OmniUpdate®
Empower Web Excellence

How to Build Gadgets

OmniUpdate User Training Conference 2014

OmniUpdate, Inc.
1320 Flynn Road, Suite 100
Camarillo, CA 93012

OmniUpdate, Inc.
1320 Flynn Road, Suite 100
Camarillo, CA 93012
800.362.2605
805.484.9428 (fax)
www.omniupdate.com

Document Number: TC14-133.1
Publish Date: 3/9/2014

About OmniUpdate, Inc.

OmniUpdate® is the leading web content management system (CMS) provider for higher education. The company focuses on providing an exceptional customer experience to its 40,000+ OU Campus™ CMS users who manage more than 700 college and university websites in the U.S. and around the world. OU Campus is secure and scalable, and empowers institutions to effectively manage their web and mobile presence by taking advantage of the latest technologies through its complete feature set, extensible tools, deployment flexibility, and community resources. For more information, visit www.omniupdate.com.

About This Guide

The *How to Build Gadgets* provides fundamentals regarding creating and installing a gadget, as well as sample code for gadgets.

OU Campus Support

The Support site is available to everyone and users are encouraged to visit and browse the site for information. An institution's administrators are also available if the answer cannot be found on the Support site or further explanation and clarification is needed. Administrators may contact the OmniUpdate Support Team. Ways to access the OU Campus support documentation include:

- Support site: http://support.omniupdate.com/
- The help link in the main interface of OU Campus
- The WYSIWYG Help link
- Help links embedded in the system
- Text instructions are provide onscreen for specific fields and functionality
- OmniUpdate Community Network (OCN): http://ocn.omniupdate.com/

Conventions

Shorthand for navigation through the OU Campus CMS is indicated with a greater-than sign and bolded: **>** For example, **Setup > Sites**. Code snippets use `Courier New` and a shaded background.

Contents

Introduction to Gadgets

Just What Is a Gadget, Anyway?

Gadgets are best thought of as tightly-focused, self-contained web applications that add functionality to OU Campus. Here are some gadget facts:

- A gadget is just a web page (or a web app).
- Like other web pages or web apps, gadgets are generally written in HTML, CSS, and JavaScript.
- Each gadget is loaded into its own iframe within the OU Campus interface; hence it is self-contained.
- Gadgets have full access to the OU Campus API, and they can even interact with the main OU Campus application in limited ways.

Some Potential Applications for Gadgets

- Content creation utilities
  - Color picker
  - Placeholder text generator
- Automate repetitive tasks
  - Assign a user to multiple groups at once
  - Schedule a publish on multiple files at once
- Third-party API clients
  - Upload an image from Flickr to OU Campus
  - Compose tweet and post to Twitter

Ingredients of Gadgets

Every gadget must have:

1. An http(s) URL that is accessible by the intended users. (Example: http://www.gallenauniversity.com/mygadget/.)
   a. The URL must end in a slash.
   b. The gadget must successfully load from that URL.
2. A *config.xml* document that is accessible over http(s) by appending "config.xml" to the end of the gadget URL. (Example: http://www.gallenauniversity.com/mygadget/config.xml.)

That's all!

Note that the gadget author or gadget administrator must have control over what gets served at the gadget's URL. Without that, various problems may be encountered, including an inability to place the *config.xml* there.

Where to Host the Gadget

Gadgets are hosted on web servers, like any other web page or web app. The gadget can live on any web server (web host) to which the gadget author or gadget administrator can write, and

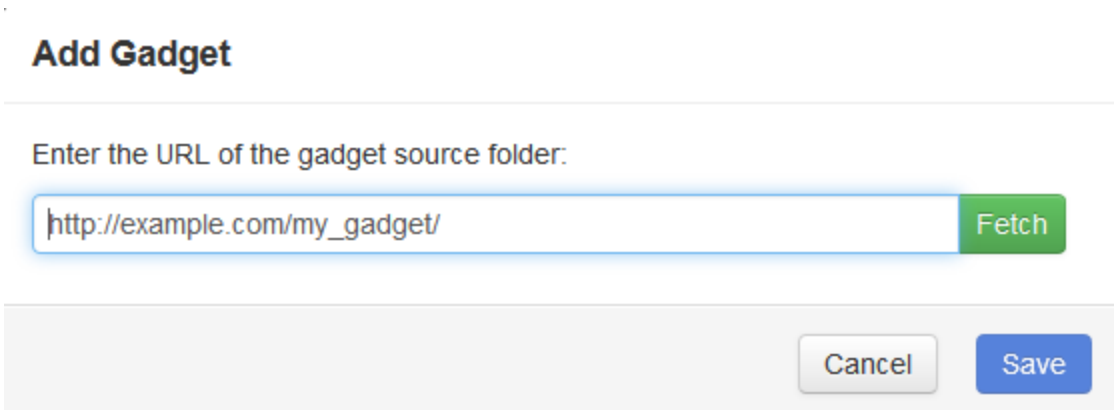
that is accessible by the intended users from the intended locations. If an intended user can load the gadget's URL in a browser tab and not get an http error, the URL is good.

