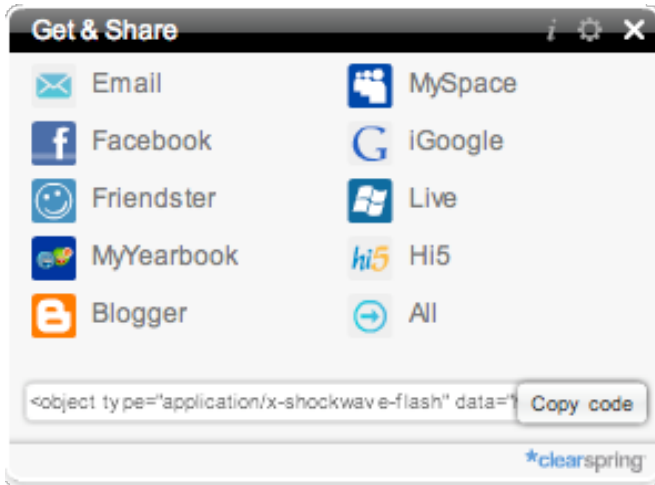


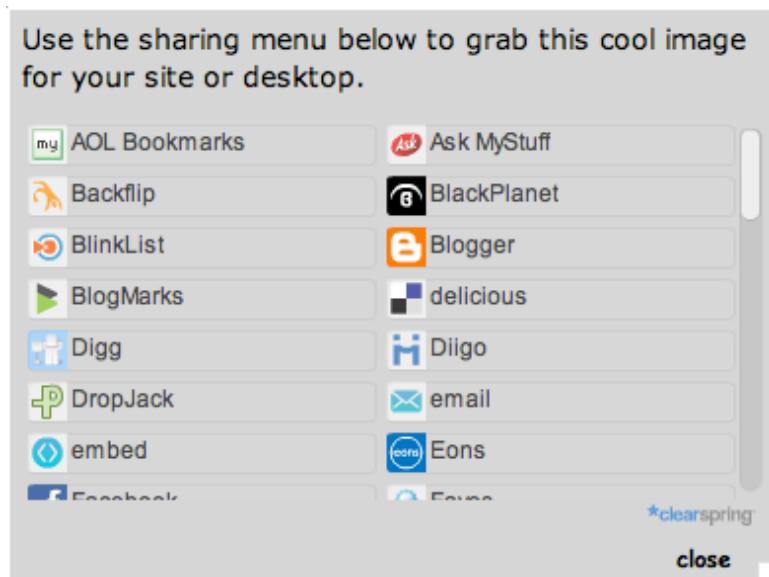
Launchpad – Chromeless Menu Mode

The Launchpad menu can be easily added to you widget using the Launchpad In-Widget API. When using the API the default behavior is to provide the full sharing menu, which is customizable, similar to the following:



In most implementation, the full (default) menu provides the user with the most capabilities. However if you would like to make the menu seem fully integrated into you widget, the chromeless menu mode may be your option.

The chromeless menu mode allows you to add the sharing menu to your widget without the “get & share” title bar and tabs (Popular and All). You simply get the option to display the sharing destinations of the Popular or All tab, similar to the following:



In the above example, the sharing menu’s “All” tab was integrated into a custom window.

Using Chromeless Menu Mode

To use the chromeless menu mode simply set the menu configurations prior to showing the menu using the In-Widget API. We assume that you are familiar with the In-Widget API and will not go into any details here. If you are not familiar with the In-Widget API, please check out the following documentation:

<http://www.clearspring.com/docs/tech/apis/in-widget> and our Guides:

<http://www.clearspring.com/docs/guides> to get up to speed.

