White Paper:
How QlikTech uses QlikView – Google Maps

Best Practices for integrating Google Maps into Applications.

v1.1

MLW

QlikTech

June 2009

www.qlikview.com

White Paper: How QlikTech uses QlikView – Google Maps

Table of Contents:

Overview of using Google Maps within QlikView……………………………………………3

Scripting Needs for Google Maps………………………………………………………5

Google Map Variables………………………………………………………………………..5

Geo-code Information………………………………………………………………………6

Chart Creation for Google Maps…………………………………………………………7

Additional Information……………………………………………………………………9

Variable Sized Maps…………………………………………………………...………………9

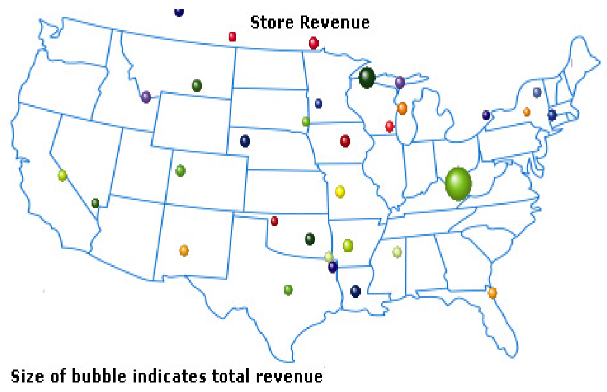
Multiple Maps of Difference Sizes…………………………………………………….………10

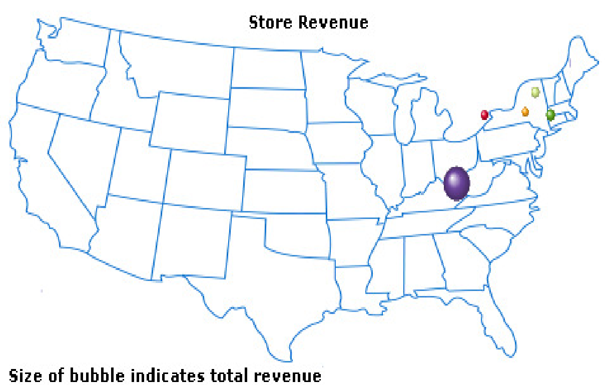
Additional Details for QlikView 8.5……………………………………………………………..11

Google Maps Documentation/Terms of Service……………………………………..…………..11

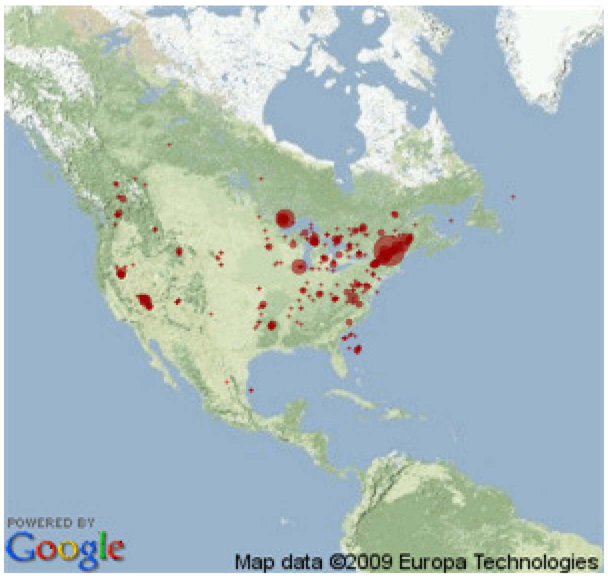
Overview of using Google Maps within QlikView

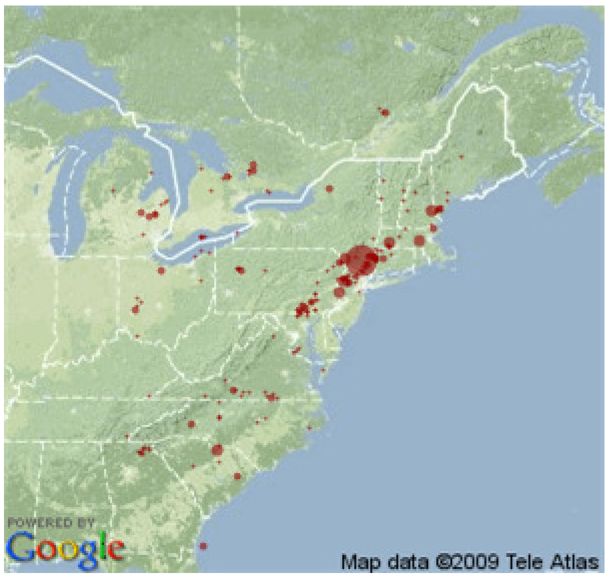
In the past, if a user wanted to show information related to locations on a map, a scatter chart had to be created with a map as a static background image. This was a useful tool for analyzing different data sets based on location specific information. The major drawback to this process was that the maps were not dynamic. If you wanted to zoom in on just the Eastern Region of the map below, the map would not change, it would just remove the data that was not selected as shown:





QlikView has the ability to integrate the use of Google Maps into an application to perform spatial analysis of data points plotted against areas mapped across the globe. This functionality can allow a user to analyze data sets across operational regions, locales, countries, etc. depending on the needs of the organization. These maps are also dynamic and allow for simple zooming in and out of information. The implementation of Google Maps can turn the above maps into the maps shown below:





To successfully integrate Google Maps into a QlikView application, you will need to ensure you have the following:

QlikView 9.x Installation (QlikView 8.5 can also be used – See Additional Information)
Google Maps API Key: This can be obtained from Google at http://code.google.com/apis/maps/signup.html
Geo-code information for your data (i.e., Longitude and Latitude coordinates of the data to be plotted)
A working Internet connection: The maps require access to the internet to render properly.

In addition to the above items, there are several scripting and charting techniques that need to be incorporated into the application for these maps to function properly. This document will explain these methods in detail. In addition to the proper scripting and expression building, additional data with Geo-tagging information also needs to be loaded into the application and properly associated to the data within the application to plot appropriately.

The information in this document gives a user a step-by-step method of creating a single Google Map within QlikView. The examples assume a map size of 400x400 pixels. Information on how to modify the map size as well as create multiple sized maps can be found at the end of the document.

The instructions in this document are based on a QlikView 9.x installation. To implement Google Maps in versions prior to QV 9.x, please refer to the Additional Information section of this document.

This document is meant to be used as both a reference for the proper syntax of the expressions required to build a Google Map within QlikView, as well as a guideline for some additional best practices on simplifying the process of creating these objects going forward. Some of the recommendations mentioned within may not suffice for some user's needs, so each user should determine the best method of implementation to support their implementation.

Scripting Needs for Google Maps

The scripting that is required to implement Google Maps within QlikView can be classified into two separate needs: variable creations and geo-code information loading. We will look at these separately.

Google Map Variables

To use the Google Maps within QlikView, you must first acquire a key from Google. This key only requires you to signup for the use of Google Maps. To get this key code, simply connect to the following URL and follow the directions on screen.

http://code.google.com/apis/maps/signup.html

In order to have the maps render properly, the following cold must be added to the script and then loaded into the application. You must enter the key generated above in the script below where indicated. This script sample is based on a 400x400 pixel Google Map image and uses some default variable names.

```
// Google Maps Key
// get a key here http://code.google.com/apis/maps/signup.html
gmap_key = 'xx';   ←-enter the Google Maps Key here.
max_zoom_level = 14; //maximum value 17
def_zoom_level = 1;
def_map_size = 400;

// Variables required for calculating map
// No need to change these
var_pi180=                    '=pi()/180';
var_lat_offset= '0';
var_mc2=        '=256*pow(2,$(var_zoom))';
var_mc1=        '=256*pow(2,($(var_zoom)-1))';
var_mid_lat=    '=median(Latitude)';
var_mid_long=   '=median(Longitude)';
var_zoom=       '=If(max(aggr(if(max( round(256*pow(2,(_zoom_level -1)))+( Longitude  *((256*pow(2,_zoom_level
))/360)) )-min( round(256*pow(2,(_zoom_level -1)))+( Longitude  *((256*pow(2,_zoom_level ))/360)) )
<def_map_size AND max((256*pow(2,(_zoom_level-1)))+((0.5*log((1+(sin((Latitude)*pi()/180)))/(1-
(sin((Latitude)*pi()/180)))))*((-256*pow(2,_zoom_level))/(2*pi()))))-min((256*pow(2,(_zoom_level-
1)))+((0.5*log((1+(sin((Latitude)*pi()/180)))/(1-(sin((Latitude)*pi()/180)))))*((-
256*pow(2,_zoom_level))/(2*pi()))))<def_map_size,_zoom_level,null()),_zoom_level))>def_zoom_level,
max(aggr(if(max( round(256*pow(2,(_zoom_level -1)))+( Longitude  *((256*pow(2,_zoom_level ))/360)) )-min(
round(256*pow(2,(_zoom_level -1)))+( Longitude  *((256*pow(2,_zoom_level ))/360)) ) <def_map_size AND
max((256*pow(2,(_zoom_level-1)))+((0.5*log((1+(sin((Latitude)*pi()/180)))/(1-(sin((Latitude)*pi()/180)))))*((-
256*pow(2,_zoom_level))/(2*pi()))))-min((256*pow(2,(_zoom_level-
1)))+((0.5*log((1+(sin((Latitude)*pi()/180)))/(1-(sin((Latitude)*pi()/180)))))*((-
256*pow(2,_zoom_level))/(2*pi()))))<def_map_size,_zoom_level,null()),_zoom_level)),def_zoom_level)';
var_maptype=    '=if(isnull(only(maptype)),fieldvalue( '&chr(39)&'maptype'&chr(39)&', 4 ),maptype)';

// Field required for calcualting best zoom level
SET HidePrefix='_' ;

_zoom_level:
Load RecNo( ) as _zoom_level autogenerate(max_zoom_level);

maptype:
LOAD * INLINE [
                Maptype
                roadmap
                mobile
                satellite
                terrain
                hybrid
];
```

