

Submitted to Open Ajax Alliance by TIBCO Software Inc. and Dojo for discussion and consideration

June 19, 2006, Luke Birdeau, Alex Russell, Kevin Hakman

Common Open Ajax Event Hub Extensions

This proposal builds upon the Open Ajax Event Hub (OAH) recommending some standard methods for gracefully managing multiple toolkits in context of each other.

Specifically it looks at recommended standard toolkit methods and events dispatched across the OAH for managing the loading and unloading of any compliant toolkit.

In addition, a recommendation for an AJAX “markup container” capable of interpreting a myriad of potential XML markup constructs is outlined and included with a sample implementation.

Loading and Unloading

Task	Description	Requirement
Assert	Asserts the existence of a named library/toolkit in the browser. Assumes a Hash of standard toolkit names/identifiers.	MUST. Would require a list of reserved names, ids, etc per a given toolkit.
Unload	Remove a given toolkit/library from the JavaScript memory space. Cleans up object references for the given toolkit, etc.	SHOULD. Toolkits should gracefully clean up if they are no longer needed.
Load	Load a given toolkit at runtime. This might be called via Load Embed, which would actually load a given toolkit germane to a given fragment of markup	MUST. Caveat: once a toolkit is loaded, the existence of a toolkit should be broadcast by the toolkit to the Open Ajax Hub by its onload event.
Load Embed	Runtime loads a given fragment of XML markup associated with a toolkit. Assuming a single toolkit is able to generate multiple widgets/components, the given toolkit could be told to load different fragments of markup at multiple points on the screen. It is very possible that this method would first check to see if a given toolkit were loaded before proceeding. If it didn't exist, it would load	MUST. This is dependent upon approval of a common markup or serialization format. I don't care if this markup ultimately describes a common API, only that it exists and can be handed off to a given toolkit for processing.

	<p>the script defined by a src Attribute, assuming that the item would register itself with the OAH when it loaded (as part of its regular onLoad event). At any rate, if the given toolkit doesn't provide such a mechanism, one could imagine having an additional attribute on the oa-embed tag (oa-assert) that basically would result in a boolean if it ever evaluated to true. At such a point, the given toolkit would be auto-registered by the OAH under the id defined by the oa-id attribute. The value of the handler-for attribute would then be evaluated via a JavaScript eval(). If such an eval returned a function instance, the function would be directly called, passing an object reference to the oa-embed object. At any rate the result of evaluating the handler-for attribute would be the processing of the embed by the appropriate toolkit.</p>	
--	---	--

Event	Description	Open Standard
onLoadToolkit	Called when a given toolkit/library has fully loaded into the browser and is fully initialized.	MUST. Any compliant toolkit will register itself with the hub when it loads. Any registered listeners would then be alerted to this fact. It is possible that this can be supported by toolkits already out in market by allowing for an oa-assert attribute that would allow the assertion engine to auto-track when a given toolkit had loaded. At such a time the oa-id attribute would be used as the toolkit identifier, designating the given ID as the named toolkit identifier.
onUnloadToolkit	Called when a given toolkit/library has fully removed itself from the browser, ensuring proper closure	MUST.. Any compliant toolkit is required to unregister itself when it is no longer part of the browser DOM

A Markup Container for Mixing Markups within a Document

A key aspect of the above methods and events is that they would be routed via the Open Ajax Hub (OAH). It would be especially useful to GI if the OAH supported methods such as **assert**, **load**, **unload**, and **loadEmbed**. (**loadEmbed** would accept a standard-format descriptor such that the OAH could read the following attributes when **loadEmbed** is called (the attributes being set on the given GUI element (in this case what is shown as the **oa-embed** tag in the markup example below):

- **GUI Element.** Native HTML element into which the given toolkit should paint (basically the **oa-embed** tag as shown below). This element would contain whatever native html content was needed by a given toolkit as well as the following named attributes to identify how the given toolkit should be called:
 - **Toolkit ID (oa-id).** For example: gi, dojo, ymap, zimbra. This is perhaps a known name, accepted as the known identifier for the given toolkit. It could also simply be an arbitrary ID that is used to register a toolkit for the given browser session, but not necessarily an Ajax GUIID.
 - **Library Source (oa-src).** URL for the source JavaScript file that if embedded as the *src* attribute on an HTML *SCRIPT* tag would load the given toolkit. This would be useful for those cases in which the **oa-id** was unknown to the OAH, meaning the toolkit could be requested via a DOM2 append following the **oa-embed** tag (the script tag shown below depicts where the SCRIPT element would be appended to the Browser DOM)
 - **Initializer (oa-handler-for).** Named JavaScript function to call such that an eval (i.e., safe eval) of the value would resolve to a valid *function* which would then be passed an object reference to the oa-embed object. Note that this code will never be executed by the hub if the **oa-id** is not registered with the OAH as a valid Ajax toolkit. Instead, the **oa-src** attribute will be used to load the given toolkit.
 - **Assertion (oa-assert).** Named object whose existence could be evaluated (safe eval) to assert its existence if the given toolkit did not self-register with the OAH when it loaded. An example of this would be to pass *YMap* as the value of the **oa-assert**. Even though Yahoo Maps does not support the notion of the OAH, it could be made to play nicely by simply telling the OAH to auto-register the YMap toolkit when the given assertion existed. The OAH would simply manage an interval of its own when asked to perform an auto-registering assertion.

For example, consider the following markup in a web page. Assuming compliant toolkits supported such a markup container, the OAH could be designed such that it would serve as a common infrastructure element, capable of generically loading the declarative markup from any compliant toolkit. Consider the following:

```
<oa-embed oa-handler-for="dojo.widget.getParser().createComponents(frag) "  
  oa-src="http://url/to/xdomain/dojo/installation/src"  
  oa-id="dojo"  
  oa-assert="dojo.widget.FishEye"  
  id="uniqueHtmlDomId">
```

```
<div class="outerbar">
  <div class="dojo-FisheyeList" dojo:itemWidth="50" >
    <div class="dojo-FisheyeListItem"></div>
    <div class="dojo-FisheyeListItem"></div>
    <div class="dojo-FisheyeListItem"></div>
  </div>
</div>

</oa-embed>

<script language="JavaScript"
  src=" http://url/to/xdomain/dojo/installation/src"/>
```

As part of the *onload* event for the **document body**, one could call:

```
openAjaxHub.loadEmbed(document.getElementById("uniqueHtmlDomId"));
```

A key part of this strategy is that the contents of the `oa-embed` tag be left to the domain of the respective toolkits, thereby ensuring this as an enabling rather than constraining mechanism. On some level, one could imagine the OAHub being an extremely lightweight bootstrap/loader that served as the initializing element in an open framework. Via this object all other toolkits could be easily loaded in a 'play nicely' manner, consistent across all toolkits.