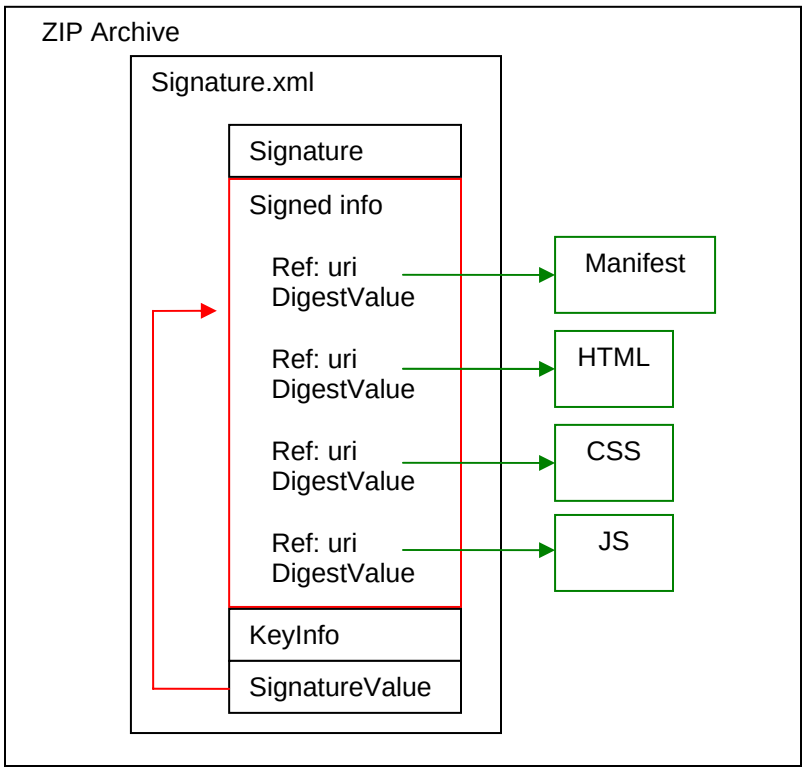
Current proposal (see http://dev.w3.org/2006/waf/widgets-digsig/)



Process

1. Download ZIP Archive
2. Unzip Archive
3. Locate Signature.xml
4. Check that required certificate chain is available to verify public key of signer by checking information in KeyInfo
5. For each Reference Element in the SignedInfo Element of the Signature.xml:
    a. Get the referenced data object
    b. Calculate the digest of the data object
    c. Compare the computed digest to the DigestValue
6. Verify SignatureValue calculated over SignedInfo Element
7. Install the widget

Steps 4- 6 defined by XML-Signature Syntax and Processing

Question: Will only SignatureInfo Element be canonocalised or will it be necessary to canonocalise the referenced files before the digets algorithm is applied?