Installing Gadgets in OU Campus

Gadgets are installed in OU Campus at the account level. Each gadget installed has an associated access group. If a group is not selected, the gadget will be accessible by administrators only.

Gadget installation is done in **Setup > Gadgets** (Level 10 administrators only).

1. Click the **New** button.
2. Enter the gadget URL.
3. Click **Fetch**.
4. Confirm everything looks okay in the configuration dialog.
5. **Save** the new gadget.



Users can enable or disable individual gadgets to which they have access. An administrator can also bulk-enable a gadget for all members of its access group, and/or turn on auto-enable for a gadget, so that any new members of its access group have the gadget enabled automatically.

The Gadget Configuration File (config.xml)

Every gadget must have a configuration file named *config.xml* that can be read at the URL constructed by appending "config.xml" to the gadget's URL. (This URL need not correspond to a physical file, as long as the web server responds with an XML document in the correct format.)

The purpose of *config.xml* is to tell OU Campus about the gadget. This includes things like

- Where the gadget can be displayed
- What it is called

The file can also specify custom configuration parameters for the gadget's own, internal use.

When a gadget is installed into an account in OU Campus, the system reads the *config.xml* and adds its data to the OU Campus database. Thereafter, the *config.xml* is not read again unless an administrator uses the **Refresh** function on the gadget management screen.

Here is a simple example *config.xml*:

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

<config>
      <entry key="types" private="true">sidebar</entry>
      <entry key="title">My Gadget</entry>
      <entry key="icon" private="true">icon16.png</entry>
      <entry key="description" private="true">My awesome gadget</entry>
      <entry key="thumbnail" private="true">thumbnail96.png</entry>
      <entry key="initial_height" private="true">200</entry>
</config>
```

Of the entries in this or any *config.xml*, only the first entry element, with `key="types"`, is required. The others in the example are mainly cosmetic, although including them is highly encouraged. The `private` attribute, which is used by most of the entries in the example, is explained later.

Standard and Custom Config Entries

Some config entries are known as *standard* entries; anything else is a *custom* entry. Standard entries are those that are recognized by, and have special meaning to, OU Campus. When OU Campus sees one of these entries in your *config.xml*, it changes how it displays or interacts with your gadget in some way. The example config above contains only standard entries.

By contrast, custom entries are ones that are not recognized by OU Campus; they are purely for your gadget's own use. Custom entries will be discussed more later.

There is a table that describes all the standard config entries in Appendix A.

Custom Config Entries

A custom config entry can serve at least two purposes. But first, it's important to understand a couple of facts about gadget config data:

- Since gadget config data is stored in the OU Campus database, it persists across sessions.
- Custom config data is (potentially) stored in the database on a per-user basis.

Given these facts, it is apparent that a custom config entry can serve, first, as a way for the gadget to store a variable value persistently, so that it is remembered across sessions. Second, since custom config data can be stored per-user, it can serve as a way to provide user settings or preferences to the gadget.

The per-user aspect of custom configs will be explained further in the discussion of the `overwritable` attribute, below.

Examples of Custom Config Entries

```
<entry key="font_size" label="Font Size"
overwritable="true">13px</entry>
```

This entry could be used to allow users to change the font size the gadget uses.

```
<entry key="last_selection" private="true"
overwritable="true">Finland</entry>
```

This entry could be used to remember the user's last selection in a menu or something.

