Misconceptions about W3C Widgets

Prepared for the W3C Workshop on *The Future of Off-line Web Applications*, 5 November 2011, Redwood City, CA, USA.

By: Marcos Cáceres

Throughout the last few years, a number of misconceptions have been perpetuated about W3C Widgets. This document addresses some of those misconceptions (hopefully setting the record straight). Unfortunately, the misconceptions continue to be perpetuated at the detriment of the W3C Widget standard, while the market continues to fragment as new widget formats continue to emerge (e.g., Google's Installable Web apps, Windows 8 Metro apps) and old formats linger (e.g., Dashboard Widgets in Mac Os Lion).

As the main editor of the W3C Widget Family of Specifications, it is my position that: although the format has some issues and limitations, the format is well suited for the continued evolution of client-side applications that are written with Web standards. With backing from significant industry players, the format could yield industry wide benefits.

Misconception 1: Widgets reinvent the wheel

One of the most common misconceptions about the W3C's packaging format is that it reinvents (or could simply have used) existing packaging formats: namely, Oracle's Jar format, Mozilla's XPInstall (XPI) format, or the Open Document Format.

The widget packaging specification tried to overcome obvious limitations and complexity in the aforementioned formats. For instance, the above formats all require special authoring tools (or make use of complex internal file/folder structures). I say explicitly because the Working Group looked at all the aforementioned formats when we were specifying the W3C widgets format. After a long hard-look at each, we concluded that none were fit for purpose.

To show why each one would not make a suitable format:

- **JAR:** depends on the META-INF directory and defines files designed to load Java-based applications and libraries. Hence, it is specifically designed around the needs of Java and the semantics of various aspects of the packaging format that don't make sense for Widgets.
- **ODF:** amongst other things, it requires that a special file ('mimetype') be found at byte position 38, making it extremely difficult to create a package without a special tool.
- **Mozilla's XPI:** format (which itself reinvents JAR) makes use of RDF, which is notoriously difficult for developers to learn, read, write, and

maintain. Hence, the working group concluded that XPI would make a lousy widget-packaging format. Furthermore, XPIs suffer from versioning issues, which causes them to stop working when Mozilla Firefox is updated.

By contrast, and by design, W3C Widgets require no special tools to make a Widget package. The format is designed to take advantage of Zip tools that have come bundled with all operating systems for at least a decade. The format puts very few requirements on authors: the only two being to include a config.xml file (which can be almost empty) and a file to start the application (index.html)[1]. Furthermore, we designed W3C widgets so they were not bound to particular runtimes: in effect, overcoming the issues seen in XPIs and other platform-version-dependent applications.

Misconception 2: JSON is easier than XML

Another misconception revolves around the choice to use XML over the now *in vogue* JSON format. Proponents of JSON argue that XML is too complicated for developers and for implementers, while other more conspiratorial individuals claim that there is a "W3C strategy tax" at work. Proponents of this misconception ignore the fact that JSON can also easily be abused to make overly complicated data structures, and has a lot of restrictions that mirror XML's complexity in many ways (e.g., JSON also needs to be "well-formed", some characters are reserved, only certain data types can be used, etc.).

Having said that, I personally can agree that XML, when used badly (e.g., RDF, XML Schema, XACML) can overwhelm authors to the point of wanting to rip one's hair out. But, the use of XML in Widgets has never been proven to be an issue for Widget developers. This is because the Working Group was well aware of authoring issues with XML authoring and so the configuration document was designed to be as simple as possible to write: the document format is mostly flat in structure, taking away most authoring complexity found in XML. In other words, we knew "the tools won't save us": so we designed everything assuming it would be written by hand.

Further, to make Widget's use of XML as developer-friendly as possible, the Widget packaging specification provides:

- A meticulously defined XML processing model, which handles error conditions gracefully.
- Sensible defaults and fallback behavior in case of errors.
- A limited number of cases where processing needs to halt and a widget is

[1] Instructions of how to construct a widget are so simple, in fact, that Opera Software once published all the instructions on a business card.

rejected.
- That only one element (the widget element) be declared in the configuration document, so to allow the file to be identified as a W3C Widget and treated correctly on runtimes that support more than one widget format (e.g, in Opera's widget runtime).

The processing model, which draws heavily from the HTML Standard[2]'s processing model, is also designed to be future proof: allowing the format to be easily expanded as needed through either proprietary extensions or through new W3C specifications.

The choice to use XML for W3C Widgets was for legacy reasons: all the Widget runtimes from when we first started the widget standardization effort were using XML (see the [Landscape Document]).

The logic of the Working Group at the time was that if we used XML, then it would be easier for implementers to adapt their runtime and developer tools to use the XML W3C Widget format, and it would make it easier for existing content to be migrated to the XML W3C Widget format. Furthermore, when we started this effort in 2006, JSON was just appearing on the scene so it was not as popular a format as it is today (5 years on at the time of writing).