Geo-code Information

In addition to the script to render the Google Map images above, geo-code information also needs to be loaded into the application. At a minimum, this data must consist of three values:

An Associative Key Field – There must be a field defined to link the geo-code data to the data to be plotted on the Map(s).
Longitude – The longitudinal coordinates.
Latitude – The latitudinal coordinates.

The field should be named "Longitude" and "Latitude" in order to be properly evaluated across the expressions in this document. If this field name is changed, the expressions will need to be changed accordingly. An example of a geo-code data load is below:

```
LOAD [Country Code],
[Geo Post Code],
GeoCodeKey,
City,
State,
AltState,
County,
AltCounty,
Community,
Latitude,
Longitude,
Accuracy,
Coordinates
FROM [Source Data\CoordinatesPostCode.qvd] (qvd)
```

The geo-code information can be gather from any location a user desires. QlikView currently uses data from the following source site:

http://www.geonames.org/postal-codes/

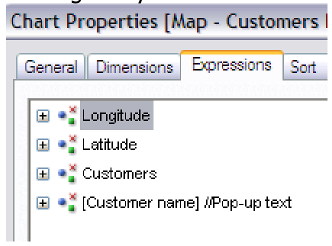
On this site, you can download individual country information based on Postal Codes, or a full data set of countries. Please refer to the website for details.

Chart Creation for Google Maps

Once the proper information is loaded through the QlikView script, you can create the proper charts within your application. A standard scatter chart is used to generate the map image. With the release of QlikView 9.x, a user now has the ability to create a dynamic background for a chart. This new functionality allows for a much easier implementation of a Google Map.

In prior versions of QlikView, this was done using a combination of a scatter chart overlaid on a text object to render the map image. For information on implementing Google Maps in versions prior to QV 9.x, please refer to the Additional Information section of this document. The following information assumes QV 9.x is being used.

1. Create a new chart, choosing a scatter chart as the type.
2. Choose the dimension you want to map. This will be specific to your data, but for example purposes, this document will map Customer Names.
3. Although only the first three expressions below are required, it is recommended to use all four on your map(s).

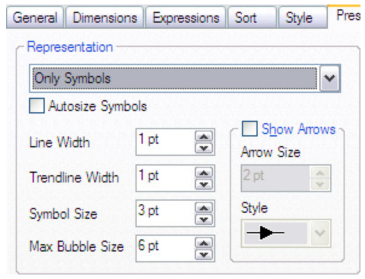


Longitude
Latitude
Customers
[Customer name] //Pop-up text

The expressions are as defined below:

Longitude: `round (256*pow(2,($(var_zoom)-1)))+( Longitude  *((256*pow(2,$(var_zoom)))/360))`
Latitude: `((256*pow(2,($(var_zoom)-1)))+((0.5*log((1+(sin((Latitude)*pi()/180)))/(1-(sin((Latitude)*pi()/180)))))*((-256*pow(2,$(var_zoom)))/(2*pi()))))`
Customers: `Count (distinct([Customer name]))  //This is the bubble size expression`
Customer Name: `[Customer name] //Pop-up text`

4. On the Presentation tab of the chart, you can adjust the size of the bubble symbol as needed:



Only Symbols
Autosize Symbols
Line Width 1 pt
Trendline Width 1 pt
Symbol Size 3 pt
Max Bubble Size 6 pt

