

Simio API Note: Interfacing with GIS Data

Document History

Creation: 15 Feb 2018

Revisions: 14 Mar 2018 – Including extra DLL instructions.

Revisions: 01 Aug 2019 – Fixed and enhanced Scaling login

Revisions: 16 Aug 2020 – Added Google Map Provider, and demonstrate using address files

Contents

Simio API Note: Interfacing with GIS Data	1
Overview	2
Installing the GIS Add-In.....	6
Modifying/Developing the Add-In	6
GIS – Getting the Data from the Web.....	9
Data Provider Examples	10
GIS Service Provider: Microsoft’s Bing Maps (VirtualEarth)	12
Response Information Detail	13
Bing Maps Key.....	15
The Simio Side.....	16
A Note on Coordinates.....	16
Appendix – References	Error! Bookmark not defined.
Bing Maps (VirtualEarth) References.....	20
BingMapsRESTToolkit Samples	22

Overview

GIS (Geographic Information System) data can be an important source of data for Simio models. However, the collection and loading of this data into a simulation can be cumbersome.

This GisAddIn Simio Add-In was constructed to demonstrate how Simio can be extended to work with 3rd party GIS APIs (Application Programming Interfaces) to automatically construct Simio models.

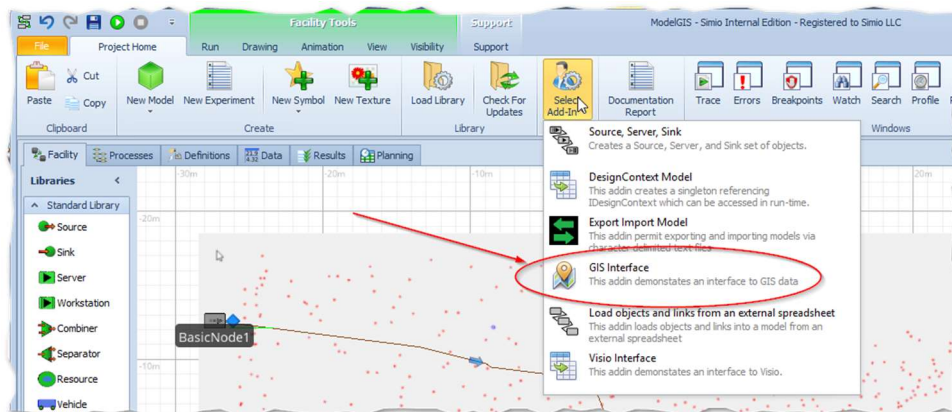
This Simio Add-In interface provides a demonstration of this GIS-to-Simio capability. It can be used to automatically construct simulation paths from routes obtained via a GIS service.

After being launched by Simio, the GisAddIn does this by:

1. Prompting a user for two locations: From and To.
2. Getting GIS information to provide a set of highway directions (a route) using a web-based GIS API.
3. Using GIS-obtained route to construct Simio objects. Specifically, paths between a 'from' and 'to' node.

It is constructed as an Simio "Add-In", which by convention means that it is a DLL that is called by Simio desktop application while in the Simulation Design mode of Simio.

The Add-In is launched from the "Project Home" ribbon by selecting from the "Select Add-In" button, and then choosing GIS Interface.



Once launched, the Add-In brings up its UI, which is a WinForms 'test' screen that allows you to enter the "From" and "To" points and then press a button to request a route from the GIS API. When the GisAddIn form is closed, the construction of the Simio objects takes place.

On the Bing Maps tab are found boxes to enter a “From” and “To”. GIS providers today are very flexible with what you can enter here. For example, you could enter street addresses and a zip code, or just a city with a state (in which case the GIS provider chooses a default location), or even simply a zip code.

