Exosite

Boot Camp 2015

Portals Labs

Portals Labs

Notes:
Javascript runs in the browser and has the following caveats:

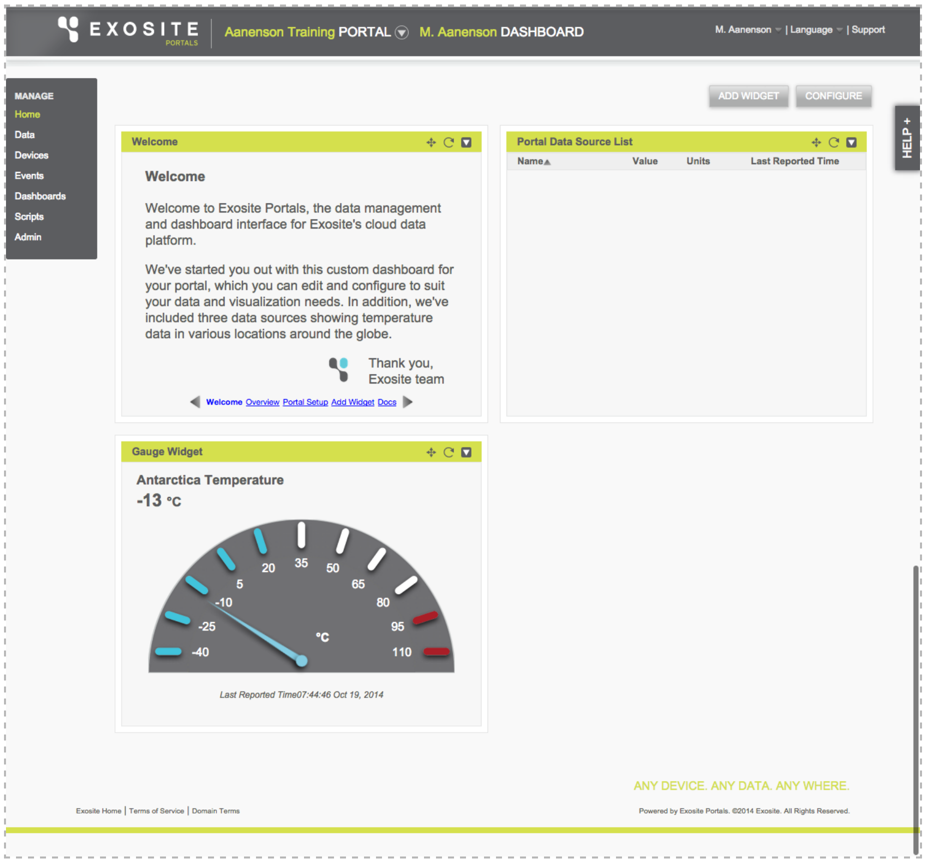
a) It can be malicious, which is why if you make your dashboard public, there will be a button that shows up that says Execute Javascript to any user who visits it. This is not true when you create domain widgets for your custom Subdomain, we'll get to this in another lab for subdomains.
b) Javascript can really do a number on your browser if you write something incorrectly, like a while loop that just runs and runs without stopping.

Exosite's Portal widget editor also has a couple of caveats:

a) Keep your code in a file on your computer, edit it with a code editor and then copy over into the widget. Use a code repository to version control it also. If a widget is deleted, it's gone.
b) If you have an error, it simply tells you there is a syntax error but doesn't tell you where. Exosite is working on an improved editor window. This is why having in a version control system will help a lot to check for changes that may have caused the error.
c) Each Custom widget has a max of 2MB of code size

Lab 1 – Create a custom dashboard using default widgets

To get started, let's assume you have the default dashboard that looks something like this:


EXOSITE PORTALS
Aanenson Training PORTAL
M. Aanenson DASHBOARD
M. Aanenson | Language | Support
MANAGE
Home
Data
Devices
Events
Dashboards
Scripts
Admin
ADD WIDGET
CONFIGURE
HELP +
Gauge Widget
ANY DEVICE. ANY DATA. ANY WHERE.
Exosite Home | Terms of Service | Domain Terms
Powered by Exosite Portals. ©2014 Exosite. All Rights Reserved.

Welcome

Welcome

Welcome to Exosite Portals, the data management and dashboard interface for Exosite's cloud data platform.

We've started you out with this custom dashboard for your portal, which you can edit and configure to suit your data and visualization needs. In addition, we've included three data sources showing temperature data in various locations around the globe.