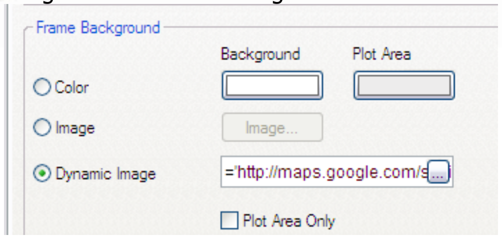
5. To ensure the maps resize appropriately with data selections and zoom levels, the Min and Max values of the Axis endpoints need to be defined as follows:

Min x-axis: `(256*pow(2,((var_zoom)-1)))+((var_mid_long)*((256*pow(2,(var_zoom)))/360))-(def_map_size*0.5)`
Max x-axis: `((256*pow(2,((var_zoom)-1)))+((var_mid_long)*((256*pow(2,(var_zoom)))/360))+(def_map_size*0.5))`

Min y-axis: `((256*pow(2,((var_zoom)-1)))+((0.5*log((1+(sin((var_mid_lat)*pi()/180)))/(1-(sin((var_mid_lat)*pi()/180)))))*((-256*pow(2,(var_zoom)))/(2*pi()))))+(def_map_size*0.5))`
Max y-axis: `((256*pow(2,((var_zoom)-1)))+((0.5*log((1+(sin((var_mid_lat)*pi()/180)))/(1-(sin((var_mid_lat)*pi()/180)))))*((-256*pow(2,(var_zoom)))/(2*pi()))))-(def_map_size*0.5))`

6. Finally, the background of the chart needs to be set to the dynamic URL for the Google Map. The ability to set a dynamic background is a new feature in QV 9.x. On the Colors tab of the Chart Properties, select Dynamic

Image as the Frame Background.


Frame Background

	Background	Plot Area
○ Color		
○ Image	Image...	
◉ Dynamic Image	='http://maps.google.com/s	

☐ Plot Area Only

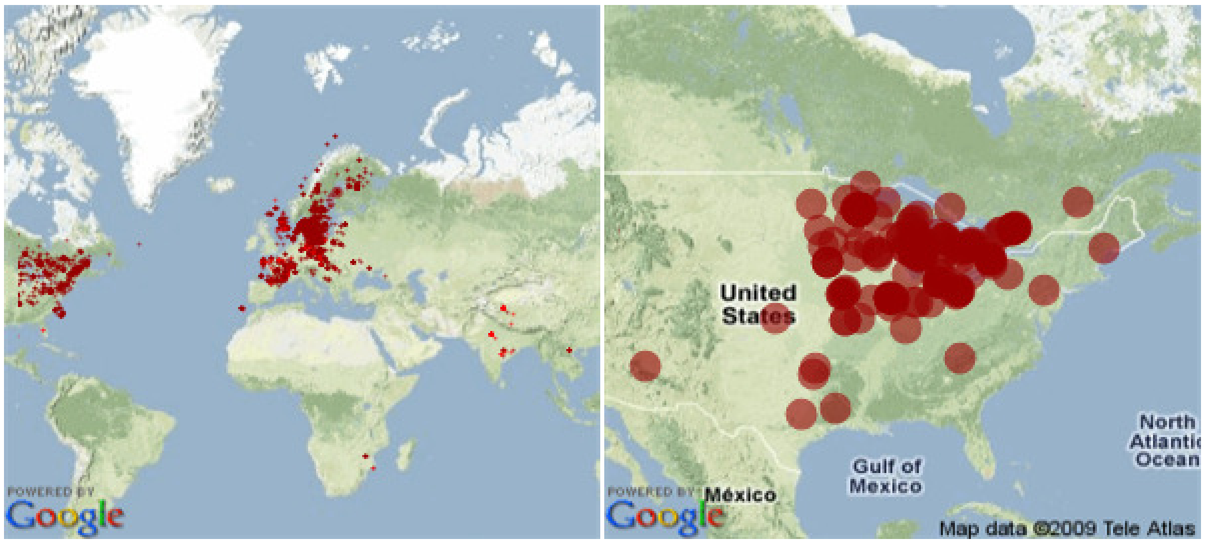
In the expression for the image, the following should be entered:

```
='http://maps.google.com/staticmap?center='&Replace(var_mid_lat,',','.')&','&Replace(var_mid_long,',','.')&
'&zoom=$(var_zoom)'&'&maptype='&var_maptype&'&size='&def_map_size&'x'&def_map_size&'&key='&gmap_key &
'.jpg'
```

This URL will make the proper calls to Google Maps and render the images as necessary based on zoom levels, map types, etc. No changes to this should be required.