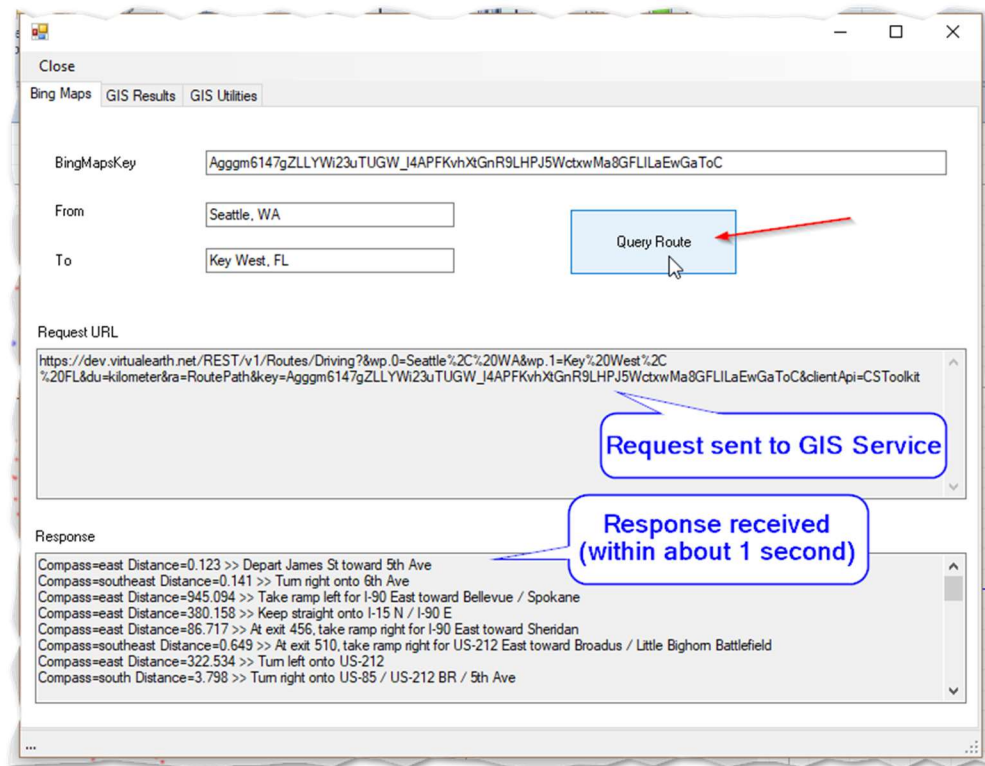


Figure 1 - The GisAddIn User Interface

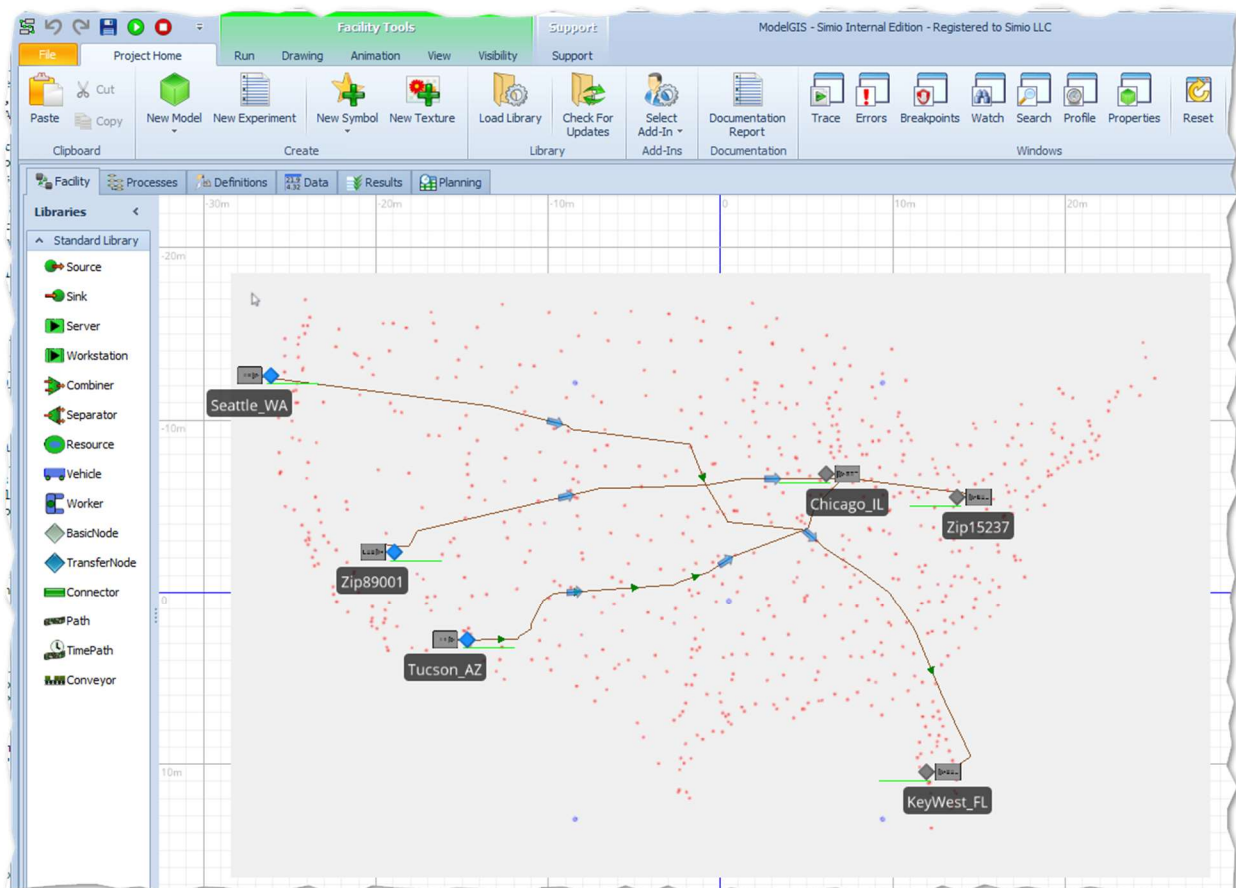
Once the “From” and “To” are chosen, press the button and the GisAddIn will construct a “Route” request query and send it to the GIS Provider. When you are satisfied with your route, close the form and the Add-In will construct the Simio objects in the proper location on the Simio Facility view.

The Simio objects are constructed comprising two “BasicNode” nodes with a Path between them. These nodes are attached to a Simio Source and Sink so that the model is immediately available to run.

Note that for this Add-In, the dimensions were chosen so that a unit of latitude and longitude each correspond to a meter in Simio’s Facility coordinate system. The convention for this demonstration uses a bounding box that is from 20 to 50 North latitude and -60 to -130 longitude, with the lat/lon mid points of -95, 35 being centered at the Simo origin (0,0).

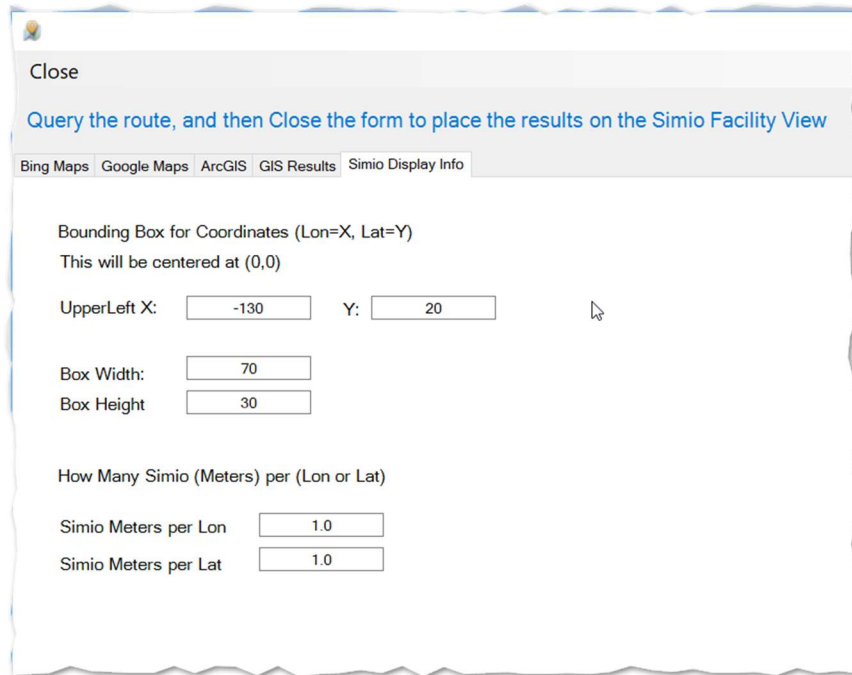
GisAddIn can be invoked repeatedly to add more routes to the model.

Here is a screenshot of a Simio Facility view with three routes added to a Model, and the model running (note the green triangle Entities). Note that the pointillism 'map' is simply an imported symbol showing cities within the contiguous US.



Scaling the Results

A tab name Simio Display Info exists to illustrate how the results can be scaled.



Close

Query the route, and then Close the form to place the results on the Simio Facility View

Bing Maps Google Maps ArcGIS GIS Results **Simio Display Info**

Bounding Box for Coordinates (Lon=X, Lat=Y)
This will be centered at (0,0)

UpperLeft X: Y:

Box Width:
Box Height:

How Many Simio (Meters) per (Lon or Lat)

Simio Meters per Lon
Simio Meters per Lat

The top four boxes allow you to create a bounding box for your entire lat/lon display area. In the example above we are intending to add multiple routes within the USA and so have chosen a box the pretty much encompasses the entire mainland USA. This box will be centered at the origin.

