**Open Integration – Getting Started**

The purpose of an Open Integration architecture is to allow you to interface your data with the analytic tools of the Mapping Broadband Health in America platform (<https://www.fcc.gov/reports-research/maps/connect2health>) so you can examine the picture of connectivity and health with respect to datasets of interest to you. Simply put, it puts the power of this platform’s visual analytics into your hands and your data. This can be done quickly once your data is in the correct format. Steps I and II below help you get your data into the correct format. Once you have saved the data in the correct format, Step III shows you how to modify the code (HTML) in 3 easy steps to integrate your data into the platform interface. There are also completed examples (**Disability Population Files**; **Veterans Population Files**) that you can examine and modify.

**(I) DOWNLOAD EXTENSIBILITY FILES:** Download the extensibility files to your local environment. You will see one HTML file (**extensible.html**) and four folders (**css**, **data**, **img**, **js**). You should save your data to the **data** folder and you will only modify the HTML file (i.e. **extensible.html**).

**(II) SAVE YOUR DATA IN THE CORRECT FORMAT - This is the only hard part (we promise)!**

STEP 1. CONVERT YOUR DATA INTO A JAVASCRIPT OBJECT: Save your dataset as a JavaScript object. An easy way to do this is to convert your CSV or other format to JSON format first, and then assign it to a variable. Because you are going to be 'merging' datasets by county geo-ids, make sure your dataset has:

* data at the county level
* a county FIPS code column
* county FIPS code values as text (not numeric)
* the county FIPS code column named as ‘countyFIPS’.

For example, Autauga county data object would have the following key and value for the county FIPS code: "countyFIPS": "01001".

There are many converters online that can help you convert your data into the JSON format (if your data is saved in CSV format then using a search engine to search for ‘csv to json’ can lead to many resources to convert your data into JSON format in a matter of seconds. Download or copy-and-paste your JSON-formatted data into a text file (e.g. notepad). At this point each row of your data would be between braces ( ‘{ }’ ); the name and value of each variable (column) corresponding to that row should be visible as well. The following is an example of how your data should look at this point (the specific names and values would depend on your data; the column with the county FIPS code should be named ‘countyFIPS’ prior to converting it to JSON format):

**[**

**{**

"countyFIPS"**:** "01001"**,**

"mapdata\_pop\_2014"**:** 55395**,**

"STATECODE"**:** 1**,**

"pop\_pct\_65andabove"**:** 14**,**

"COUNTYNAME"**:** "Autauga County"**,**

"STATE"**:** "AL"**,**

"vets\_all"**:** 6063**,**

"vets\_all\_pct"**:** 11**,**

"national\_vet\_pop"**:** 7**,**

"vets\_all\_include"**:** 1

**},...**

**]**

STEP 2. NAME YOUR DATA: Assign your data objects to a variable. We have included two example files (**Disability Population Files**; **Veterans Population Files**). In the **Veterans Population Files** folder the data variable can be found in the **data** folder, in the **vetspop.js** file. There it's defined as:

***var*** vetsdata**=[**

**{**

"countyFIPS"**:** "01001"**,**

"mapdata\_pop\_2014"**:** 55395**,**

"STATECODE"**:** 1**,**

"pop\_pct\_65andabove"**:** 14**,**

"COUNTYNAME"**:** "Autauga County"**,**

"STATE"**:** "AL"**,**

"vets\_all"**:** 6063**,**

"vets\_all\_pct"**:** 11**,**

"national\_vet\_pop"**:** 7**,**

"vets\_all\_include"**:** 1

**},...**

STEP 3. SAVE YOUR DATA: Once you have the data in the correct format (see the **disability.js** or the **vetspop.js** files as examples), save it as a JavaScript file to the **data** folder. You can open these files with notepad or any text editor.

**(III) MODIFY THE HTML - the easy part...we have done all the coding for you!**

Open the **extensible.html** file and scroll down to the Open Integration section...it comes with instructions in comments and looks like this:

<!--

(1) SET OPTION VALUES: The format for setting option values is: VARIABLENAME$LOWERBOUNDARY\_UPPERBOUNDARY

The code parses everything before the '$' sign as the variable name; and everything after the '$' sign as the numeric range.

