

# GeoIQ **GeoIQ Javascript API** Documentation

## Introduction

The GeoIQ Javascript API provides a quick and easy way to add GeoIQ content and analysis to existing or new web applications or mashups.

Using the API, you will be able to include content such as heatmaps, concentration indices, or intersection indices in your own content, and include the data provided by GeoIQ, as well as being able to include some of your own data in the analytics.

To use the GeoIQ API, all you will need is some basic Javascript and web development knowledge, and a free API key.

## Quick Start

To get started, you must first import the GeoIQ Javascript API into your HTML document. To do so, use the following code snippet:

```
<script src="http://www.geoiq.com/api/v1/geoiq.js"
    type="text/javascript">
</script>
```

Next, we will create the default GeoIQ workspace and handler callback, which contains all the properties and methods we will be using to manipulate and retrieve the analysis imagery. The example code that follows should be placed inside of a javascript "script" tag (`<script type="text/javascript">...`)

```
// GeoIQ Custom Handler Callback
GIQCustomHandler.prototype.onReady = function() {
    // References the top level "ws" Workspace object created below

    // Retrieve a list of data display groups from GeoIQ
    var display_groups = ws.getDisplayGroupsHash();
    // Create a heatmap instance with default parameters
    var heatmap = ws.heatmap();
    // Add the "Railways" display group to the heatmap
    heatmap.add("Railways", // DisplayGroup name?
        display_groups["Railways"][0] // DisplayGroup instance
    );
    // Create a simple image object to hold the heatmap
    var img = new Image();
    img.src = heatmap.getURL();
}
```

```

}

var h = new GIQCustomHandler();
var ws = GIQInitWorkspace(h, "API_KEY_GOES_HERE");

```

Be sure that you replace “KEY\_GOES\_HERE” with the API key you received after signing up.

## Integration with Google Maps

*Note: You will need to obtain a Google Maps API key in order to use the Google Maps API. A key can be obtained from google at this URL: <http://www.google.com/apis/maps/signup.html>*

The GeolQ API provides a [GIQGoogleOverlay](#) object which provides all the functionality needed to overlay GeolQ imagery inside of a Google Maps instance.

### Example

See the previous example or detailed documentation for an example of how to create the heatmap object used in this example.

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>GeoIQ API Google Maps Example</title>
    <script src="http://www.geoiq.com/api/v1/geoiq.js"
      type="text/javascript"></script>
    <!-- Be sure to include the GeoIQ Google-specific API code -->
    <script src="http://www.geoiq.com/api/v1/geoiq_google.js"
      type="text/javascript"></script>
    <script type="text/javascript">
//<![CDATA[
  // GeoIQ Custom Handler Callback
  GIQCustomHandler.prototype.onReady = function() {
    // References the top level "ws" Workspace object created below

    // Retrieve a list of data display groups from GeoIQ
    var display_groups = ws.getDisplayGroupsHash();
    // Create a heatmap instance with default parameters
    var heatmap = ws.heatmap();
    // Add the "Railways" display group to the heatmap
    heatmap.add("Railways", // DisplayGroup name
      display_groups["Railways"][0] // Classification Attr.
    );

    // See the Google Maps API documentation for more details
    // on the creation and usage of Google Map objects
    gmap = new GMap2(document.getElementById('map'));
    google_overlay = new GIQGoogleOverlay(heatmap.getURL());
    gmap.addOverlay(google_overlay);
  }

  function load() { // Call this from the document body's "onload" event

```

```

        var h = new GIQCustomHandler();
        ws = GIQInitWorkspace(h,
                               "Put your API Key Here");

        h.onReady();
    }
//]]>
</script>
</head>
<body onload="load()">
    <!-- 500x500 div to hold the map -->
    <div id="map"
        style="width: 500px; height: 500px; position: absolute; left:
10px; top: 125px"></div>
    </body>
</html>

```

## API Reference

### Top Level Methods

`GIQInitWorkspace(handler, api_key)`

The `GIQInitWorkspace` method initializes the GeolQ API and sets up an initial workspace for further use. You must call this method before any other GeolQ API methods.

#### Arguments:

***handler*** : An instance of `GIQCustomHandler` that will handle callbacks from the GeolQ API

***api\_key*** : Your GeolQ API key

#### Returns:

An instance of a `GIQWorkspace` class ready for use

### GIQWorkspace Object

The `GIQWorkspace` object encapsulates a single workspace containing information on display groups available for display and analysis, as well as providing methods for creating raster imagery, such as heatmaps.

This class should never be instantiated directly, rather, the instance returned from `GIQInitWorkspace` should be used.

#### Methods:

`getDisplayGroupsHash()`

Returns a list of display groups available for the workspace. Display groups contain one or more sets of data (geometries and attribute/value pairs) available for rendering and analysis.

Arguments: *None*

Returns: Javascript hash of Display groups and attributes. i.e.:

```
{ "Railways" : ("Tonnage", "Dollar Value"),  
  "Hurricanes" : ("Magnitude") }
```

`heatmap()`

Generates a heatmap based on the options specified

Arguments: *None*

`sci()`

Generates a spatial concentration index based on the options specified

Arguments: *None*

`intersection()`

Generates an intersection based on the options specified

Arguments: *None*

## **GIQRaster Object**

The GIQRaster class encapsulates all the logic required to retrieve raster analysis imagery from the GeolQ servers, such as a heatmap, spatial concentration index, or intersection.

### Methods:

`getURL()`

Returns the URL to be passed to the GeolQ servers to retrieve the raster imagery, suitable for use in an overlay image.

Arguments: *None*

Returns: URL as a string

`add(dg, attr)`

Adds a display group and one of its attributes to be included in the analysis (such as those returned from `GIQWorkspace.getDisplayGroupsHash()` )

### Arguments:

*dg*: Display group name

*attr*: Attribute name

```
setDistance(distance)
```

Sets the distance weighting used in the raster calculation.

### Arguments:

*distance*: This is a positive integer in the range 0-255, inclusive. Setting the distance weighting to zero will cause any geometries included in the raster calculations to not affect the “heat” of any non-overlapping geometries. Setting distance to 255 will have the opposite effect, including all geometries in the raster analysis into the overall “heat” for a particular area of the map to the maximum possible distance.

Notes: The “distance” argument is a pixel radius at present. In the future, this method or additional methods may be added or updated to allow the distance to be expressed in geographic units.

```
setPoints(points)
```

Adds a collection of lat/long points and ratings to be included in the analysis imagery output.

### Arguments:

*points*: List of point x,y coordinates (longitude,latitude) and ratings, in the format:

(x1,y1,rating1,x2,y2,rating2,...)

All longitude and latitude coordinates are in decimal degrees, and the numbers specified by rating can be any integer between 0-2<sup>31</sup>, the raster calculations and classifications will automatically scale the values used in the raster imagery accordingly based upon the total range of values passed as ratings (higher ratings will show up brighter in the analysis imagery).

See Also: [GIQRaster.addPoint](#), [GIQRaster.clearPoints](#)

### *Example:*

```
var heatmap = ws.heatmap(); // ws is an initialized GIQWorkspace
var points = new Array(-77.34, //x1
                       43.2, //y1
                       1, //rating1
                       -80.23, //x2
```

```
        37.81,      //y2
        5,         //rating2
        57.38,     //x3
        23.4,     //y3
        4         //rating3
    );
    heatmap.setPoints(points); // Add the three points in 'points' to
                              // the heatmap
```

```
addPoint(x, y, rating)
```

Adds a single lat/long point to be included in the analysis imagery output

Arguments:

*x*: X (longitude) coordinate of the point to add

*y*: Y (latitude) coordinate of the point to add

*rating*: Rating associated with the point

See Also: [GIQRaster.setPoints](#), [GIQRaster.clearPoints](#)

```
clearPoints()
```

Clears any points added with [GIQRaster.addPoint](#) or [GIQRaster.setPoints](#)

See Also: [GIQRaster.addPoint](#), [GIQRaster.setPoints](#)

```
setType(type)
```

Sets the type of the raster analysis

Arguments:

*type*: One of “heatmap”, “sci”, or “intersection”

### **GIQGoogleOverlay Object**

The [GIQGoogleOverlay](#) class allows an API user to overlay GeolQ raster analysis imagery on a Google Maps instance.

Methods:

```
GIQGoogleOverlay(url)
```

Constructs a new [GIQGoogleOverlay](#)

Arguments:

*url*: URL of the GeoIQ raster image to be overlaid, such as that returned from `GIQRaster.getURL()`