Thank you,
Exosite team

Welcome Overview Portal Setup Add Widget Docs

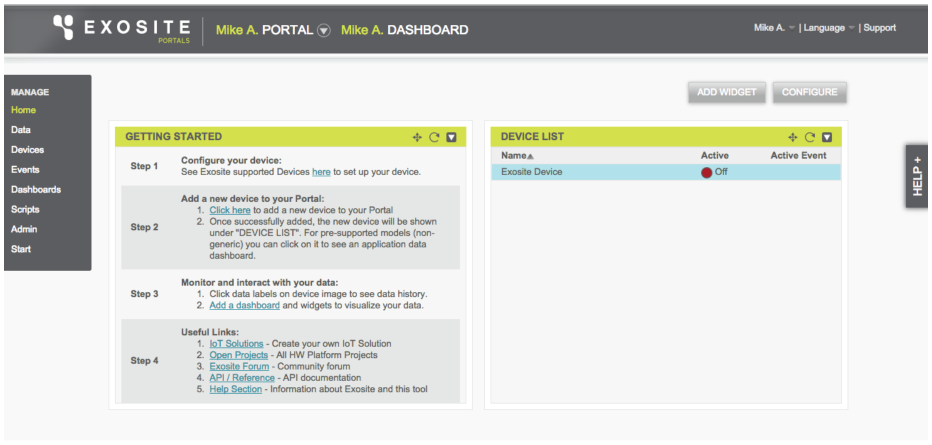
Portal Data Source List

Name	Value	Units	Last Reported Time

Antarctica Temperature
-13 °C

Last Reported Time07:44:46 Oct 19, 2014

or



GETTING STARTED

Step 1 Configure your device:
See Exosite supported Devices here to set up your device.

Step 2 Add a new device to your Portal:
1. Click here to add a new device to your Portal
2. Once successfully added, the new device will be shown under "DEVICE LIST". For pre-supported models (non-generic) you can click on it to see an application data dashboard.

Step 3 Monitor and interact with your data:
1. Click data labels on device image to see data history.
2. Add a dashboard and widgets to visualize your data.

Step 4 Useful Links:
1. IoT Solutions - Create your own IoT Solution
2. Open Projects - All HW Platform Projects
3. Exosite Forum - Community forum
4. API / Reference - API documentation
5. Help Section - Information about Exosite and this tool

DEVICE LIST

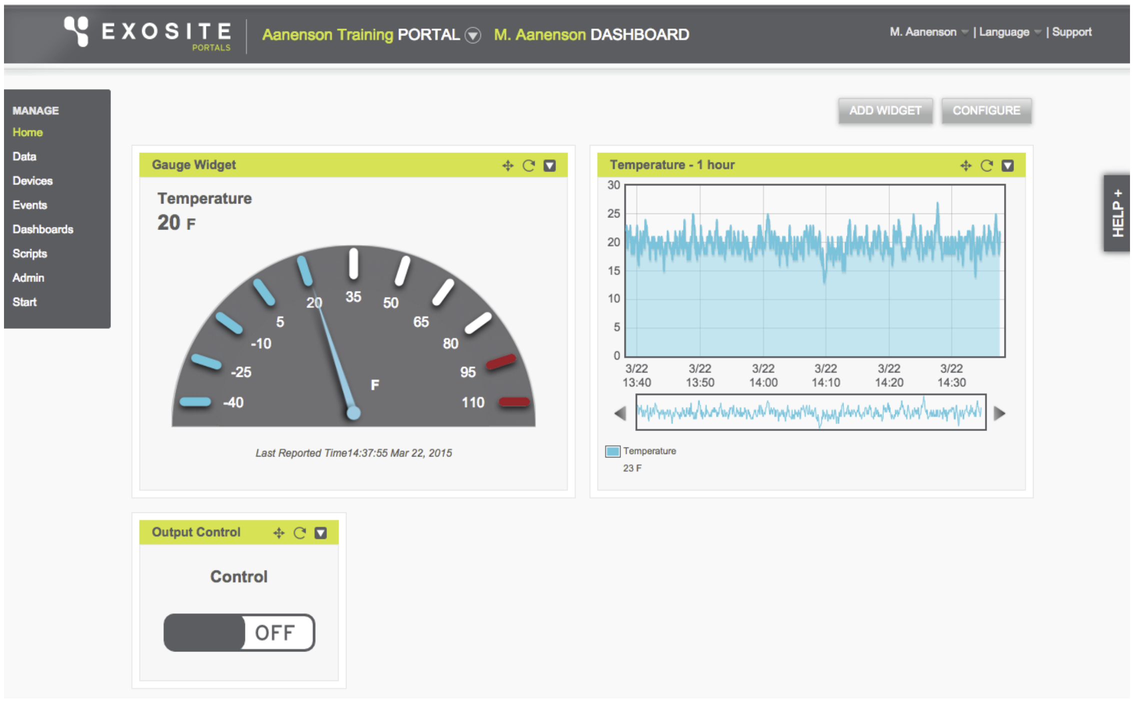
Name	Active	Active Event
Exosite Device	● Off	

The first thing we are going to do is delete the these default widgets and then we'll start to build out a custom dashboard.