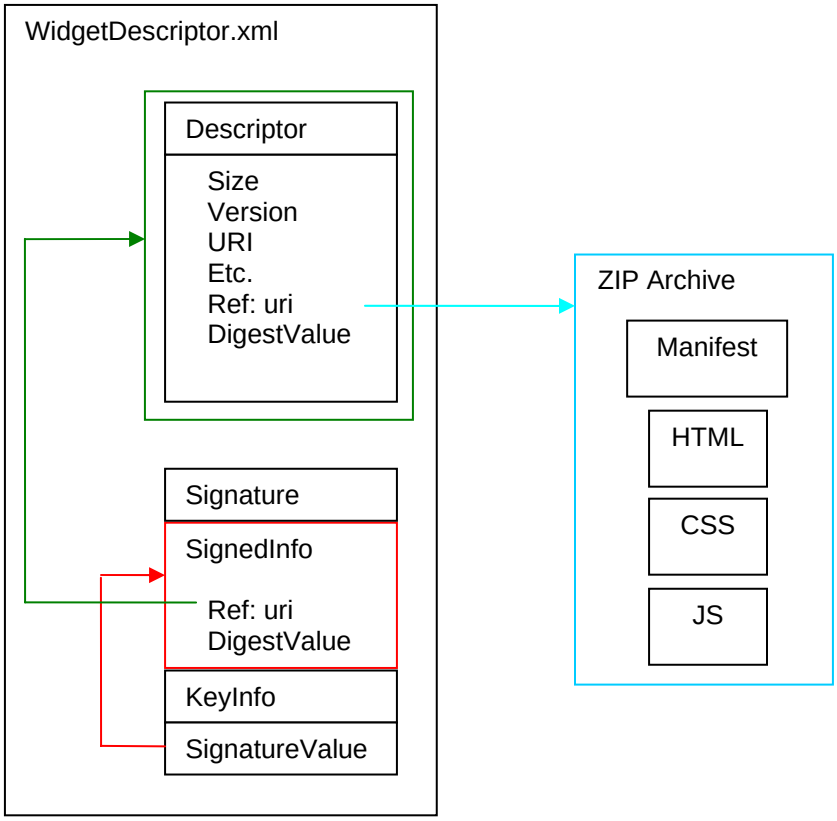
+ Pros

- Entire widget can be downloaded in a single package

- Cons

- Need to compute and verify hash for every file in ZIP. Likely to be many files, especially if graphic files are not excluded
- If not every file in the ZIP archive has a corresponding Reference Element in the SignedInfo Element (e.g. if picture files are excluded to cut down on processing) then the entire package is not protected – could lead to “widget graffitti
- Need to extract all files before checking signature and digest values
- Requires a method to exclude Signature.xml from digest validation
- It seems likely that combined ZIP and signing tool might be harder to implement / use when compared to using an existing tool for zipping and a new tool for signing.

New proposal – separate descriptor



Process

1. Download WidgetDescriptor
2. Check that required certificate chain is available to verify public key of signer by checking information in KeyInfo
3. For each Reference Element in the SignedInfo Element of the Signature.xml:
    a. Get the referenced data object
    b. Calculate the digest of the data object
    c. Compare the computed digest to the DigestValue
4. Verify SignatureValue calculated over SignedInfo Element
5. Get data object (e.g. ZIP archive) identified by URI in Descriptor Element
6. Compute and verify DigestValue of referenced data object
7. Unzip and install Widget

Steps 2 - 4 defined by XML-Signature Syntax and Processing

Notes:

There will only be one Reference Element in SignedInfo, which will reference the sibling Descriptor Element.

+ Pros

- Only ever two digest values to compute and verify (digest of ZIP, digest of Description element)
- Descriptor can be downloaded, and signature verified before ZIP is downloaded. If any step fails ZIP is not downloaded – better user experience. For example Descriptor could be used as the environment element in DLOTA
- Forwards compatible – additional fields could be added to the descriptor. While it is true that this information is also likely to be added to the manifest file, the ability to check this information before downloading the widget might be useful (e.g. API support, size (unpackaged) etc.)
- Backwards compatible – easy to take existing widgets and sign them without having to re-package them
- Injest on portal
- Closer to existing signing models, e.g Java MIDP2.0 – enterprise signing tools could be adapted more easily

- Cons

- Two step download must be handled by download client

Example schema for Widget Descriptor Element

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="widgetDescriptor">
 <xs:complexType>
  <xs:sequence>

   <xs:element name="widgetDescriptor">
    <xs:complexType>
     <xs:sequence>

      <xs:element name="id" type="xs:string"/>
      <xs:element name="author" type="xs:string"/>
      <xs:element name="version" type="xs:positiveInteger"/>
```

```
      <xs:element name="requiredAPIs" minOccurs = "0">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="api" type="xs:string"/>
        </xs:sequence>
       <xs:attribute name="critical" type="xs:boolean"/>
       </xs:complexType>
      </xs:element>

      <xs:element name="reference" minOccurs = "1">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="digestMethod" type="xs:string"/>
         <xs:element name="digestValue" type="xs:string"/>
        </xs:sequence>
       <xs:attribute name="uri" type="xs:uri"/>
       </xs:complexType>
      </xs:element>

     </xs:sequence>
    </xs:complexType>
   </xs:element>

  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:attribute name="widgetDescriptor" type="xs:string" use="required"/>
<xs:attribute name="id" type="xs:ID" use="optional"/>

</xs:schema>
```

Question: Could widget manifest schema be used instead?

Example signed widget desdriptor

```
<?xml version="1.0" encoding="UTF-8"?>

<signedWidgetDescriptor>

<widgetDescriptor id="Weather Widget"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNameSpaceSchemaLocation="widgetDescriptor.xsd">

  <author> The Widget Factory </author>
```

```
  <version> 1.0 </version>
  <requiredAPIs>
  <api id="Location" critical="true"/>
  <api id="Messaging" critical="true"/>
  <api id="OutgoingCall" critical="false"/>
  </requiredAPIs>

  <reference uri="reference to widget ZIP">

  <digestMethod algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
  <digestValue> Digest of ZIP archive </digestValue>
  </reference>

</widgetDescriptor>


<Signature id="WidgetSignature" 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#dsa-sha1" />
  <Reference URI="Reference to WidgetDescriptor Element">
  <Transforms>
  <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"
/>
  </Transforms>
  <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
  <DigestValue>Digest of WidgetDescriptor Element</DigestValue>
  </Reference>
  </SignedInfo>

  <SignatureValue>Signature over SignedInfo Element</SignatureValue>

  <KeyInfo>
  <KeyValue>
  <DSAKeyValue>
  <P>...</P>
  <Q>...</Q>
  <G>...</G>
  <Y>...</Y>
  </DSAKeyValue>
  </KeyValue>
  </KeyInfo>

</Signature>
```

</signedWidgetDescriptor>