So, although we could have chosen to go against the grain in 2006 and used JSON, the parsing and error-handling models needed for either format would have been essentially identical: i.e., there is still little reason to choose one over the other, and choice of XML has helped other groups use the format for their own purposes (e.g., WAC defines a few additions to the format through a custom namespace). If the industry moves towards Google's and Mozilla's use of a JSON format for packaged client-side Web applications, a new parsing model specifically for JSON will need to be standardized. This is not a big deal; it just needs to be written down by someone (and no one has found time to do that yet).

Regarding the mythical "W3C strategy tax", certain individuals seem convinced that the W3C's membership has some evil conspiracy to push XML on the World and seem convinced that W3C widgets are part of that conspiracy. As much as I would love to be embroiled in some evil W3C XML conspiracy (and get a few kick-backs from this "strategy tax"), there is no such conspiracy to my knowledge (and last I checked, I'm still broke).

Misconception 3: XML Digital Signatures is too hard to implement

Another misconception that has emerged is one that XML Digital Signatures and

[2] http://www.whatwg.org/specs/web-apps/current-work/multipage/

XML Canonicalization are notoriously difficult to implement correctly. This misconception seems to stem from issues that emerged in Java prior to version 1.4.1, which, like all software, had some bugs[3]. Fortunately, issues with XML canonicalization were resolved.

In the case of widgets, there are now interoperable implementations[4] of the *Widgets XML Digital Signature* specification. Although there is limited W3C Widget-based content out in the wild that make use of digital signatures, none of the implementers have reported interoperability issues thus far. So, this misconceived claim remains unfounded and those that expound it have been unable and unwilling to provide any tests that prove there is an issue.

It should be said that cryptography is hard (independent of XML Digital Signatures). Very few individuals on this planet actually understand the mathematics and other things that make such systems secure. It is my belief that no matter what signature system would have been chosen for widgets, it would have still been hard for most developers to understand. I hold that our choice of XML Digital Signatures is OK for the common cases for which widgets are used.

Misconception 4: Widgets lack an origin

As well as being able to use a classic "http://domain.com" style origin, widgets can also use a custom scheme called the Widget URI scheme that typically looks like this:

```
widget://c1[…UUID…]a66/index.html
```

The widget URI scheme works like a "fake" HTTP server: sending back local files from inside a widget package (e.g., an image) by simulating a HTTP responses.

The misconception about origin arises because people don't really understand what an origin is: they think it means http://some.url.com (i.e., a website). Thankfully, the HTML specification makes it clear by defining an origin as:

> *"opaque identifiers or tuples consisting of a scheme component, a host component, a port component, and optionally extra data."*

---

[3] https://issues.apache.org/jira/browse/SANTUARIO-266
https://issues.apache.org/jira/browse/SANTUARIO-236
https://issues.apache.org/jira/browse/SANTUARIO-200
https://issues.apache.org/jira/browse/SANTUARIO-191
https://issues.apache.org/jira/browse/SANTUARIO-140
https://issues.apache.org/jira/browse/SANTUARIO-108

[4] http://dev.w3.org/2006/waf/widgets-access/imp-report/

In other words, widget:// makes for a perfectly valid origin.

Misconception: Widgets lack a security model

Another ungrounded misconception is that widgets lack a security model. Widgets, like most things “Web”, rely on the same-origin security model defined in HTML5. However, because the HTML same-origin policy is quite liberal and prone to cross-site scripting attacks, the Working Group also defined an **optional** security model for widgets called The Widget Access Request Policy (WARP).

WARP allows developers to declare up front which websites they are going to be communicating with, so the engine can block communication with websites that are not listed by the author. By default, Widget engines that implement WARP will not allow widgets to access content on the Web (i.e., they have no ability to access anything on the network). So, if a widget tries to access an image on the Web, the widget runtime won't allow it:

```
<!doctype html>
<!-- this will fail by default -->
<img src="http://w3.org/logo.gif">
```

To overcome this default policy restriction, an author needs to declare what domains on the Web a widget will access. For example, if a widget needs to access an image on "w3.org", the author makes a WARP declaration in the configuration document of a widget:

```
<widget xmlns="http://www.w3.org/ns/widgets">
  <!-- Gimme access to W3C resources -->
  <access origin="http://w3.org"/>
</widget>
```

Having declared an access request, the widget engine will now allow the image to be loaded from the domain w3.org:

```
<!doctype html>
<!-- the user agent grants access to w3.org! -->
<img src="http://w3.org/logo.gif">
```

If at runtime, the widget tries to access foo.com, the widget engine blocks that access because it violates the access request policy.

```
 <!doctype html>
<!-- this will fail: foo.com not in declared in
      config.xml
-->
<img src="http://foo.com/hello.gif">
```

The WARP declaration can also be used to do other useful things. For instance, when the user installs a widget, the widget engine can tell the end-user which domains the widget may try to contact.

It should be said that this misconception around if widgets have a security model or not is quite humorous. It's funny because, for anyone that has followed the widget work for a while knows, widget security is exactly what Apple's patent exclusion for widgets is all about[5]. Apple holds an extensive, and somewhat questionable, patent portfolio around Widgets and what they term "Widget Security".

[5] http://www.w3.org/2010/12/cfpa
http://www.w3.org/2009/11/widgets-pag-charter