1. On each default widget on your home custom dashboard, select the little triangle in the upper-right hand corner of the widget, select 'delete' and confirm.

2. Do the same for the the other widgets.
3. Attach the ‘Gauge Widget’ to your ‘temperature’ data source from the One Platform Labs (emulating data script lab) by clicking on the widget drop down arrow and select ‘Edit’. There should be a form menu in the widget options for ‘Default Active Data’, select your Temperature data source here. Also set the Refresh rate to 10 instead of 60.
4. Save the widget options. You should now see your Temperature data source on the gauge.
5. Edit the widget again to change the color scheme to be blue, green, yellow and change the scale to be appropriate for degrees Celsius.
6. Next, let’s add a Line Graph widget. Click ‘ADD WIDGET’ button.
7. Select widget type ‘Line Graph’ and call it ‘Temperature - 1 hour’, then hit ‘Continue’.
8. Next set the widget options as follows:
    a. Time Window Range: 1 hour
    b. Leave the min/max
    c. Default Active Data Source: <Select your Temperature data port>
    d. Refresh Rate: 10 seconds
    e. Leave other settings as is.
    f. Save.
9. Next, let’s add an On/Off widget. Click ‘ADD WIDGET’ button.
10. Select widget type ‘On/Off Switch’ and call it Control’, then hit ‘Continue’.
11. Next set the widget options as follows:
    a. Widget Size: Small 1x1
    b. Data Source: <Select the Control data port>
    c. Refresh Rate: 0 seconds
    d. Save.
12. In Exoline, run the following command using your Device client’s CIK.
    a. `$ exo read <DEVICE CIK> control --follow`
13. Now click the On/Off button a few times in the portal dashboard and watch the terminal window for your exo read command output.

You should have a dashboard that looks something like this now.


EXOSITE PORTALS
Aanenson Training PORTAL
M. Aanenson DASHBOARD
M. Aanenson | Language | Support
MANAGE
Home
Data
Devices
Events
Dashboards
Scripts
Admin
Start
ADD WIDGET
CONFIGURE
HELP +
Gauge Widget
Temperature
20 F
-40
-25
-10
5
20
35
50
65
80
95
110
F
Last Reported Time14:37:55 Mar 22, 2015
Temperature - 1 hour
Temperature
23 F
Output Control
Control
OFF

Lab 2 - Adding Users as Roles

Portals has the option to add other users in various roles to your portal. When you signed-up, portals.exosite.com created this portal for you, with you as the owner. You can add others with level ‘manage’ or ‘viewer’.

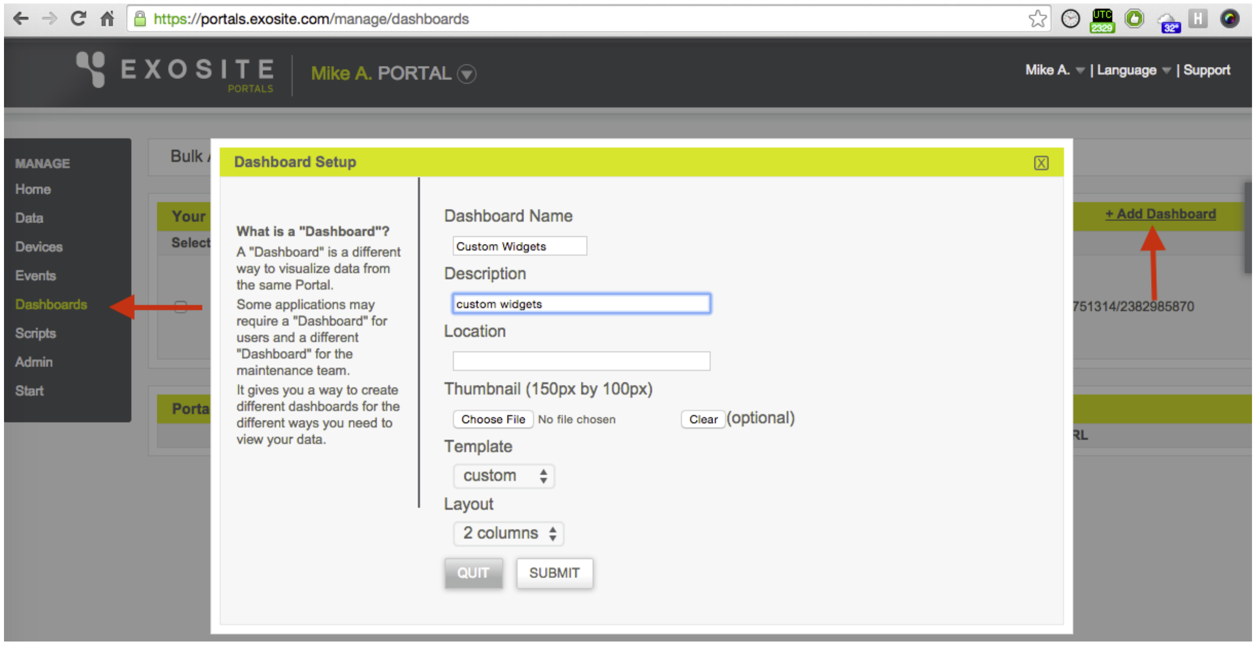
1. In your portal, click on the ‘Admin’ menu item on the left side manage menu.
2. In the Portal Roles box, add your instructor or someone else as a viewer, entering their email address and select ‘Viewer’ role and selecting your your custom dashboard you were just editing.
3. Click the ‘Invite User’ button.
4. Check with that user if they can log-in and view your dashboard.