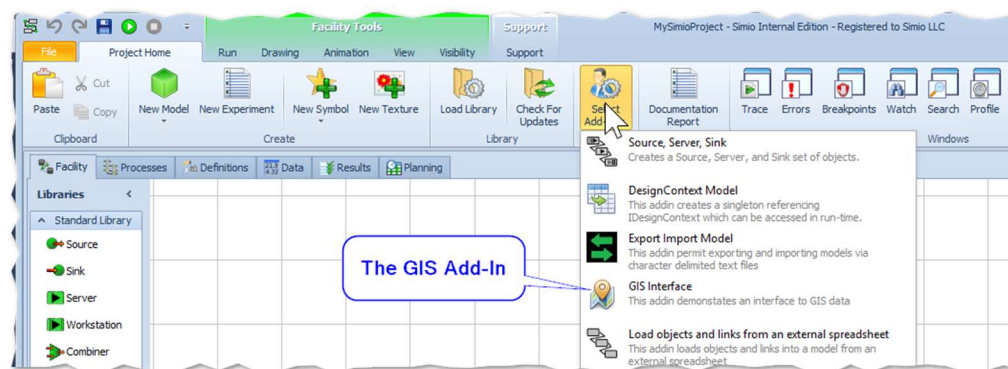
The bottom two boxes show how we want to scale longitude and latitude to the meters on the Simio Facility View.

Installing the GIS Add-In

The GIS Add-In is distributed to the Forum as a zip file, which – besides this documentation - includes:

1. GisAddIn.dll
2. BingMapsRESTToolkit.dll
3. GoogleApi
4. Mathnet.Numerics.dll
5. ModelGIS.spfx

The first three are DLL files that should be placed in a directory together in a standard location so that Simio can find them and place the Add-In in the toolbar when it runs:

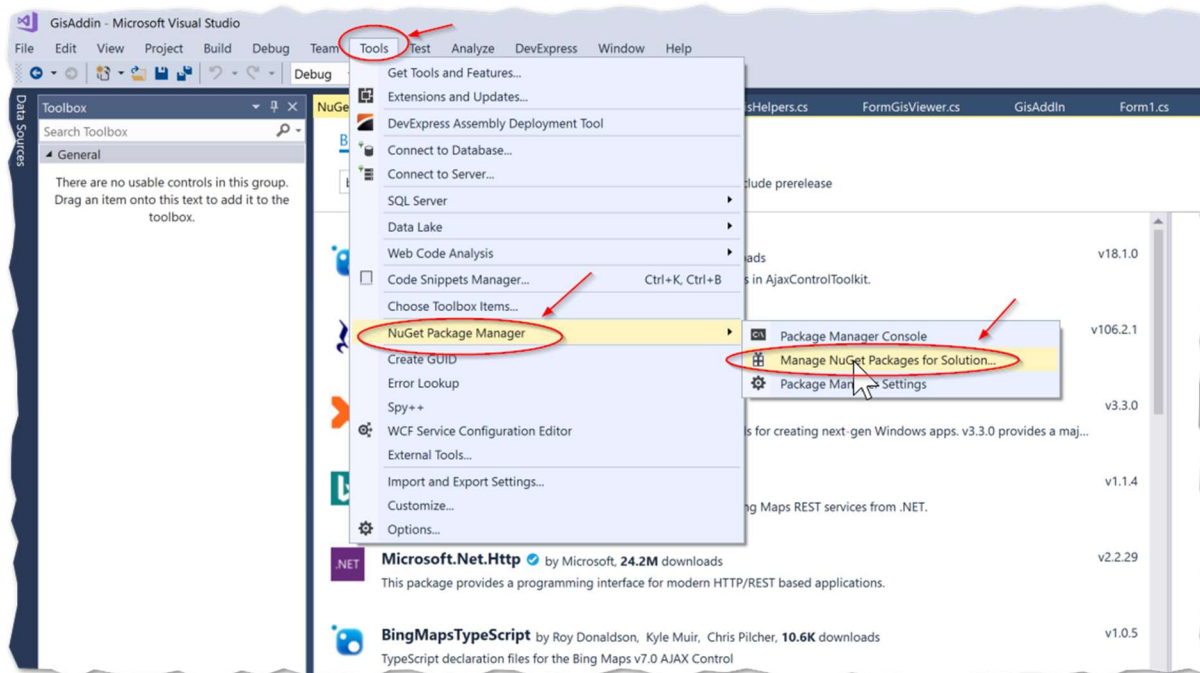


One such standard location is under {your-user}\documents\SimioUserExtensions. Place all the DLL files in this folder.

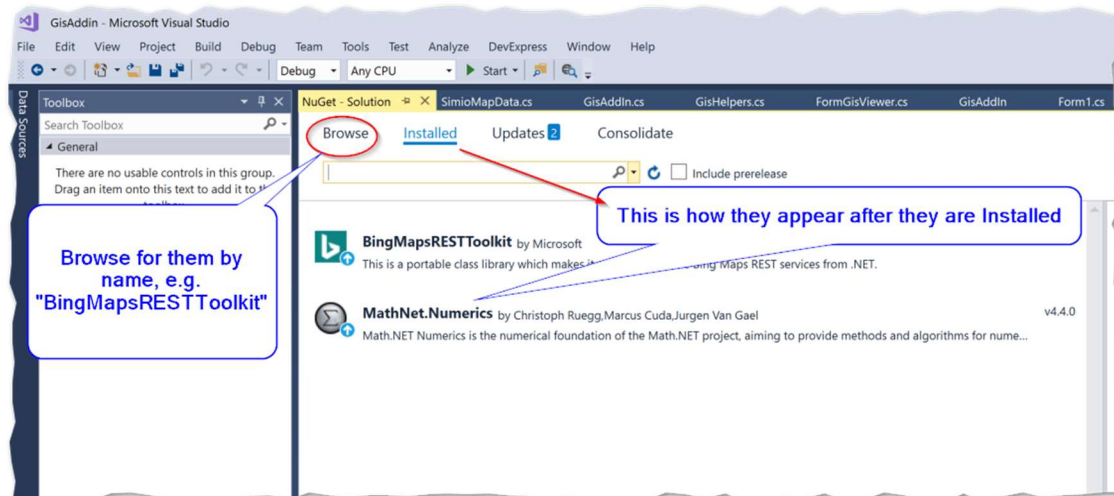
The “.spfx” file is the Simio model, and can be placed wherever you wish. A standard location is {your-user}\documents\SimioModels

Modifying/Developing the Add-In

If you are modifying the Add-In (for example, in Visual Studio) you should get the additional files (the GIS Map SDKs and numeric library using NuGet). This is done by going to Tools > NuGet Package Manager > Manage NuGet Packages for Solution...



And then browsing for the packages as shown. Using NuGet (as opposed to copying DLLs) ensures that you get the most recent versions.



GIS – Getting the Data from the Web

GIS data used in the GisAddIn is gathered from the web with what is called a REST-compliant service on the internet. The REST (Representational State Transfer). What this means is that the calling techniques – regardless of the GIS Service Provider used - will likely be consistent.

How the data is returned will most likely be XML or JSON, but the organization of the data will vary widely depending on the provider. Each provider will likely have their own ‘kit’ or SDK (Software Development Kit) to provide us with the tools to easily implement the communications. But, it will always end up being a Request to the service, following (within a few seconds) by a Response with our data.