To begin we need to create a new widget in the Widget Console. Select “Add Widget” and then “Use the In-Widget API” option. Once you’ve created the widget you will be presented with the In-Widget API code that you will integrate into your widget. The code will look like:

```
package {
    import flash.display.*;
    import flash.events.*;
    import flash.system.*;
    import flash.utils.*;
    import flash.net.*;

    public class MyApiTest extends MovieClip
    {
        private const kernelUrl:String = "http://widgets.clearspring.com/o/<WID>/-/-TRK/1/lib.v3.as3.swf";
        private var kernel:Object;
        private var kernelLoader:Loader;
        private var isOpen:Boolean = false;

        public function MyApiTest()
        {
            Security.allowDomain("bin.clearspring.com");
            Security.allowDomain("widgets.testspring.com");
            kernelLoader = new Loader();

            kernelLoader.addEventListener('on_ready', onApiLoad);
            addChild(kernelLoader);

            // Load Clearspring kernel
            kernelLoader.load(new URLRequest(kernelUrl),
                new LoaderContext(false, ApplicationDomain.currentDomain, SecurityDomain.currentDomain));
        }

        private function onApiLoad(e:Event):void
        {
            // Grab reference to actual kernel
            kernel = Object(kernelLoader.content).kernel;

            // Send a custom event
            // @see http://www.clearspring.com/docs/tech/apis/in-widget/track
            kernel.track.event('Kernel loaded');

            // @see http://www.clearspring.com/docs/tech/apis/in-widget/context
            trace('You have viewed this widget ' + kernel.context.user.WIDGET_VIEWS + ' times');
            trace('You seem to be in ' + kernel.context.user.geo.getCountry());

            // Retrieves a new embed tag for this widget
            // @see http://www.clearspring.com/docs/tech/apis/in-widget/share
            // Uncomment the following line to see how it works:
            // kernel.share.get('tag', {}, shareCallback);

            // Shows menu on mouse click
            // @see http://www.clearspring.com/docs/tech/apis/in-widget/menu
            kernel.menu.addEventListener(kernel.menu.event.OPEN, onMenuEvent);
            kernel.menu.addEventListener(kernel.menu.event.CLOSE, onMenuEvent);
            stage.addEventListener(MouseEvent.CLICK, function (e:MouseEvent):void {
                if (!isOpen && e.target == stage) {
                    kernel.menu.show();
                }
            });
        }
    }
};
```

```

// WARNING:
// Do not publicly deploy a widget containing your user ID, as it can
// be used to modify any of your widgets. Treat it like your password.

// Updates widget's content template permanently. Use wisely!
// @see http://www.clearspring.com/docs/tech/apis/in-widget/widget-1
// To use this call, modify the following line to contain
// 1. your own Clearspring user ID
// 2. your desired content
// and then uncomment it:
// kernel.widget.update(YOUR-USER-ID, '4950f78cd952c452', {content: '<h1>insert your own
content</h1>'});

}

/**
 * Callback for Clearspring share services; since sharing is asynchronous,
 * we need to specify a callback function to interpret the results.
 *
 * In this example, we're using share.get() below to return an embed tag.
 *
 * @see http://www.clearspring.com/docs/tech/apis/in-widget/share#share.get
 */
public function shareCallback(status:Number, result:Object):void
{
    if (status) {
        trace('An error occurred: ' + status);
    } else if (result.tag) {
        trace('Got embed tag: ' + escape(result.tag));
    } else {
        trace('No embed tag returned');
    }
}

/**
 * Callback for Clearspring service menu events. Traces open/closed state of menu each
 * time it changes.
 *
 * @see http://www.clearspring.com/docs/tech/apis/in-widget/menu#menu.addEventListener
 */
public function onMenuEvent(event:Event):void
{
    switch (event.type)
    {
        case kernel.menu.event.OPEN:
            isOpen = true;
            trace('Menu opened!');
            break;
        case kernel.menu.event.CLOSE:
            isOpen = false;
            trace('Menu closed. ');
            break;
        default:
            trace('Got a menu event: ' + event);
    }
}
}
}

```

This code will add the standard sharing menu to your widget. To make the menu chromeless you need to use the In-Widget API *setOptions* method. In the generated code we will add the following *setOptions* method in the *onApiLoad()* function before we call the *menu.show()* – lets add it before the comment: *Shows menu on mouse click*

```

kernel.menu.setOptions({
    global : {template : 'custom',skin : 1, height:220, width:385},
    style: {menu : {background : 0xCCCCCC,chrome : 0xCCCCCC}},

```

```
display: {single_tab : 'all',branding : false,menu_nav:{close:{show:false}}}
});
```