Portals has an advanced user / group level that is built into it’s Portals API.

Lab 3 - Add a custom widget - hello world
Custom Widgets allow developers to create UI features that meet the application needs they are looking for in their product. They can also be used to offset or provide a different way to do things in Portals than the default functionality with the out of box Portals UI, such as creating a custom Add Device window, a custom Device Table, etc.
http://docs.exosite.com/widget/

This lab will get you started on creating a basic custom widget in it's most basic form. Let's do this on a new custom dashboard, to keep it separated from the one with the off the shelf widgets on it.

1. Click on the 'Dashboards' menu option on the left side manage menu.
2. Click on '+Add Dashboards' to create a new custom dashboard, use the following:
    a. Dashboard Name: Custom Widgets
    b. Description: Custom widget development
    c. Location: (leave blank)
    d. Template: Custom
    e. Layout: 2 columns
    f. Submit.

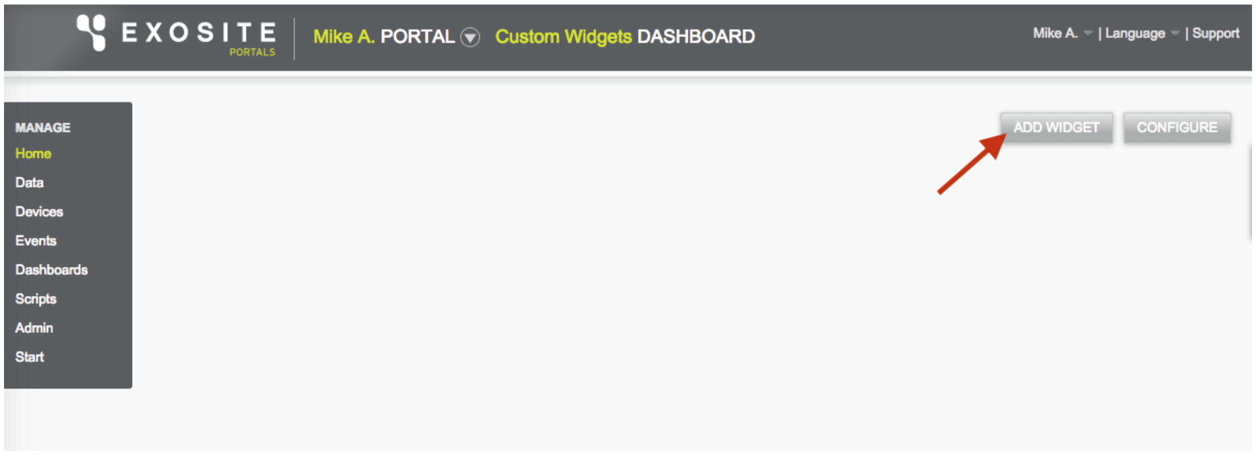