```
<entry key="access_token" private="true" overwritable="true"></entry>
```

If the gadget communicates with a third-party API, such as Facebook's, the access token could be stored for each user by means of an empty entry like this.

Attributes

There are three optional attributes that custom config entries can use: `private`, `overwritable`, and `label`.

private

If an entry has `private="true"`, it will not be exposed anywhere in the OU Campus user interface, either on the management screen or in the user-level gadget configuration dialog box. Use `private` for configs that users and administrators should not be able to edit directly. No user or administrator can change the value of a private config entry, unless the gadget itself provides a user interface for changing it. If a config is not private, it is exposed in the gadget configuration dialog box as a text input field.

There is one exception to the above rule. The standard "title" config entry is always exposed in gadget configuration dialog boxes, even if its `private` attribute is set to true. However, if `private` is true, OU Campus will not allow administrators to edit the title.

overwritable

By default, only administrators have the ability to change configuration values for a gadget. They can do so in the gadget's configuration dialog box that appears when clicking the gadget's name on the gadget management screen (**Setup > Gadgets**). Config values that administrators

set are applied account-wide. The new value is used by all the instances of the gadget in users' OU Campus sessions throughout the account.

However, if a config entry has `overwritable="true"`, the config value is eligible to be modified and saved on a per-user basis. This means that every user in the OU Campus account where the gadget is installed can have their own value for that config entry.

Note: A value is not saved for any user until the user actually opens the gadget's configuration dialog box and saves the configuration. Until that happens, the user's instance of the gadget will continue to use the config value set by an administrator (or the default value, if no other value was set). But once the user does set a value, the new value will be used by that instance of the gadget, even if an administrator later changes the value for the account.

label

As mentioned before, if a config entry is not marked as private, it will be exposed as a text input field in the gadget's configuration dialog box. Unless a `label` attribute is added, the input field will have no label. Set the `label` attribute value to a concise description of what the config is about. For example:

```
<entry key="fullname" label="Your Name" overwritable="true"></entry>
```

Refreshing the Config

Once a gadget is installed into an OU Campus account, the OU Campus app does not read the *config.xml* file again, since its entries have now been transferred to the OU Campus database. If the gadget author, or someone else who has write access to the *config.xml* file, subsequently modifies the file, the changes will not be picked up by OU Campus and written into the database unless an account administrator uses the **Refresh** function. This function is one of the options available on each row of the gadget management screen.

When a gadget is refreshed, a few things happen:

- Any new config entries are added to the database.
- Any entries that have been removed in *config.xml* are removed from the database.
- For existing, unremoved entries:
    - If the state of either the `private` or the `overwritable` attribute has changed in *config.xml*, the change is reflected in the database and the UI. For example, if a formerly private entry becomes non-private, it will now be exposed in configuration dialogs.
    - If the entry's value changes in *config.xml*, the change is written to the database at the account level, *but only if the entry is private*. If the entry is not private, the database will keep the entry's old value.

Building a Very Simple Gadget

Start with two nearly blank files: a *config.xml* and an *index.html*. The latter will provide the entire contents of the gadget.

For the workshop, in the workshop site:

1. Navigate to the *workshop-gadgets* directory. Inside will be a folder named *blank*, which contains a *config.xml* and an *index.html*.
2. Make a copy of this folder inside the same parent folder and rename the copy “hello.”
3. Publish the *hello* folder so that it will be uploaded to the production server.
4. Open *hello*.
5. Check out *config.xml.*
6. Choose **Edit > Source**.

Notice that *config.xml* already contains an XML declaration and the root node, `<config>`. Add a few entries inside the root node.

```
<config>
      <entry key="types" private="true">dashboard</entry>
      <entry key="title">My First Gadget</entry>
      <entry key="icon"
private="true">http://www.omniupdate.com/favicon.ico</entry>
</config>
```

7. Save the file.
8. Open *index.html* for editing. Within OU Campus, check it out first, and edit using **Source**. Notice it already has a little HTML in it. This prevents having to start from scratch.
9. Start by deleting the words “Gadget content goes here” in the `<div id="main">` element.
10. Enter a little content by typing this inside the now-empty div element:

```
<strong>Hello!</strong> I am your new gadget.
```

11. Save and publish the file.

Testing the New Gadget

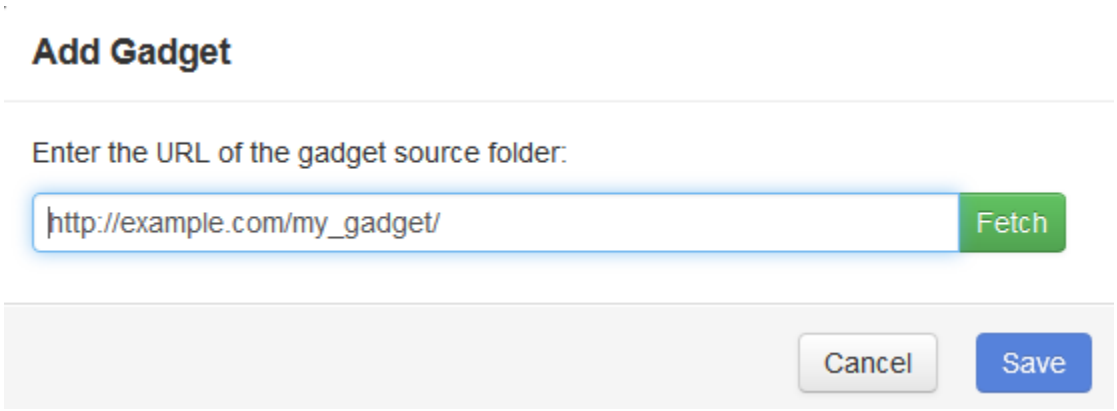
To test the gadget, first make sure the gadget loads successfully in a browser tab from its web address. If it fails this test, it won’t be usable in OU Campus either. For this example and during the workshop, the URL will be: http://workshop[#].outc14.com/workshop-gadgets/hello/. Remember to change [#] to the workshop number assigned.

Open a new browser tab and load the URL. If successful, the words “**Hello!** I am your new gadget.” will be displayed.

Installing the Gadget in an OU Campus Account

1. Log into OU Campus as a Level 10 administrator.
2. Navigate to **Setup > Gadgets**.

3. Click **New**.
4. Enter the gadget URL.
5. Click **Fetch**.
6. Confirm everything looks okay in the configuration dialog.
7. **Save** the new gadget.



The gadget should now be displayed in the list. Navigate to the **Dashboard** and verify that the gadget is displayed in the configuration dialog box.

Inspect the gadget using Chrome's Inspect Element command to see that it displays in an ordinary iframe.

Building a More Useful Gadget

Next a gadget that does something will be created to expand on the gadget process. This gadget will take advantage of some gadget-supporting features of OU Campus.

A color picker gadget will let users select a color visually and copy its hexadecimal representation. An open source color picker will be downloaded from the internet and used. The color picker will then be "wrapped" in a gadget.

1. Start by making a copy of the *hello* directory in OU Campus.
2. Name the copy *colorpicker*.
3. Open the *colorpicker* folder.
4. Edit *config.xml* so it looks like this. (Changes are in bold.) Save.

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

<config>
	<entry key="types" private="true">sidebar</entry>
	<entry key="title">Color Picker</entry>
	<entry key="icon" private="true">icon.png</entry>
	<entry key="initial_height" private="true">288</entry>
</config>
```

5. Save.
6. Edit *index.html*.
7. Change the contents of the `<title>` element to "Color Picker," and set the contents of the `<div id="main">` element to `<input type="text">`.
8. Save.
9. The color picker, called *Spectrum*, consists of a JS file (*spectrum.js*) and a CSS file (*spectrum.css*). These have been included in the */workshop-gadgets/lib* folder. In the `<head>` of *index.html*, add a link to *spectrum.css* like this:

```
<link rel="stylesheet" href="../lib/spectrum.css">
```

10. Again in the head section, add a script tag linking to *spectrum.js*. Since Spectrum requires jQuery, linking to jQuery is necessary. A copy of jQuery 2.1.0 is included in the *lib* folder as well.

```
<script src="../lib/jquery-2.1.0.min.js"></script>
<script src="../lib/spectrum.js"></script>
```

11. Add the following `<script>` element at the end of the body section. This code will create the color picker and append it below the `<input>` element, hiding the `<input>`.

```
<script>
	$('input').spectrum({
		flat: true,
		showInput: true,
		showButtons: false,
```

```
            preferredFormat: 'hex'
      });
</script>
```

12. Save.
13. Publish the entire *colorpicker* folder.
14. Add the gadget to the OU Campus account.

The gadget should now be available in the sidebar.

To improve the appearance of the color picker, add some style rules that will override parts of Spectrum's default stylesheet. In *index.html*, add the following stylesheet in the head section:

```
<style>
      #main {
            text-align: center;
      }
      .sp-container {
            margin-bottom: -15px;
            border: none;
            background: none;
      }
      .sp-picker-container {
            border-left: none;
            padding-left: 0;
            padding-right: 0;
      }
      .sp-picker-container {
            width: 220px;
      }
      .sp-input {
            background: white;
      }
</style>
```

After saving (and publishing) the file, reload the gadget's iframe to see the changes. (In Chrome, right-click inside the gadget and choose **Reload Frame**.)