When you have added the *setOption* method to the *onApiLoad()* function your code should look like:

```
private function onApiLoad(e:Event):void
{
    // Grab reference to actual kernel
    kernel = Object(kernelLoader.content).kernel;

    // Send a custom event
    // @see http://www.clearspring.com/docs/tech/apis/in-widget/track
    kernel.track.event('Kernel loaded');

    // @see http://www.clearspring.com/docs/tech/apis/in-widget/context
    trace('You have viewed this widget ' + kernel.context.user.WIDGET_VIEWS + ' times');
    trace('You seem to be in ' + kernel.context.user.geo.getCountry());

    // Retrieves a new embed tag for this widget
    // @see http://www.clearspring.com/docs/tech/apis/in-widget/share
    // Uncomment the following line to see how it works:
    // kernel.share.get('tag', {}, shareCallback);

    kernel.menu.setOptions({
        global : {template : 'custom', skin : 1, height:220, width:385},
        style : {menu : {background : 0xCCCCCC,chrome : 0xCCCCCC}},
        display: {single_tab : 'all', menu_nav:{close:{show: false}}}
    });

    // Shows menu on mouse click
    // @see http://www.clearspring.com/docs/tech/apis/in-widget/menu
    kernel.menu.addEventListener(kernel.menu.event.OPEN, onMenuEvent);
    kernel.menu.addEventListener(kernel.menu.event.CLOSE, onMenuEvent);
    stage.addEventListener(MouseEvent.CLICK, function (e:MouseEvent):void {
        if (!isOpen && e.target == stage) {
            kernel.menu.show();
        }
    });

    // WARNING:
    // Do not publicly deploy a widget containing your user ID, as it can
    // be used to modify any of your widgets. Treat it like your password.

    // Updates widget's content template permanently. Use wisely!
    // @see http://www.clearspring.com/docs/tech/apis/in-widget/widget-1
    // To use this call, modify the following line to contain
    // 1. your own Clearspring user ID
    // 2. your desired content
    // and then uncomment it:
    // kernel.widget.update(YOUR-USER-ID, '<WID>', {content: '<h1>insert your own content</h1>'});
}
}
```

This code will now display the sharing menu in chromeless mode, give it a try.

***setOptions()* Values for Chromeless Menu Mode**

Now that you know how to enable the menu in chromeless mode, lets explain what the options mean. There are several options, defined as objects, set to provide you with the most control over the menu.

global object: defines the global setting for the sharing menu.

```
global : {template : 'custom', skin : 1, height:220, width:385}
```

template: the style template for the menu. For chromeless mode set the value to “custom”. This informs

the menu to use the custom colors defined in the **style** object, below. If not set, then the Color Theme set in the widget console is used.

skin: the type of skin to use for the menu. For chromeless mode set the value to “1”, this give the menu a flat skin. For a 3D look set the value to “0”. If no value is set then the menu will display the 3D skin.

height: value to set the menu height in pixels

width: value to set the menu width in pixels

style object: defines the color theme for all aspects of the menu. If any of these values are not set then the values found in the Widget Console Color Theme – Menu section – will be used.

```
style: {menu : {background : 0xCCCCCC,chrome : 0xCCCCCC}},
```

menu object: define the colors for the core sharing menu

background: HEX value of the menu background color, for example 0xCCCCCC – grey

chrome: HEX value of the menu outline, for example 0xCCCCCC – grey

foreground: HEX value of the font color, for example 0xFFFFFFFF - black

title: HEX value of the font color for dialog titles, for example 0xFFFFFFFF - black

hover: HEX value of mouse over highlight color, for example 0xAAAAAA – dark grey

display object: defines how the sharing menu is displayed.

```
display: {single_tab : 'all', menu_nav:{close:{show: false}}}
```

single_tab: value of the menu tab to display without chrome. For chromeless mode set the value to “all”, which displays the All tab. To display the Popular tab set the value to “popular”. If no value is set then the full menu, with all the chrome, is displayed.

menu_nav: the object to define the menu navigation elements

close: display the menu close button. For chromeless mode set the value to “false”, this hides the close button and allows you to implement your own close method for the menu. To display the close button set the value “true”. If no value is set then the close button will be displayed.

Displaying Configuration Option in Chromeless Menu Mode

For widgets that have end user configurable parameters you will need a method to display the Options panel within the sharing menu. This is simply accomplished by setting the show() method parameter to “options”, e.g. **kernel.menu.show(“options”)**; This will launch the menu with the Options panel displayed. Note, that if your widget does not any configuration parameters setting this parameter will have no effect.

Side note: set the parameter to “info” to display the widgets Info panel.