dev.w3.org/cvsweb/~checkout~/2006/waf/widgets/Overview.html?rev=1.6&content-type=text/html;%20charset=iso-8859-1

Normative Text

Widgets may be signed. This is done by placing a signature file as the first file in the ZIP archive. The name of the file is *signature.xml*. It contains an XML Signature [XML Dsig] over the files in the ZIP archive (regarded as resources for the purpose of signing).
In order to support additional signatures (by the same signer or other signers) in an efficient and flexible way, the Manifest element is used to list the references. Additional signatures are placed in files *signature1.xml*, *signature2.xml* and so on. These must be the first files in the ZIP archive.
The Manifest element is only in *signature.xml*. Additional signature files refer to the Manifest in *signature.xml*.

Signing is required for all files in the ZIP archive; each signature.xml file must include a Digest for each files in the archive:

Signing procedure for the first signature is as follows

1. Calculate digests of *all* files in the ZIP archive. For each file, create a <Reference> element and place the digest in <DigestValue> child element according to [XML Dsig].
2. Collect the <Reference> elements within a <Manifest> element. Calculate the digest of <Manifest>.
3. Create the <SignedInfo> element with a reference to the <Manifest> element, including its digest.
4. Calculate the digest of the <SignedInfo> element, sign it and place the signature in the <SignatureValue> element.
5. Place the signer certificate in the <KeyInfo> element.
6. Place the <SignedInfo>, <SignatureValue>, and <KeyInfo> elements into a <Signature> element.
7. Store the <Signature> element in the ZIP root directory as “signature.xml”.

Signing procedure for 2nd etc. signature is as follows

1. Create the <SignedInfo> element with a reference to the <Manifest> element (in “signature.xml”).
2. Calculate the digest of the <SignedInfo> element, sign it and place the signature in the <SignatureValue> element.
3. Place the signer certificate in the <KeyInfo> element.
4. Place the <SignedInfo>, <SignatureValue>, and <KeyInfo> elements into a <Signature> element.
5. Store the <Signature> element in the ZIP root directory as “signature1.xml” (“signature2.xml” etc.)

Verification procedure is as follows

1. Read the contents of signature.xml. Verify the certificate stored in <KeyInfo> (details to do this are out of scope of this recommendation). If you are not able to do this proceed to the next signature.
2. Verify the signature of the <SignedInfo> element.
3. Calculate the digest of the <Manifest> element and check that it is the same as the corresponding <DigestValue> in <SignedInfo>.

4. Recalculate the digests of the references (contents of the files in the ZIP archive) contained within the <Manifest> element and compare them to the digest values expressed in each <Reference> element's corresponding <DigestValue> element. Check that all files in the ZIP archive are listed in <Manifest>.

Example (signature.xml):

```
<?xml version="1.0" encoding="UTF-8"?>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

<SignedInfo>
    <CanonicalizationMethod
      Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <SignatureMethod
      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
    <Reference URI="#References">
      Type="http://www.w3.org/2000/09/xmldsig#Manifest">
        <DigestMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>725x3fVasdfvBGFGjhjyDSFvUk=</DigestValue>
    </Reference>
</SignedInfo>

<SignatureValue>MC0E~LE=</SignatureValue>
<KeyInfo>
    <X509Data>
        <X509Certificate>MI...lVN</X509Certificate>
    </X509Data>
</KeyInfo>

<Manifest Id="References">
    <Reference URI="config.xml">
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
        <DigestValue>j6...8nk=</DigestValue>
    </Reference>
    <Reference URI="index.html">
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>lm...34=</DigestValue>
    </Reference>
    <Reference URI="pictures/picture1.gif">
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>pq...56=</DigestValue>
    </Reference>
</Manifest>

</Signature>
```

Example (signature1.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">

<SignedInfo>
    <CanonicalizationMethod
      Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <SignatureMethod
      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
    <Reference URI="signature.xml#References">
      Type="http://www.w3.org/2000/09/xmldsig#Manifest">
        <DigestMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>725x3fVasdfvBGFGjhjyDSFvUk=</DigestValue>
    </Reference>
</SignedInfo>

<SignatureValue>MC0E~LE=</SignatureValue>
```

```
<KeyInfo>
    <X509Data>
        <X509Certificate>MI...lVN</X509Certificate>
    </X509Data>
</KeyInfo>

</Signature>
```

Normative reference

[XML Dsig] XML-Signature Syntax and Processing. http://www.w3.org/TR/xmldsig-core/