Taking Advantage of gadgetlib.js

The color picker's functionality can be extended by *automatically* inserting the hex value into the source code by clicking a button in the gadget. This is accomplished by using a special function, `oucInsertAtCursor`, which is included in OmniUpdate's gadget library, *gadgetlib.js*. There is a copy of *gadgetlib.js* in the */workshop-gadgets/lib* folder.

To use gadgetlib in the gadget, add a `<script>` tag linking to it:

```
<script src="../lib/gadgetlib.js"></script>
```

Before using any function in gadgetlib, create an instance of the `Gadget` object, whose constructor is included in gadgetlib. Once there is a `Gadget` instance, call on the various methods of `Gadget` to get things done.

To create a `Gadget` instance, just add the following line at the top of the body script:

```
var gadget = new Gadget();
```

Using the oucInsertAtCursor Method

A gadget can insert arbitrary text or HTML into the source editor or the WYSIWYG Editor of OU Campus by calling `gadget.oucInsertAtCursor`.

1. Start by adding an **Insert** button by which the user can indicate that the color value should be inserted. In the main div, add a `<button>` element below the `<input>` tag, like this:

```
<div id="main">
      <input type="text">
      <button id="insert-btn">Insert</button>
</div>
```

2. Next, style the button by adding styling to the gadget's internal stylesheet:

```
      #insert-btn {
            width: 220px;
            font-size: 13px;
      }
```

3. Then add the code that calls `gadget.oucInsertAtCursor` when the user clicks the **Insert** button. Insert this at the bottom of the body script:

```
$('#insert-btn').on('click', function () {
      var color = $('input').spectrum('get').toHexString();
      gadget.oucInsertAtCursor(color).then(function (result) {
            if (result.error) {
                  alert(result.error);
            }
      });
});
```

`oucInsertAtCursor` takes one argument, which is the text to be inserted into the editor in OU Campus. If the text is HTML and the editor is WYSIWYG, the editor will create the appropriate DOM elements and insert them at the cursor. Otherwise, the text will be inserted as-is. In this case, this argument is set to the hex string of the selected color.

The method returns a jQuery Deferred object. This object can then be chained with a `then` function to the call. This function will run once OU Campus has finished executing the content insertion, whether successful or not. The `then` function will pass an argument providing the result of the insertion. The argument will look like one of the following: `{success: true}` or `{error: "An error occurred."}`.

Test the **Insert** button now. Save and publish *index.html*, reload the gadget's iframe, and then return to the source editor for *index.html*.

1. Add a `style` attribute to the `<body>` tag and set the background color for the gadget body in this attribute. Add a `style` attribute like this, leaving an empty space for the color value:

```
<body style="background-color: ;">
```

2. Place the cursor (insertion point) just before the semicolon.
3. Pick a color in the color picker and click **Insert**.

The selected color's hex value should be inserted into the source editor at the location of the cursor.

Reading From and Writing to the Gadget Config

The gadget can take advantage of configuration reading and writing. For instance, the last-used color value can be stored in such a way that it can be retrieved later. This can be done by adding a custom configuration parameter to the gadget.

Add a custom entry to the *config.xml* file:

```
<entry key="initial_color" label="Starting Color"
overwritable="true"></entry>
```

To transfer this new config entry to the OU Campus database, click the **Refresh** hover option for the gadget on the gadget management screen.

Since the config entry is not private and is overwritable, users will be able to edit the value of `initial_color` in the gadget's configuration dialog box by clicking the new "gear" icon in the gadget's title bar. Enter any valid CSS value, including both text and hex values.

Add some code to the gadget to read the `initial_color` value from the database when the gadget is loaded. The `Gadget` object has a method, `fetch`, that will fetch a gadget's configuration using the OU Campus API. This method also returns a jQuery Deferred object, so a `then` function can be chained to the fetch call and complete an action with the fetched data when the fetch is complete. In this case, the action will be to set the color of the picker to the fetched color value.

```
function setColor(color) {
      $('input').spectrum('set', color);
}

gadget.fetch().then(function () {
      var color = gadget.getConfig('initial_color');
      setColor(color);
});
```

After reloading the gadget's iframe, the gadget will read the user-specified color from the configuration and automatically set the picker to it.