Each Provider will require a “key” to make requests of their system. This is because they must maintain an expensive infrastructure to supply this information, and because they wish accountability of the users of their system. Often this key will be free if your volume of requests is below a given threshold.

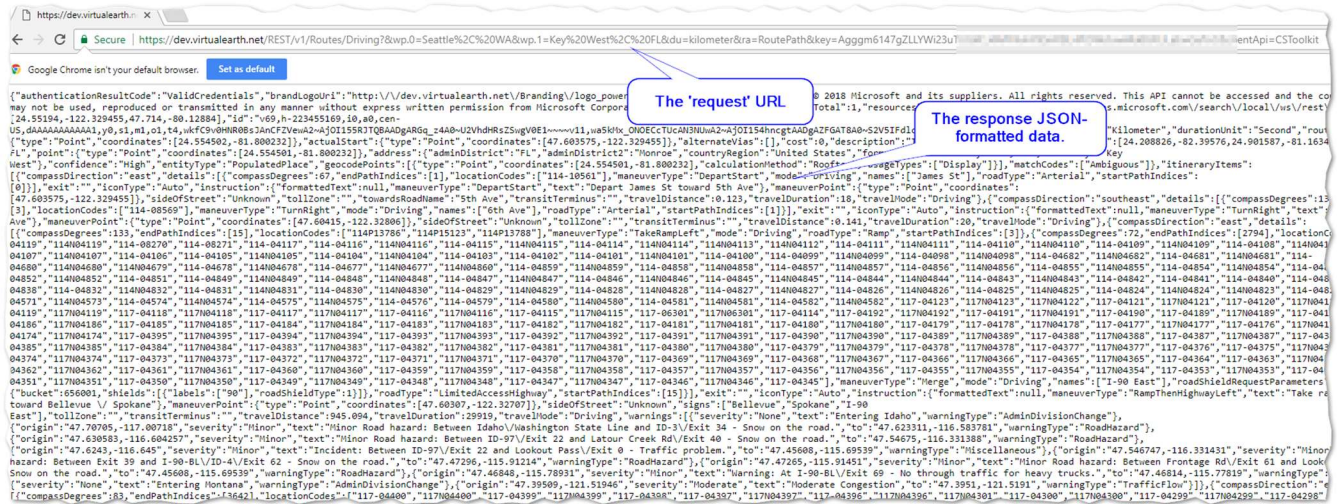
For example, Microsoft, via Bing Maps and their Virtual Earth service, a REST-full API (Application Programming Interface). A ‘free’ key available, so long as you stay below a certain data transaction level.

These are the same services that power most smart-phone, desktop, and web applications, but in developing this Add-In we are going to use the services directly.

In fact, you can copy the contents of the Add-In’s “Request URL” textbox (place your cursor in the Add-In’s Request URL box) and then:

1. CTRL-A to select all,
2. CTRL-C to cut, and then
3. go to your browser’s URL and CTRL-V to paste.

Then hit the ENTER and you’ll see the results:



The GIS data that can be received from these providers is not simply routing. Each provider has a large and rapidly growing set of features.

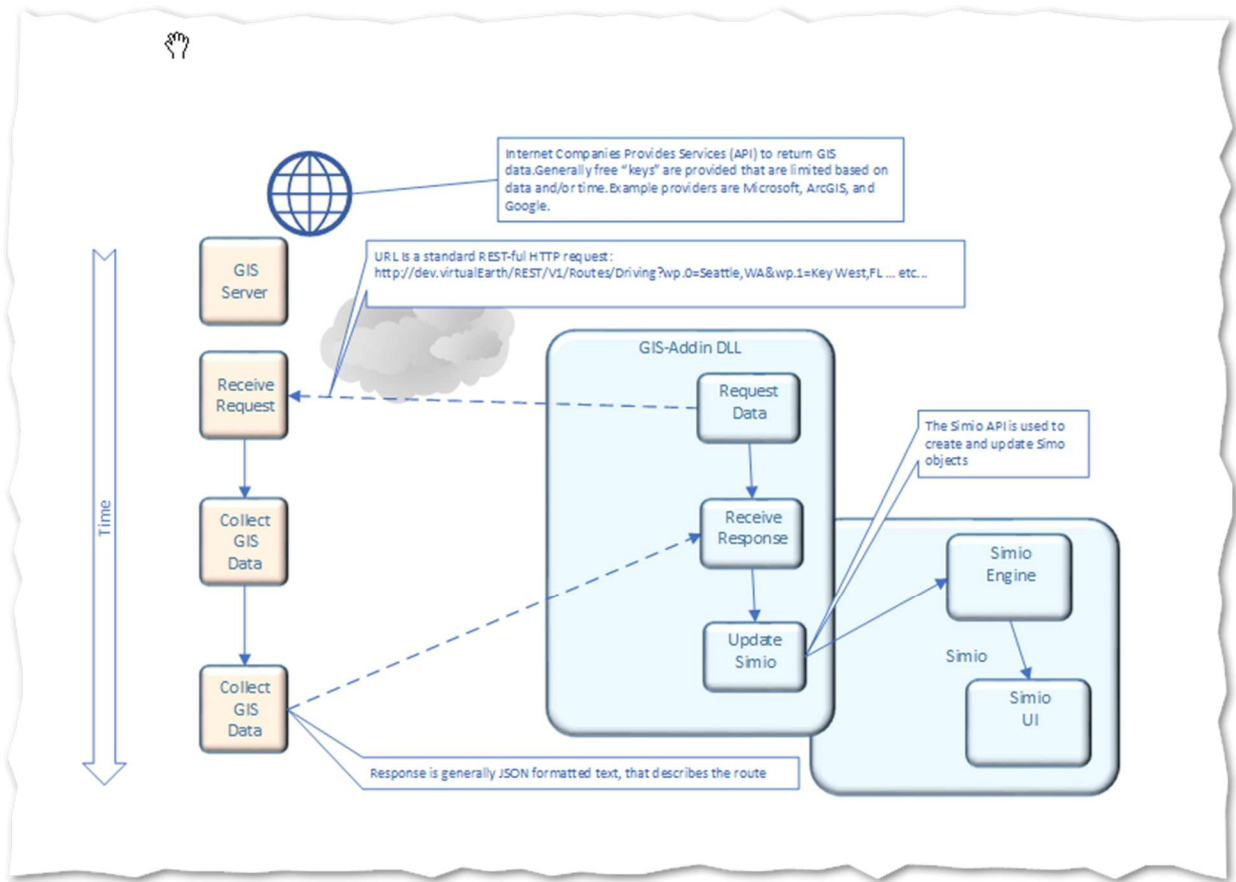
For example, Bing Maps can supply routes for Trucks, where the characteristics of the truck (e.g. weight) are taken into consideration.

Other Responses include distance matrices and even solutions to the Travelling Salesman problem.