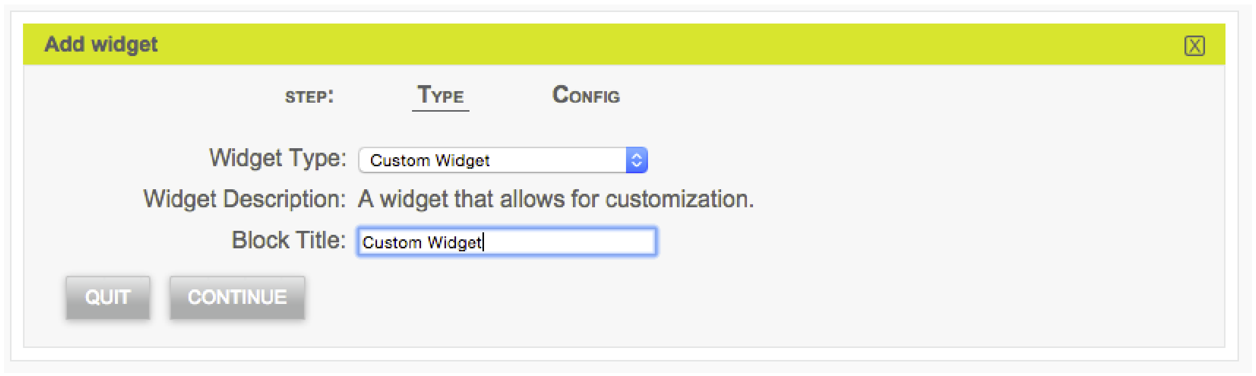
Dashboard Setup

What is a "Dashboard"?
A "Dashboard" is a different way to visualize data from the same Portal.
Some applications may require a "Dashboard" for users and a different "Dashboard" for the maintenance team.
It gives you a way to create different dashboards for the different ways you need to view your data.

Dashboard Name: Custom Widgets
Description: custom widgets
Location:
Thumbnail (150px by 100px): Choose File No file chosen Clear (optional)
Template: custom
Layout: 2 columns

3. Now click on your new Dashboard in the dashboard table.
4. Notice, you should have a blank white dashboard. Also at the top, the Portals drop-down menu will let you switch between your dashboards.
5. Click the 'ADD WIDGET' button on your new custom dashboard and select 'Custom Widget' for widget type, giving it a name of 'Custom Widget'.

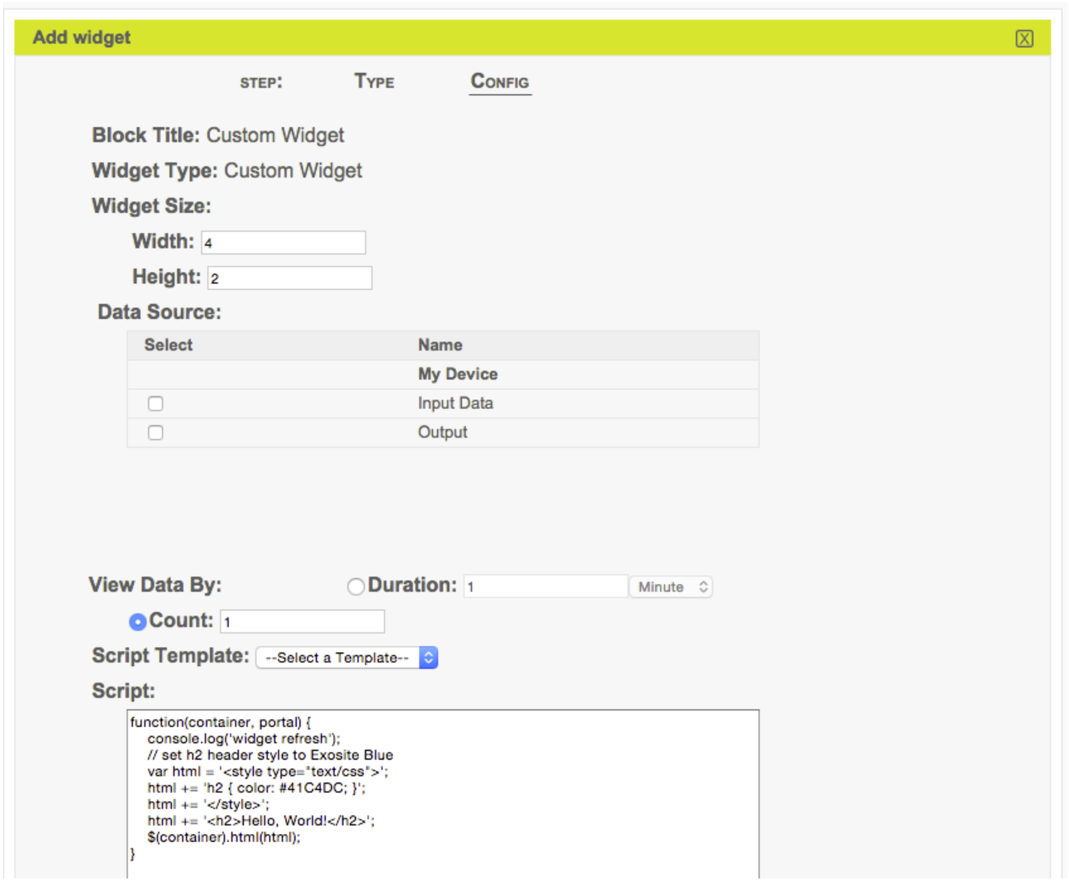




Widget Type: Custom Widget
Widget Description: A widget that allows for customization.
Block Title: Custom Widget