7. Finally, set the Width and Height of the chart to be the size of the Google Map image you are rendering. In this case, the map size is 400x400 as defined by the **def_map_size** variable.

After the above steps are implemented, you will then see a rendered map within QlikView that will look similar to the following image. This image is centered on the median values of Longitude and Latitude and will re-center and zoom in as appropriate as filter criteria is applied to the map. The second image shows the same data set filtered on the criteria specific to the United States. As you can see the map has re-centered and zoomed accordingly.



Additional Information

Variable Sized Maps

The above information can be used to implement a single chart utilizing Google Maps. However, there are some simple time saving methods that can be done to make this simpler process to implement over several charts.

It is recommended to create variables for many of the expressions defined. This way, instead of having to enter the long equation for each Min/Max axis value; you can just enter the variable. For example, if we define a variable **vMinXaxis** as:

```
vMinXaxis = (256*pow(2,((var_zoom)-1)))+((var_mid_long)*((256*pow(2,(var_zoom)))/360))-(def_map_size*0.5)
```

We can simply use the $(**vMinXAxis**) variable to define the Static Min for the x-axis on all the maps in the application making it much simpler to setup. Similar variables can be defined for the y-axis as well.

Additionally, the steps defined in this document allow you to build a single scatter chart to render a Google Map. The assumption being that you only want to render one map size. If, however, you want to render multiple maps in different sizes, there are some simple steps that you can do to accommodate this requirement without too much additional work.

The example in this document uses a **def_map_size** of 400. What this does is sets the map image size to a 400x400 image. If a user wants to use a smaller, or larger, map size, this variable just needs to be set to the image size required. The Google Map images are square images, so the setting will determine both the x and y pixel sizes. If a user requires a 300x300 image; this value should be set to 300. Google offers a wide range of image sizes – square values only - anywhere from 100x100 to 800x800 and beyond. A user should experiment with the map sizes to find one that accommodates their particular needs.

Multiple Maps of Difference Sizes

There may be times, however, when a user will want to use maps of multiple sizes. In order to accommodate this need, it may be necessary to create the appropriate expressions for the size of the maps required. The details outlined here are not required to use different map sizes. The need for these additional steps is dependent on the granularity of the accuracy of the mapping features being implemented. The recommendation here is to create multiple variable expressions for each sized map required. As an example, instead of using the **def_map_size** variable, a user would create multiple variables for each specific map size and then use those variables in the definition of the maps. So, as an example, if a user wanted to show maps in 400x400 as well as 300x300, they may do something like the following:

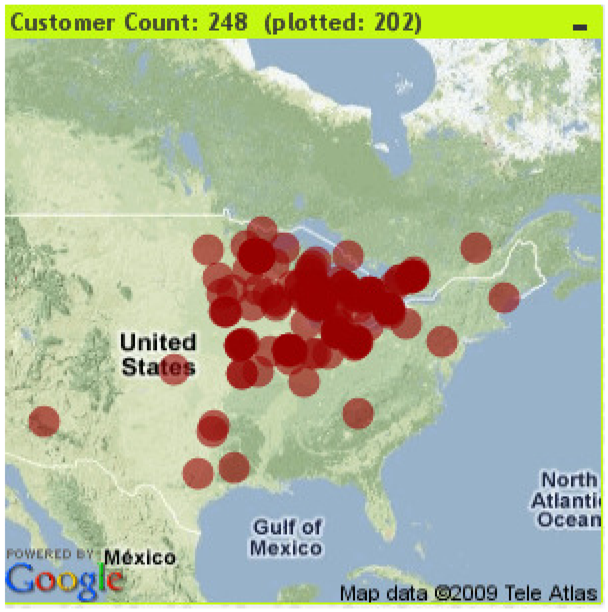
1. Create **map_size_400** and **map_size_300** variables and then duplicate the **var_zoom** expressions to handle each image size accordingly, replacing **def_map_size** with the newly defined variables.
2. Additionally, replace the **def_map_size** variable in the expressions for Min/Max x/y-axis expressions as well.

```
map_size_400 = 400;
map_size_300 = 300;

// Variables required for calculating map
// No need to change these
var_zoom_400=  '=If(max(aggr(if(max( round(256*pow(2,(_zoom_level -1)))+( Longitude  *((256*pow(2,_zoom_level
))/360)) )-min( round(256*pow(2,(_zoom_level -1)))+( Longitude  *((256*pow(2,_zoom_level ))/360)) ) <
map_size_400 AND max((256*pow(2,(_zoom_level-1)))+((0.5*log((1+(sin((Latitude)*pi()/180)))/(1-
(sin((Latitude)*pi()/180)))))*((-256*pow(2,_zoom_level))/(2*pi()))))-min((256*pow(2,(_zoom_level-
1)))+((0.5*log((1+(sin((Latitude)*pi()/180)))/(1-(sin((Latitude)*pi()/180)))))*((-
256*pow(2,_zoom_level))/(2*pi()))))<map_size_400,_zoom_level,null()),_zoom_level))>def_zoom_level,
max(aggr(if(max( round(256*pow(2,(_zoom_level -1)))+( Longitude  *((256*pow(2,_zoom_level ))/360)) )-min(
round(256*pow(2,(_zoom_level -1)))+( Longitude  *((256*pow(2,_zoom_level ))/360)) ) < map_size_400 AND
max((256*pow(2,(_zoom_level-1)))+((0.5*log((1+(sin((Latitude)*pi()/180)))/(1-(sin((Latitude)*pi()/180)))))*((-
256*pow(2,_zoom_level))/(2*pi()))))-min((256*pow(2,(_zoom_level-
1)))+((0.5*log((1+(sin((Latitude)*pi()/180)))/(1-(sin((Latitude)*pi()/180)))))*((-
256*pow(2,_zoom_level))/(2*pi()))))<map_size_400,_zoom_level,null()),_zoom_level)),def_zoom_level)';

var_zoom_300=  '=If(max(aggr(if(max( round(256*pow(2,(_zoom_level -1)))+( Longitude  *((256*pow(2,_zoom_level
))/360)) )-min( round(256*pow(2,(_zoom_level -1)))+( Longitude  *((256*pow(2,_zoom_level ))/360)) ) <
map_size_300 AND max((256*pow(2,(_zoom_level-1)))+((0.5*log((1+(sin((Latitude)*pi()/180)))/(1-
(sin((Latitude)*pi()/180)))))*((-256*pow(2,_zoom_level))/(2*pi()))))-min((256*pow(2,(_zoom_level-
1)))+((0.5*log((1+(sin((Latitude)*pi()/180)))/(1-(sin((Latitude)*pi()/180)))))*((-
256*pow(2,_zoom_level))/(2*pi()))))<map_size_300,_zoom_level,null()),_zoom_level))>def_zoom_level,
max(aggr(if(max( round(256*pow(2,(_zoom_level -1)))+( Longitude  *((256*pow(2,_zoom_level ))/360)) )-min(
round(256*pow(2,(_zoom_level -1)))+( Longitude  *((256*pow(2,_zoom_level ))/360)) ) < map_size_300 AND
max((256*pow(2,(_zoom_level-1)))+((0.5*log((1+(sin((Latitude)*pi()/180)))/(1-(sin((Latitude)*pi()/180)))))*((-
256*pow(2,_zoom_level))/(2*pi()))))-min((256*pow(2,(_zoom_level-
1)))+((0.5*log((1+(sin((Latitude)*pi()/180)))/(1-(sin((Latitude)*pi()/180)))))*((-
256*pow(2,_zoom_level))/(2*pi()))))<map_size_300,_zoom_level,null()),_zoom_level)),def_zoom_level)';
```

Additional Details for QlikView 8.5

Previous version of QlikView can support Google Mapping as well as QlikView 9.00. The main difference is that, in prior versions, there is no feature for a dynamic background in the scatter chart. This can easily be worked around by implementing a two-layer approach with the scatter chart overlaid on a text object that will render the map images.

The expressions used above for the dynamic background is entered into an empty text object. The scatter chart is created exactly as documented above. A user then overlays the scatter chart on top of the text object to render the chart as a map. There is some diligence necessary to line the chart up properly on top of the map image, but once it is set, the feature works as above. It is recommended that, once you have the two-layer map created and working properly, you de-selected the "Allow Move/Size" option from the Layout Properties from both objects to avoid having to realign them. Zoom levels and plotting sequences work as documented. Here is an example of a two-layer map object.



Google Maps Documentation/Terms of Service

For additional information related to Google Maps APIs and Google's Terms of Service, please refer to the Google websites below:

Google Maps API: http://code.google.com/apis/maps/
Terms of Service: http://code.google.com/apis/maps/terms.html