Data Provider Examples

This section shows examples of data providers and the Add-In is used to process the data. The first API (Feb 2018) is Bing Maps, with the intent that more will be added as they are encountered.

Most of the GIS APIs work basically in the same fashion, as illustrated by the diagram below:



GIS Service Provider: Microsoft's Bing Maps (VirtualEarth)

Bing Maps provides a GIS data service for getting information.

For example, the following URL:

<http://dev.virtualearth.net/REST/V1/Routes/Driving?wp.0=Pittsburgh,PA&wp.1=Louisville,KY&optmz=distance&routeAttributes=routePath&key=Agggm6147gZMa...aToC>

This my key, but you can get your own for free!

Will return Response GIS data in JSON format:

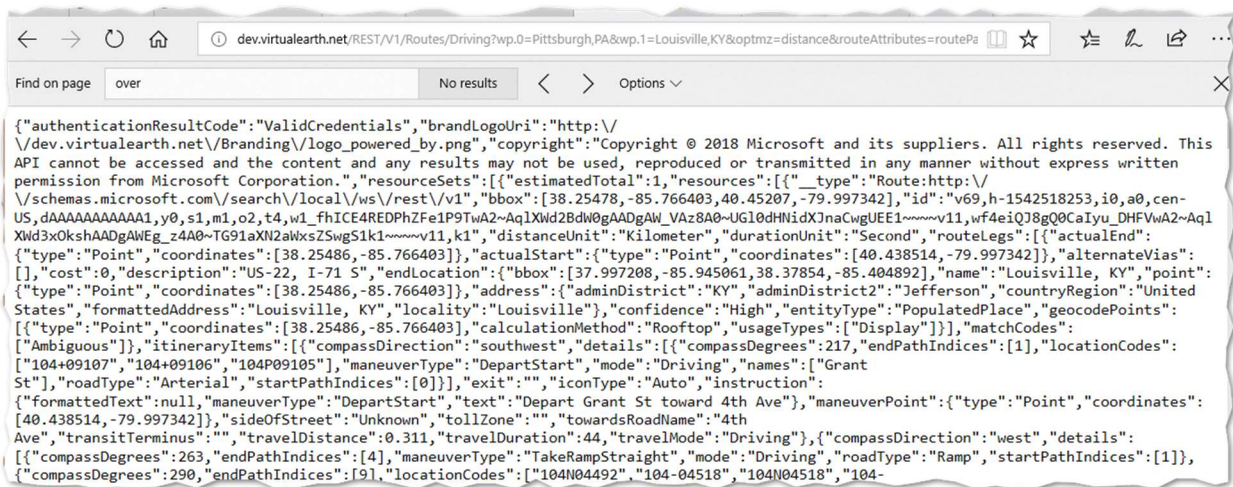
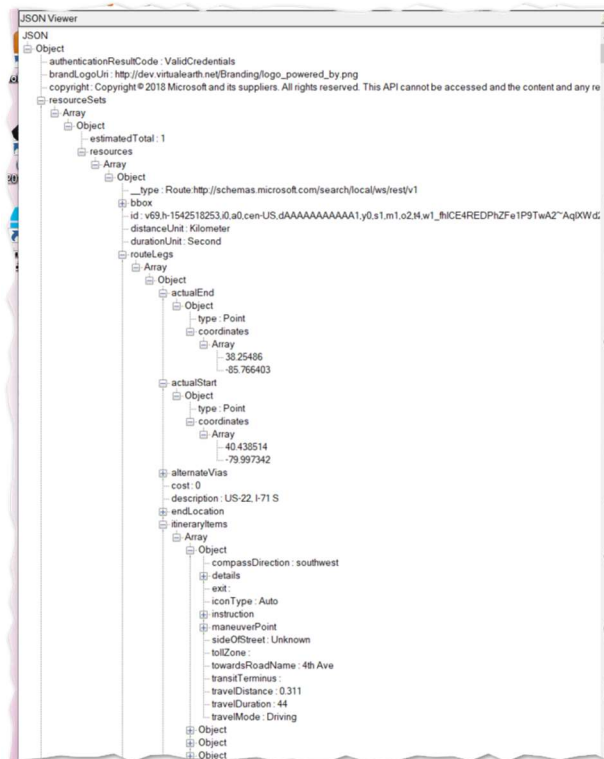


Figure 2 - A Portion of the Response Data in Unformatted JSON

Which can be viewed in Notepad++ with the JSON Viewer plugin:



Response Information Detail

The response information contains all the information about all the paths from here to there. And it is a lot of data and can be controlled by a myriad of options. Refer to the Appendix for more detailed information on this interface.

The add-in processes this information, simplifies it according to the user's wishes, and then creates Simio nodes and links from the data.

From the Tree outline from Notepad++, you can see that file contains:

1. A bounding box, in lat/lon coordinates
2. An array of routeLegs
3. An array of routePaths

The routePath is an array of coordinates. Each adjoining coordinates (lat,lon) are referenced (by index) by a RouteLeg itineraryItems, which are essentially driving instructions.