6. Next set the size to Width: 4 and Height: 2.
7. Leave everything else alone except insert this code into the Script window:

```
function(container, portal) {
    console.log('widget refresh');
    // set h2 header style to Exosite Blue
    var html = '<style type="text/css">';
    html += 'h2 { color: #41C4DC; }';
    html += '</style>';
    html += '<h2>Hello, World!</h2>';
    $(container).html(html);
}
```



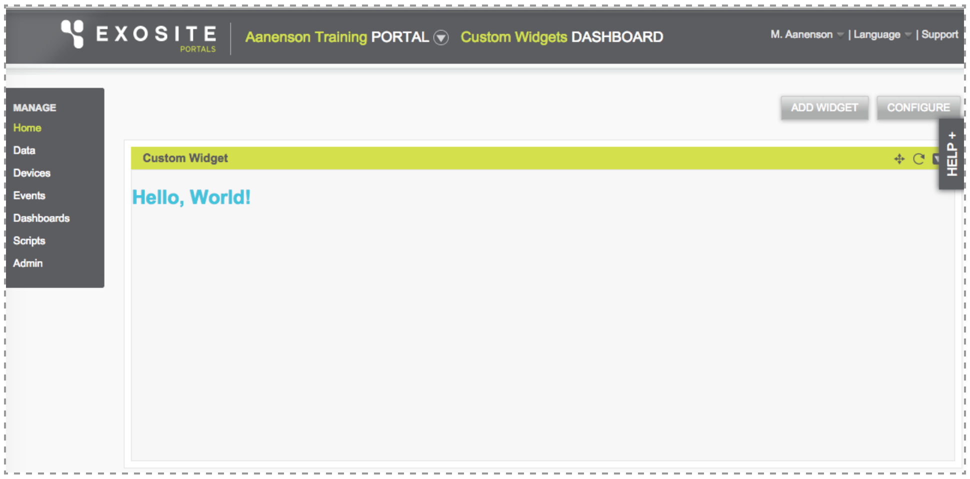
Block Title: Custom Widget
Widget Type: Custom Widget
Widget Size:
Width: 4
Height: 2
Data Source:

Select	Name
	My Device
☐	Input Data
☐	Output

View Data By: ○ Duration: 1 Minute
● Count: 1
Script Template: --Select a Template--
Script:

```
function(container, portal) {
    console.log('widget refresh');
    // set h2 header style to Exosite Blue
    var html = '<style type="text/css">';
    html += 'h2 { color: #41C4DC; }';
    html += '</style>';
    html += '<h2>Hello, World!</h2>';
    $(container).html(html);
}
```

8. Click ‘Save’
9. You’ve created a Javascript widget now!
10. Examine what happens when it loads by using the browser's built-in developer console. In Chrome got to the ‘View’ menu and select Developer -> Javascript Console. You can do this also by right clicking on the page and selecting ‘Inspect Element’.
11. Do you see your ‘Widget Refresh’ message printed out?



Lab 4 - Make custom widget that reads data
The Portals Custom Widget API includes a read and write javascript functions. These functions take a specific client and alias dataport. Let’s modify our custom widget to now show a data value that will reload with the click of a link.