-->

<div class=**"row row-space"** style=**"background-color:lightgray; border: solid 0.1em darkblue; padding-top:0.5em"**>

<div class=**"col-sm-5"**>

<div class=**"data-label"**>

<label for=**"ov-select-extend"**>**DATA\_NAME:**</label>

</div>

</div>

<div class=**"col-sm-7 no-left"**>

<select id=**"ov-select-extend"** class=**"selectpicker select-advanced advanced-extend"**>

<optgroup label=**"ENTER\_SUB\_GROUP\_NAME 1"**>

<option value=**""** selected>**None** </option>

<option value=**"VARNAME1$LB\_UB"**>**LB\_UB**</option>

<option value=**"VARNAME1$LB\_UB"**>**LB\_UB**</option>

</optgroup>

<optgroup label=**"ENTER\_SUB\_GROUP\_NAME 2"**>

<option value=**"VARNAME2$LB\_UB"**>**LB\_UB**</option>

<option value=**"VARNAME2$LB\_UB"**>**LB\_UB**</option>

</optgroup>

</select>

</div>

</div>

<!--

(2) ENTER PATH TO FILE WITH OBJECTS W/ GEOGRAPHIC FEATURES & DATA; MAKE SURE COUNTY FIPS CODES ARE TEXT VALUES IN THE OBJECT AND IS NAMED AS: countyFIPS...SO FOR AUTAUGA COUNTY IT SHOULD BE: "countyFIPS": "01001"

-->

<script src=**"data/PATH\_TO\_DATA\_OBJECT.js"**></script>

<!--

(3) SET extsdata VARIABLE BELOW TO REFER TO OBJECT CONTAINING GEOGRAPHIC FEATURES

-->

<script>

***var*** extsdata**=**OBJECT\_NAME\_W\_GEOGRAPHIC\_FEATURES**;**

</script>

<!--

To increase page load speed, the above scripts can be placed at the bottom of the <body> element

-->

STEP 1. SET OPTION VALUES: This is where you define the variables in the dataset that you want displayed and the value levels that you want to use as selection options. The format for setting option values is: VARIABLENAME$LOWERBOUNDARY\_UPPERBOUNDARY

The code parses everything before the '$' sign as the variable name; and everything after the '$' sign as the numeric range. For example, if your variable name is 'vets\_all\_pct' representing the percentage of the total population in a given county that are veterans, then you would place that name before the '$' sign: vets\_all\_pct$LOWERBOUNDARY\_UPPERBOUNDARY.

Then if you want the first option to be to show all counties where veteran population is 0-5%, then '0' is the lower boundary and '5' is the upper boundary (separated by an underscore character: \_ )...the option value would be: vets\_all\_pct$0\_5. You can modify the text between the <option></option> tags to let the user know what values they would be selecting. A completed example from **Veterans Population Files** is below ('vets\_all\_pct' and 'vets\_all' are variables in the dataset in the **vetspops.js** data file):

<div class=**"row row-space"**>

<div class=**"col-sm-5"**>

<div class=**"data-label"**>

<label for=**"ov-select-extend"**>**Vet Pop Filter:**</label>

</div>

</div>

<div class=**"col-sm-7 no-left"**>

<select id=**"ov-select-extend"** class=**"selectpicker select-advanced advanced-extend"**>

<optgroup label=**"Veteran Pop Filter 1"**>

<option value=**""** selected>**None** </option>

<option value=**"vets\_all\_pct$0\_5"**>**0-5% vet pop.** </option>

<option value=**"vets\_all\_pct$5\_10"**>**5-10% vet pop.** </option>

<option value=**"vets\_all\_pct$10\_15"**>**10-15% vet pop.** </option>

<option value=**"vets\_all\_pct$15\_35"**>**15%+ vet pop.** </option>

</optgroup>

<optgroup label=**"Veteran Pop Filter 2"**>

<option value=**"vets\_all\_pct$0\_10"**>**0-10% vet pop.** </option>

<option value=**"vets\_all\_pct$10\_35"**>**10%+ vet pop.** </option>

</optgroup>

<optgroup label=**"Veteran Pop Filter 3"**>

<option value=**"vets\_all$0\_5000"**>**0-5000 vet pop.** </option>

<option value=**"vets\_all$5000\_1000000"**>**5000+ vet pop.** </option>

</optgroup>

</select>

</div>

</div>

STEP 2. ENTER PATH TO FILE WITH OBJECTS W/ GEOGRAPHIC FEATURES & DATA: In the portion of the code in the <script> tags, modify the path to your data file by simply entering the name of your data file. To do this, replace the text “PATH\_TO\_DATA\_OBJECT” with the name of the data file that you saved into the data folder:

<script src=**"data/PATH\_TO\_DATA\_OBJECT.js"**></script>

For the example in **Veterans Population Files** you will see that the data file name is **vetspops.js** and is saved in the folder named ‘**data’**. Therefore, the path in the <script> tags is changed to the following:

<script src=**"data/vetspops.js"**></script>

STEP 3. LET THE PLATFORM KNOW THE NAME OF YOUR DATA OBJECTS: To do this, set the ‘extsdata' variable to equal to the name you gave your data object:

<script>

***var*** extsdata**=**OBJECT\_NAME\_W\_GEOGRAPHIC\_FEATURES**;**

</script>

In the example in the **Veterans Population Files** we gave the name ‘vetsdata’ to the data objects with veterans data (see above: (II) SAVE YOUR DATA IN THE CORRECT FORMAT; Step 2):

<script>

***var*** extsdata**=**vetsdata**;**

</script>

You are done! Once the above steps are completed, you can rename and open the extensible.html file in your browser and you should see that a feature to manipulate your dataset has been added to the user interface. You should now be able to manipulate the map and see how the picture of connectivity and health changes based on varying levels of your data variable.

The complete code for the example we just examined is given below (i.e. this is the only part of the HTML file you need to modify once you have saved your dataset to the data folder):

<!--

(1) SET OPTION VALUES: The format for setting option values is: VARIABLENAME$LOWERBOUNDARY\_UPPERBOUNDARY

The code parses everything before the '$' sign as the variable name; and everything after the '$' sign as the numeric range.

-->

<div class=**"row row-space"**>

<div class=**"col-sm-5"**>

<div class=**"data-label"**>

<label for=**"ov-select-extend"**>**Vet Pop Filter:**</label>

</div>

</div>

<div class=**"col-sm-7 no-left"**>

<select id=**"ov-select-extend"** class=**"selectpicker select-advanced advanced-extend"**>

<optgroup label=**"Veteran Pop Filter 1"**>

<option value=**""** selected>**None** </option>

<option value=**"vets\_all\_pct$0\_5"**>**0-5% vet pop.** </option>

<option value=**"vets\_all\_pct$5\_10"**>**5-10% vet pop.** </option>

<option value=**"vets\_all\_pct$10\_15"**>**10-15% vet pop.** </option>

<option value=**"vets\_all\_pct$15\_35"**>**15%+ vet pop.** </option>

</optgroup>

<optgroup label=**"Veteran Pop Filter 2"**>

<option value=**"vets\_all\_pct$0\_10"**>**0-10% vet pop.** </option>

<option value=**"vets\_all\_pct$10\_35"**>**10%+ vet pop.** </option>

</optgroup>

<optgroup label=**"Veteran Pop Filter 3"**>

<option value=**"vets\_all$0\_5000"**>**0-5000 vet pop.** </option>

<option value=**"vets\_all$5000\_1000000"**>**5000+ vet pop.** </option>

</optgroup>

</select>

</div>

</div>

<!--

(2) ENTER PATH TO FILE WITH OBJECTS W/ GEOGRAPHIC FEATURES & DATA; MAKE SURE COUNTY FIPS CODES ARE TEXT VALUES IN THE OBJECT AND IS NAMED AS: countyFIPS...SO FOR AUTAUGA COUNTY IT SHOULD BE: "countyFIPS": "01001"

-->

<script src=**"data/vetspops.js"**></script>

<!--

(3) SET extsdata VARIABLE BELOW TO REFER TO OBJECT CONTAINING GEOGRAPHIC FEATURES

-->

<script>

***var*** extsdata**=**vetsdata**;**

</script>

<!--

To increase page load speed, the above scripts can be placed at the bottom of the <body> element

-->