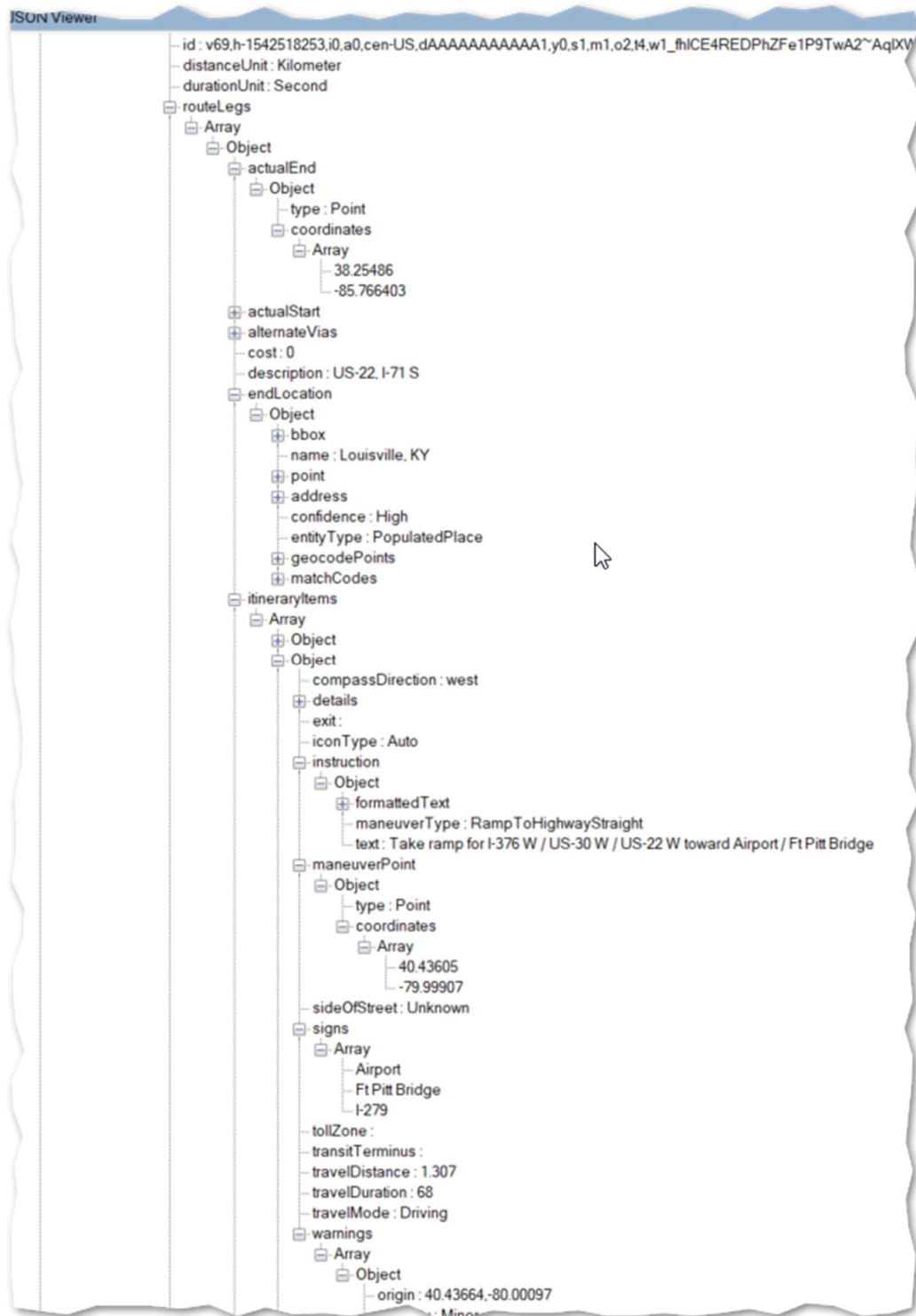


Figure 3 - Response Data Shown in a Tree Structure

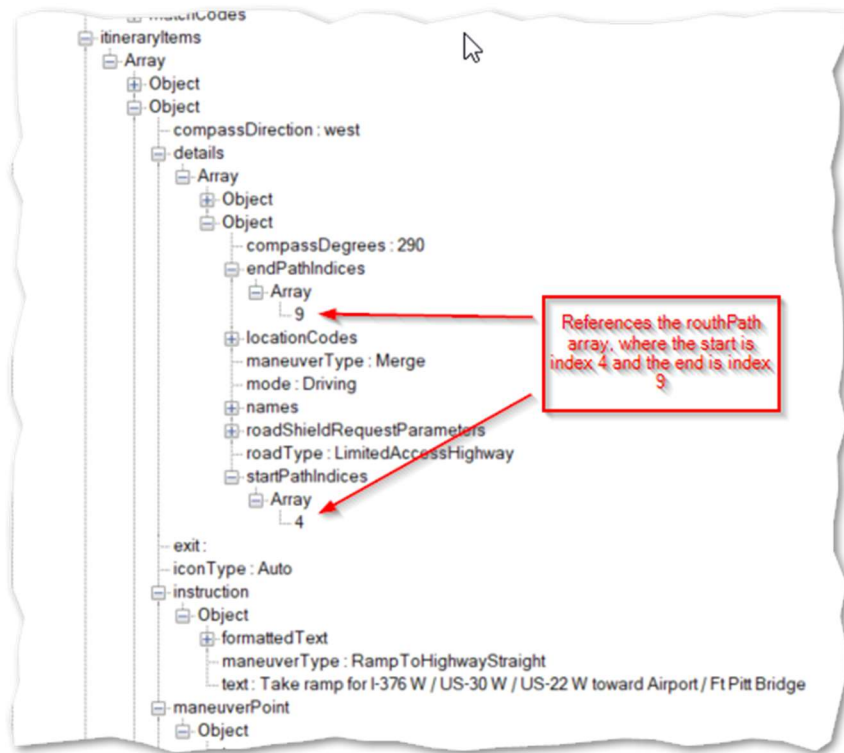


Figure 4 - Response Data Showing Itinerary Details

Data Processing

To help with the processing, the GisAddIn employs the BingMapsRESTToolkit package. This package can be located and installed in Visual Studio by invoking Tools > NuGetPackageManager .

PM>Install-Package BingMapsRESTToolkit

Bing Maps Key

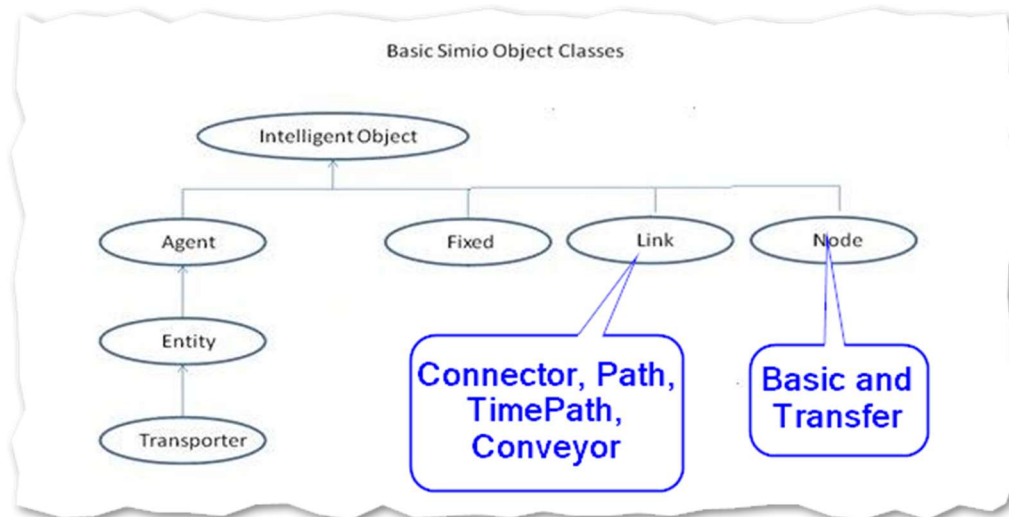
This link:

<https://msdn.microsoft.com/en-us/library/ff428642.aspx>

explains the rules and the procedures for getting a Bing Maps Key.

The Simio Side

Now we can talk about the Simio side of this equation. From the docs, we have these objects:



For this demonstration we are going to create two Basic Nodes (at the beginning and end), and we'll use the Path to connect them.

For this example, the Path Link was used. However, you should consider the TimePath as well, as it provides a way to set the time for the entire route. (This time was included in the Bing Maps response data).

Because each GIS Provider is going to have a different format for the response, a canonical set of data classes is provided in the GisAddIn, and it is these classes that the methods that build the Simio objects from. These can be found in the SimioMap*.cs files.

Methods within the GisHelpers.cs file include helper methods for serializing and deserializing JSON data. There is also a method used to convert lat/lon data to Simio's Facility coordinates.