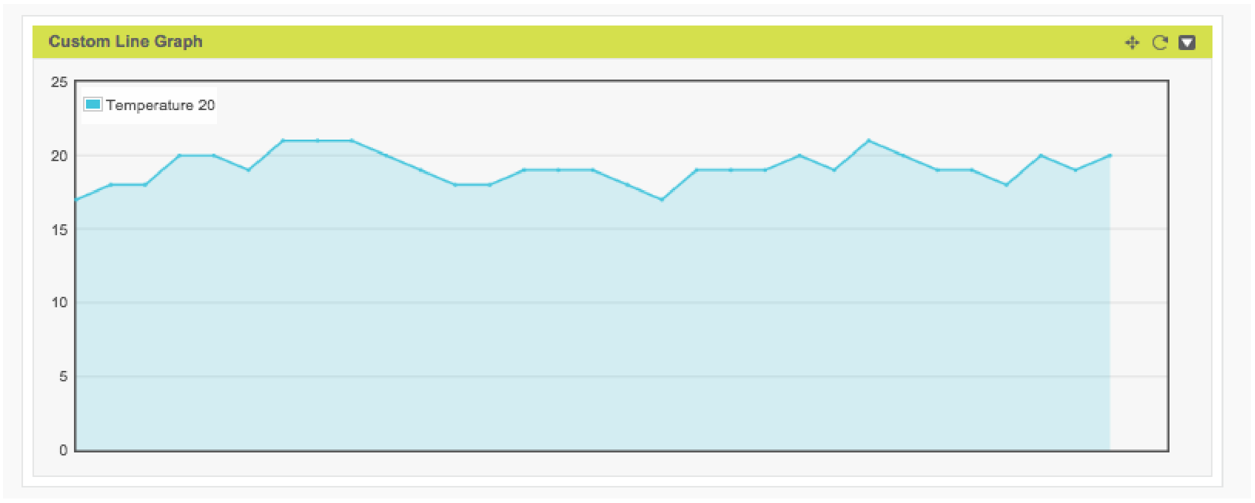
http://docs.exosite.com/widget/

1. Start by making sure your device in your Exosite Portal has an ‘alias’. By default, when creating a generic device through the Portals Add Device window, it does not. Go to the ‘Devices’ page, click your device, and set an Alias to ‘mydevice’, save this.
2. Go to your custom dashboard ‘Custom Widgets’ and edit the current widget that you created in the previous lab, that likely says ‘Hello World!’ on it right now.
3. When the widget editor opens, copy and paste the code from the file ‘custom_basic_read.js’.
4. Make sure to select the **‘Temperature’** dataport from the widget options.
5. Save the widget and run it.
6. After ~10 seconds, click the ‘reload’ link, did the the value and timestamp change?
7. Now let’s make this widget refresh on it’s own using the JavaScript setTimeout function.
8. Go back into this widget code and uncomment out the section at the bottom for self refresh. **Comment out the ‘reload();’ line right above it using ‘//’.**
9. Now save the widget and run it again.

Lab 5 - Add a custom widget - scrolling line graph
Building upon the idea that a widget can make read requests, let's try building a line graph that updates itself quickly by having the read command ask for many data points instead of just 1.

The code for this lab can be found here:
https://github.com/exosite-garage/custom_widget_scripts/tree/master/line_graphs/line_graph_self_refresh

1. Add a new custom widget to your custom dashboard.
2. Call it 'Custom Line Graph'
3. Width: 4, Height: 2
4. Select your Temperature dataport again from the Data Source list.
5. Make sure the refresh is left at 0. Leave Count and Duration as is.
6. Copy the code into the widget and hit 'Update'.
7. Do a full browser page refresh to make sure to restart any javascript functions that may be running from the other widget.
8. Your widget should look something like this:



9. Can you change the color?
10. Can you change the time window which is currently 5 minutes?

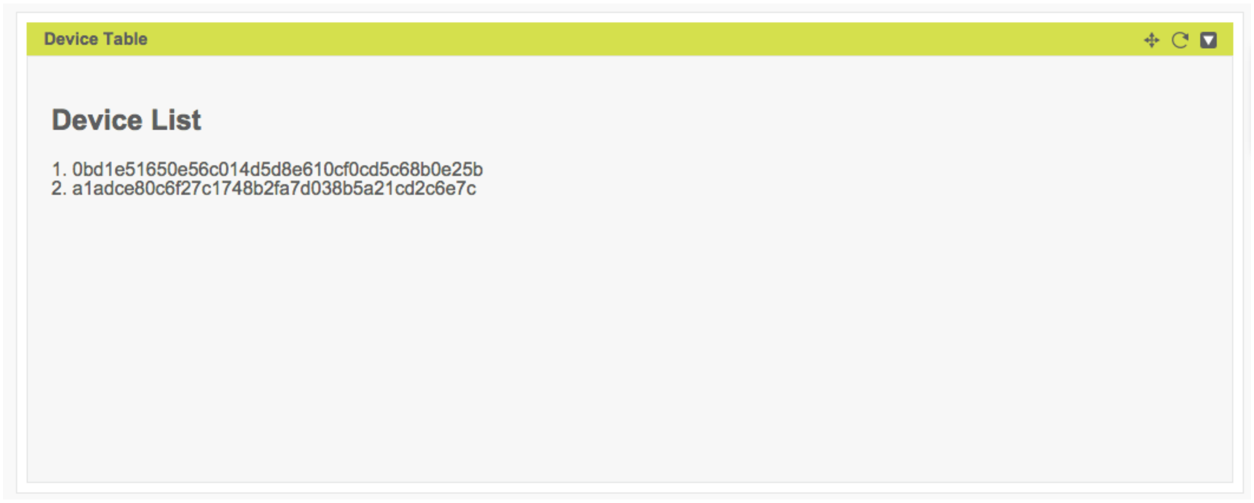
Note, when updating, do a full page refresh.

Lab 6 - Using the Portals API in widgets
Portals now has it's own API that allows external applications or Portals widgets make calls to like get a list of devices owned by the portal, get groups associated with a user, etc. The key with the Portals API is that it does not use CIKs for authentication but instead uses the log-in authorization of the user.