To further improve the user experience, the gadget can change the picker color as soon as the user configures it; instead of having to wait for a reload. This is accomplished by adding an event listener for the `configuration` event to the gadget. This works because whenever a gadget's configuration changes, OU Campus sends the new configuration to the gadget using HTML5's `postMessage` method, and then the `Gadget` object converts the message into a `configuration` event, which can be listened for with `$(gadget).on('configuration')`. Here's the code:

```
$(gadget).on('configuration', function (evt, data) {
      setColor(data.initial_color);
});
```

The `data` argument will contain the entire configuration, as if the gadget had been fetched with `gadget.fetch`. The handler function then simply calls `setColor`.

When the gadget receives a configuration message from OU Campus, the `Gadget` object will set its `config` property to the new configuration, so that subsequent `gadget.getConfig` calls will return updated values.

Finally, add the code that will cause the gadget to save the selected color to the OU Campus database. In this case, the gadget will not be reading the stored configuration but instead writing to it. This will be done when the user clicks the **Insert** button. All that is needed is to add one line to the click handler on the button.

Here is the click handler again, with the added line in bold:

```
$('#insert-btn').on('click', function () {
      var color = $('input').spectrum('get').toHexString();
      gadget.oucInsertAtCursor(color).then(function (result) {
            if (result.error) {
                  alert(result.error);
            }
      });
      gadget.save('initial_color', color);
});
```

The code calls the `save` method of the `Gadget` object and passing it two arguments, which are the name of the config parameter to save, and the new value. The `Gadget` object will first update its internal configuration with the new value, and then it will use the OU Campus API to update the stored configuration in the database.

After saving and publishing the file and reloading the gadget iframe, by clicking **Insert** the gadget will save the selected color back to the database. This can be verified by reloading the gadget iframe again. The picker should be set to the saved color.

Appendices

Appendix A – Table of Standard Config Entries

Key	Purpose
types	**Required**. Determines where the gadget can appear in the OU Campus user interface. Currently supported types are “dashboard” and “sidebar”. Multiple types can be defined with a comma-separated list.
title	The name that appears in the gadget’s title bar, in the gadget chooser, and on the management page. The title of an installed gadget can be edited by an admin unless this key is set to private (see explanation of the `private` attribute below). Do not leave this blank.
icon	The icon that appears in the gadget’s title bar and on the management page. For best results, use a 16 x 16 pixel image. The URL can be absolute, root-relative, or relative to the gadget’s index page.
description	The description that appears in the gadget chooser.
thumbnail	The thumbnail image that appears in the gadget chooser. This is displayed at 96 x 96 pixels. The URL can be absolute, root-relative, or relative to the gadget’s index page.
columns	*Dashboard gadgets only*. Determines the width of the dashboard gadget in columns. Supported values are 1, 2, and 3.
initial_height	*Sidebar gadgets only*. Determines the default height of the sidebar gadget in pixels when it is expanded. OU Campus may override this value under various circumstances.
sidebar_context	*Sidebar gadgets only*. Specifies contexts in which the sidebar gadget will be visible. If the current context of OU Campus is not one of the specified ones, the gadget will be hidden. Currently supported contexts include: **Context** — **In Effect When** page — The “subject” of the current view is a single page. asset — The “subject” of the current view is a single asset. file — The “subject” of the current view is either a page or an asset. edit — The current view is either the WYSIWYG editor or the source code editor. If you do not specify any sidebar contexts, the gadget will appear in all contexts.
notifications	Specifies types of notification events that will be passed on to the gadget.

Appendix B – Table of Standard Messages Sent to Gadgets

Name	Description
configuration	Sent when a gadget's configuration changes. The content of the message is a plain object giving the (entire) configuration.
expanded	Sent when the gadget is expanded—i.e., its disclosure button is toggled to the "down" state, showing the gadget. Message has no content.
collapsed	Sent when the gadget is collapsed—i.e., its disclosure button is toggled to the "up" state, hiding the gadget. Message has no content.
view_changed	Sent when the OU Campus application's view has changed—i.e., the user has navigated to a different place in the application. The message content is a plain object with, at minimum, a `section` property that tells what area of the app the user is in. Currently, the possible values of `section` are "dashboard", "content", "reports", and "setup." If the current view is a directory listing, the message data will have a `directory` property giving the root-relative path of the directory. If the current view deals with an individual page, the message data will have a `page` property giving the root-relative path of the file. If the current view deals with an individual asset, the message data will have an `asset` property giving the ID of the asset.
notification	If the gadget's `config.xml` contains a "notifications" entry, when the OU Campus application receives a notification from the OU Campus server, if the notification type matches one of the types specified in the config entry, the notification content will be passed on to the gadget in the content of this message.