A Note on Coordinates

Note that by convention most humans would say lat-lon instead of the reverse, however usually the longitude is thought of as the "x" axis. Also remember that Simio is a 3D visualization model. As such, we are generally "looking down" on the X-Z coordinate space, where Z becomes more negative as it goes away from us. So generally we are translating lat/lon to an (X,Y,Z) of something like (LON, 0, -LAT).

Using Files

The tabs for testing Map Providers are examples to show how to query a Map Provider to get routes and then how to transform that data into a Simio facility.

In practice you would want the location data to be stored in files so that you would not have to query the Map Providers each time.

This feature is demonstrated in the Add-In's "Use Address File" tab.

This shows only one of an infinite number of ways that this can be done and is provided as reference material only. In other words, do not simply take this and place it into a production system; understand it, modify it, and make it your own.

The sample files described below are in the GitHub location under the Data folder.

Scenario

Our scenario is this:

We have been provided with a delimited text file that has two addresses: the origin and the destination, and it contains 100 such pairings.

We want to get our map provider to provide routes for each of these pairings and then store all this information in a single file with JSON format.

Once this is done, we can read the route data and produce a Simio facility view without bothering the Map Provider. The major reason for not wanting to re-query the Map Provider is simply a time and money issue.

The Data:

Note: the sample data used here is from the Data folder in our GitHub repository. The CSV file was created from addresses provided by HeroicEric and then randomly paired using the Tools tab of our GisAddin form.

The source data of the address pairs is the SamplePairData.csv file. CSV is a convention for delimited files and stands for "Comma Separated Values", but often – and such is the case here – the delimiter is not a comma. We are using a tab for our delimiter since our address contain commas. Here is a view of the beginning of the file:

```

1 0 8551 Whitfield Ave, Leeds AL 35094 139 Merchant Place, Cobleskill NY 12043
2 1 330 Sutton Rd, Huntsville AL 35763 3300 South Oates Street, Dothan AL 36301
3 2 8064 Brewerton Rd, Cicero NY 13039 3164 Berlin Turnpike, Newington CT 6111
4 3 165 Vaughan Ln, Pell City AL 35125 910 Wolcott St, Waterbury CT 6705
5 4 301 Falls Blvd, Quincy MA 2169 1004 County Road 48, Fairhope AL 36533
6 5 2200 South Mckenzie St, Foley AL 36535 139 Merchant Place, Cobleskill NY 12043
7 6 10675 Hwy 5, Brent AL 35034 3371 S Monroeville AL 36460
8 7 630 Coonial Promenade Pkwy, Alabaster AL 35007 2575 Us Hwy 43, Winfield AL 35594
9 8 66-4 Parkhurst Rd, Chelmsford MA 1824 3222 State Rt 11, Malone NY 12953
10 9 312 Palisades Blvd, Birmingham AL 35209 1470 S Washington St, North Attleboro MA 2760
11 10 630 Coonial Promenade Pkwy, Alabaster AL 35007 3949 Route 31, Clay NY 13041
12 11 200 Dutch Meadows Ln, Glenville NY 12302 10675 Hwy 5, Brent AL 35034
13 12 2780 John Hawkins Pkwy, Hoover AL 35244 11697 US Hwy 431, Guntersville AL 35976
14 13 20 Soojian Dr, Leicester MA 1524 2181 Pelham Pkwy, Pelham AL 35124

```

... and of course, the tab is not shown.

When the “Query and Save Routes” button is pressed, a call to the Map Provider is made for each line. This results in a series of segments with latitude and longitude being provided. This information is stored in a MapRoute object, with all these routes being stored in a parent MapRoutes object.

It is a great convenience that C# and .NET enable us to save this large complex object to a JSON file in a single call. An added benefit (for humans) is that it is stored in a text file and therefore easily readable. Here is the portion of that file:

```
2188 ],
2189 "StartName": "69 Prospect Hill Road, East CT 6088",
2190 "EndName": "30 Memorial Drive, MA 2322"
2191 },
2192 {
2193   "SegmentList": [
2194     {
2195       "Index": 0,
2196       "StartLocation": {
2197         "Lat": 41.280777,
2198         "Lon": -72.831844
2199       },
2200       "EndLocation": {
2201         "Lat": 41.277439,
2202         "Lon": -72.829647
2203       },
2204       "Duration": 53,
2205       "Distance": 0.444
2206     },
2207     {
2208       "Index": 1,
2209       "StartLocation": {
2210         "Lat": 41.277439,
2211         "Lon": -72.829647
2212       },
2213       "EndLocation": {
2214         "Lat": 41.276839,
2215         "Lon": -72.827884
2216       },
2217       "Duration": 52,
2218       "Distance": 0.162
2219     },
2220     {
2221       "Index": 2,
2222       "StartLocation": {
2223         "Lat": 41.276839,
2224         "Lon": -72.827884
2225       },
2226       "EndLocation": {
2227         "Lat": 41.27619,
```

As can be seen, the file can be quite large since all data is stored for all legs of the route. Note also that ancillary data from the Map Providers, such as Duration and Distance can be stored, but that the nature and format of this data would depend upon the Map Provider. For example, Bing stores doubles for this data, while Google stores integers.

Appendix: Bing Maps (VirtualEarth) References

Using the REST Services with .NET

<https://msdn.microsoft.com/en-us/library/jj819168.aspx>

Microsoft has produced an Open Source project that incorporates 'best practices' for using their API and maintains it on GitHub as BingMapsRESTToolkit.

<https://github.com/Microsoft/BingMapsRESTToolkit/>

Here is a sample of the documentation; the The RouteRequest Class:

RouteRequest Class

A request that calculates routes between waypoints. Inherits from the BaseRestRequest class.

Methods

Name	Return Type	Description
Execute()	Task<Response>	Executes the request.
Execute(Action<int> remainingTimeCallback)	Task<Response>	Executes the request.
GetRequestUrl()	string	Gets the request URL to perform a query for route directions.

Properties

Name	Type	Description
RouteOptions	RouteOptions	Options to use when calculate route.
Waypoints	List< SimpleWaypoint >	Specifies two or more locations that define the route and that are in sequential order. A route is defined by a set of waypoints and viaWaypoints (intermediate locations that the route must pass through). You can have a maximum of 25 waypoints, and a maximum of 10 viaWaypoints between each set of waypoints. The start and end points of the route cannot be viaWaypoints.