Let's create a device table that will list any device clients in this portal. This lab will make calls to the Portals API which is documented here http://docs.exosite.com/portals/ and makes use of something called 'AJAX' (Asynchronous JavaScript + XML). What AJAX does is allow Javascript to make HTTP requests to post or get information from the server after the page loads. This means you can load your HTML and some data, but then make calls to get updated information like a list of devices, device info, data, etc. You get to control how these requests are made, so you can do things like load long lists yet show the user the start of the list, they may not even notice the rest of the list isn't there yet if they have to scroll for it.

The code for this is found in the repository.

1. Delete other custom widgets on this page, if any are left.
2. Add a new custom widget to your custom dashboard.
3. Call it 'Custom Device Table'
4. Width: 4, Height: 2
5. Don't select any data or change any options.
6. Make sure the refresh is left at 0. Leave Count and Duration as is.
7. Copy the the code from the file **'device_table_using_portal_api_1**.js' into the widget and hit 'Update'.
8. Do a full browser page refresh to make sure to restart any javascript functions that may be running from the other widget.
9. Your widget should look something like this, which is not all that interesting is it, but it lays the foundation for using the Portals API to get the information you want to display in this widget. This code makes use of the GET PORTAL function call in the API (http://docs.exosite.com/portals/#get-portal) which will return a portals resources such as devices, users, etc. The list of devices only contains each devices RID, which is what you are seeing.

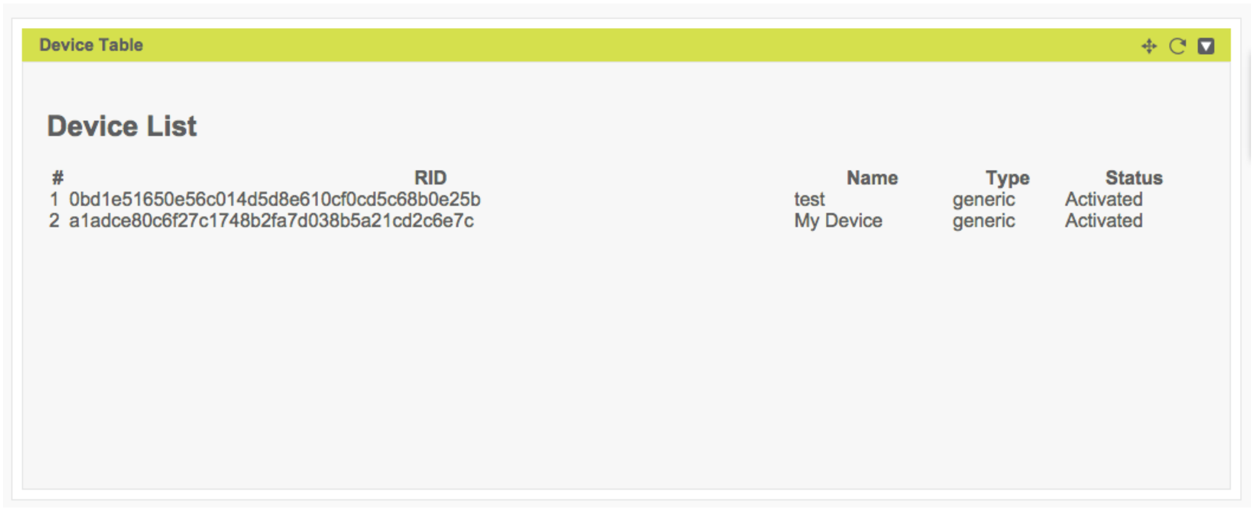


Device Table

Device List

1. 0bd1e51650e56c014d5d8e610cf0cd5c68b0e25b
2. a1adce80c6f27c1748b2fa7d038b5a21cd2c6e7c

10. Let's make this a bit more interesting by using each Devices RID to get more info about it.
11. Now grab the code from the file '**device_table_using_portal_api_2.js**' and update your widget with it and refresh the page.
12. What do you see now?



Device Table

Device List

#	RID	Name	Type	Status
1	0bd1e51650e56c014d5d8e610cf0cd5c68b0e25b	test	generic	Activated
2	a1adce80c6f27c1748b2fa7d038b5a21cd2c6e7c	My Device	generic	Activated

13. This very generic HTML Table (no styling) shows the RID, the name, type, and CIK's activation status. How did we get this?
14. Add another device in Portals, does it show up?
15. How can this be used to build more application functionality?

Advanced:
Here is another example device table that is more advanced, https://github.com/exosite-garage/custom_widget_scripts/blob/master/tables/device_table_using_jquery/device_table_using_dataTables.js