Extended Properties

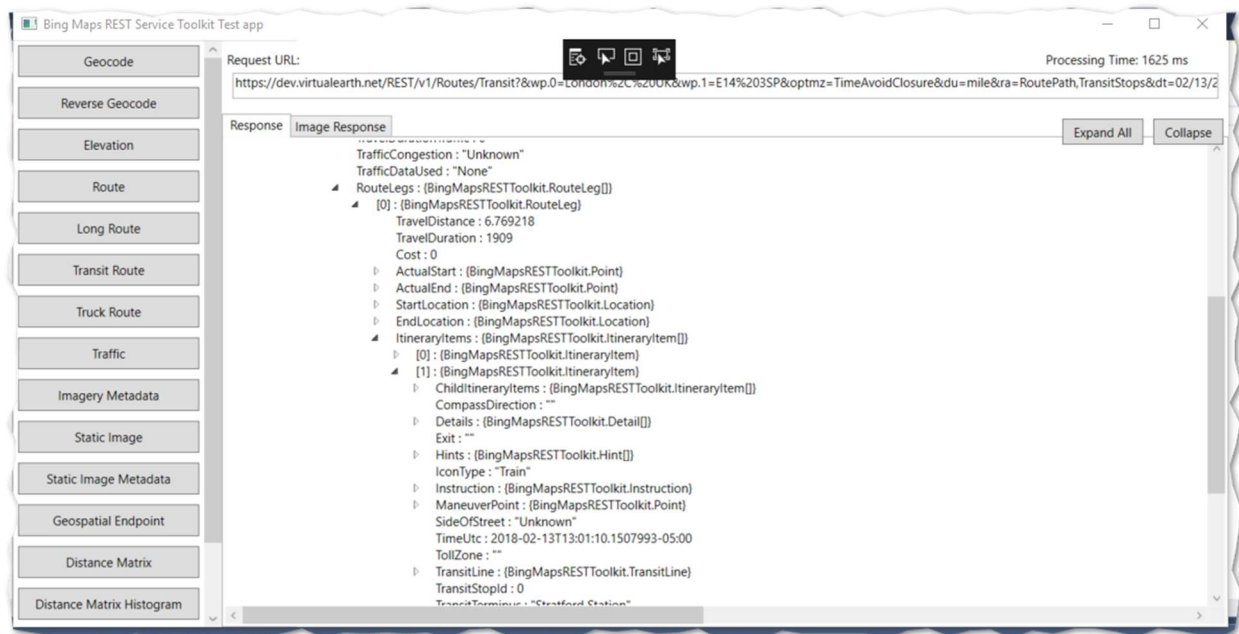
Some additional options have been added to the route request to increase its functionality.

Name	Type	Description
BatchSize	int	The maximum number of waypoints that can be in a single request. If the batchSize is smaller than the number of waypoints, when the request is executed, it will break the request up into multiple requests, thus allowing routes with more than 25 waypoints to be . Must be between 2 and 25. Default: 25.
WaypointOptimization	TspOptimizationType	Specifies if the waypoint order should be optimized using a travelling salesmen algorithm which metric to optimize on. If less than 10 waypoints, brute force is used, for more than 10 waypoints, a genetic algorithm is used. Ignores IsViaPoint on waypoints and makes them waypoints. Default: false Warning: If travel time or travel distance is used, a standard Bing Maps key will need to be required, not a session key, as the distance matrix API will be used to process the waypoints. This can generate a lot of billable transactions.

BingMapsRESTToolkit Samples

There are two sample projects provided; Console and WPF. The WPF is much more full-featured than the console, and is probably your best choice for seeing how the Toolkit operates and for grabbing pieces of code.

WPF Example Screen



Note that there are Long, Transit, and Truck routes as well as plain Route.

Here is a sample of the "Route" Code, showing how to employ Route options:



Forward Thinking

```
Solution | MainWindow.xaml.cs | ObjectNode.cs | MainWindow.xaml
- RESTToolkitTestApp.Mai | RouteBtn_Clicked(object
1 reference
private void RouteBtn_Clicked(object sender, RoutedEventArgs e)
{
    var r = new RouteRequest()
    {
        RouteOptions = new RouteOptions(){
            Avoid = new List<AvoidType>()
            {
                AvoidType.MinimizeTolls
            },
            TravelMode = TravelModeType.Driving,
            DistanceUnits = DistanceUnitType.Miles,
            Heading = 45,
            RouteAttributes = new List<RouteAttributeType>()
            {
                RouteAttributeType.RoutePath
            },
            Optimize = RouteOptimizationType.TimeWithTraffic
        },
        Waypoints = new List<SimpleWaypoint>()
        {
            new SimpleWaypoint(){
                Address = "Seattle, WA"
            },
            new SimpleWaypoint(){
                Address = "Bellevue, WA",
                IsViaPoint = true
            },
            new SimpleWaypoint(){
                Address = "Redmond, WA"
            }
        },
        BingMapsKey = BingMapsKey
    };

    ProcessRequest(r);
}
```

And also some code for handling the ProcessRequest. Note that the method is an “Async” method, and uses the “await” keyword to keep your code from stopping other threads - like the UI – in your application.



Simio

Forward Thinking

```
RESTToolkitTestApp.Mai ProcessRequest(BaseRestRequest)
/// This method has a lot of logic that is specific to the sample.
/// To process a request you can easily just call the Execute method on the request.
/// </summary>
/// <param name="request"></param>
15 references
private async void ProcessRequest(BaseRestRequest request)
{
    try
    {
        RequestProgressBar.Visibility = Visibility.Visible;
        RequestProgressBarText.Text = string.Empty;

        ResultTreeView.ItemsSource = null;

        var start = DateTime.Now;
        //Execute the request.
        var response = await request.Execute((remainingTime) =>
        {
            if (remainingTime > -1)
            {
                _time = TimeSpan.FromSeconds(remainingTime);
                RequestProgressBarText.Text = $"Time remaining {_time} ";
                _timer.Start();
            }
        });

        RequestUrlTbx.Text = request.GetRequestUrl();
        var end = DateTime.Now;
        var processingTime = end - start;
        ProcessingTimeTbx.Text = string.Format(CultureInfo.InvariantCulture,
            "{0:0} ms", processingTime.TotalMilliseconds);

        var nodes = new List<ObjectNode>();
        var tree = await ObjectNode.ParseAsync("result", response);
        nodes.Add(tree);
        ResultTreeView.ItemsSource = nodes;
        ResponseTab.IsSelected = true;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```


Appendix – References

USGS

<https://www.usgs.gov>

Bing Maps REST Services

<https://msdn.microsoft.com/en-us/library/ff701713.aspx>

BingMapsRESTToolkit (Preferred interface, according to Microsoft)

<https://github.com/Microsoft/BingMapsRESTToolkit/>

Google Web Services > Directions API

<https://google-developers.appspot.com/maps/documentation/directions/>

Appendix – Google Maps

The Google API is used. To use the API, you must have an API key, and to get the API key you must:

(from <https://developers.google.com/maps/documentation/javascript/get-api-key>)

You must have at least one API key associated with your project.

To get an API key:

Go to the [Google Cloud Platform Console](#).

Click the project drop-down and select or create the project for which you want to add an API key.

Click the menu button  and select **APIs & Services > Credentials**.

On the **Credentials** page, click **Create credentials > API key**.

The **API key created** dialog displays your newly created API key.

Click **Close**.

The new API key is listed on the **Credentials** page under **API keys**.

(Remember to [restrict the API key](#) before using it in production.)