Tell us about your PDF experience.

Bing Maps Articles

Article • 05/22/2024

> **Note**
>
> **Bing Maps for Enterprise service retirement**
>
> Bing Maps for Enterprise is deprecated and will be retired. Free (Basic) account customers can continue to use Bing Maps for Enterprise services until June 30th, 2025. Enterprise account customers can continue to use Bing Maps for Enterprise services until June 30th, 2028. To avoid service disruptions, all implementations using Bing Maps for Enterprise REST APIs and SDKs will need to be updated to use Azure Maps by the retirement date that applies to your Bing Maps for Enterprise account type.
>
> Azure Maps is Microsoft's next-generation maps and geospatial services for developers. Azure Maps has many of the same features as Bing Maps for Enterprise, and more. To get started with Azure Maps, create a free Azure subscription and an Azure Maps account. For more information about azure Maps, see Azure Maps Documentation. For migration guidance, see Bing Maps Migration Overview.

The following articles provide background information and cross-API information for Bing Maps APIs.

Expand table

Article	Description
Accessing the Bing Maps REST Services using PHP	A step-by-step tutorial that shows how to write a PHP application that uses the Bing Maps REST Services.
Accessing the Bing Spatial Data Services using PHP	A step-by-step tutorial that shows how to write a PHP application that uses the Bing Spatial Data Services.
Bing Maps Tile System	Describes the projection, coordinate systems, and addressing scheme used by the Bing Maps Tile System.
Create a Custom Map URL	Explains how to create a custom URL that displays a map with additional items such as routes, search results, and my places collections.
Custom Map Styles in Bing Maps	Information on the custom map style schema used throughout the Bing Maps platform.

Article	Description
Geocoding Japanese Addresses	Explains Japanese address geocoding and supported features.
Geospatial Endpoint Service	The Geospatial Endpoint Service is a REST service that provides information about Geospatial Platform services for the language and geographical region you specify.
SSL Certificate Validation for Java Applications	Explains how to download and install SSL certificates for Java applications.
Understanding Scale and Resolution	Explains how to determine the scale and resolution of an image.
Open Maps: understanding ODbL	Explains Bing Maps use of Open Data under ODbL license.

Accessing the Bing Maps REST Services using PHP

Article • 05/22/2024

> **Note**
>
> **Bing Maps for Enterprise service retirement**
>
> Bing Maps for Enterprise is deprecated and will be retired. Free (Basic) account customers can continue to use Bing Maps for Enterprise services until June 30th, 2025. Enterprise account customers can continue to use Bing Maps for Enterprise services until June 30th, 2028. To avoid service disruptions, all implementations using Bing Maps for Enterprise REST APIs and SDKs will need to be updated to use Azure Maps by the retirement date that applies to your Bing Maps for Enterprise account type.
>
> Azure Maps is Microsoft's next-generation maps and geospatial services for developers. Azure Maps has many of the same features as Bing Maps for Enterprise, and more. To get started with Azure Maps, create a free Azure subscription and an Azure Maps account. For more information about azure Maps, see Azure Maps Documentation. For migration guidance, see Bing Maps Migration Overview.

This article will describe how to write a PHP application that can interact with Bing Maps REST Services APIs.

Representational State Transfer (REST) is an architecture for distributed systems. It follows a stateless client-server model, meaning that there is no memory (context) of past requests stored on the server between client requests. A RESTful web service is a collection of resources, stored under a central URL, which supports a set of operations all of which can be activated using HTTP methods (POST, GET, etc.).

The Bing Maps REST Services consist of three RESTful APIs, all of which can be accessed using HTTP GET requests that include a URI and parameters indicating what information is to be returned. The three available Bing Maps REST Services APIs are:

1. **Locations** – This API performs geocoding operations, which means you can get the location (as well as other information) about an address or place. The Locations API allows you to find, or geocode, locations by structured address or unstructured query. It also allows you to reverse geocode an address by specifying a point (latitude and longitude).

2. **Imagery** – This API allows you to download a static map, which can be customized to a specific size, style, and zoom level, and can also include one or more pushpins representing different locations.

3. **Routes** – This API allows you to specify two or more waypoints and get a route between those waypoints, including a full itinerary. You can customize routes to include traffic information, provide walking directions (rather than driving), and use custom pushpins to represent itinerary points.

The Locations, Imagery, and Routes APIs can be accessed using specially formatted URLs that contain all parameters required to get the requested information. Both of these APIs will return results in an HTTP message that may contain either JSON or XML.

Working with the Bing Maps REST Services APIs in PHP is extremely straightforward. There are no extensions required, since all requests and responses work through standard HTTP (unlike the SOAP web services, which require a php_soap extension in order to function from PHP). If you request response information in XML format, it can be parsed and manipulated using built-in PHP extensions such as SimpleXML.

This article will describe how to construct PHP pages that interact with all three of the currently available Bing Maps REST Services APIs.

Setting Up Your Environment

Before continuing with this article, you should ensure that you have the correct software installed and environment setup to develop and host PHP pages that will connect to the Bing Maps REST Services APIs.

Required Software

To work with and run the sample applications in this article, you will need a PHP-enabled web server. To develop the samples, we used **WampServer** (http://www.wampserver.com), a Windows-based development environment that includes Apache, PHP, and MySQL Database. It is easy to install and has everything you need to create and host PHP applications quickly and easily. You do not need to install any special extensions in order to work with the Bing Maps REST Services APIs.

Microsoft IIS 6 or 7 can also be used to host PHP applications, and we tested several of the examples on it to confirm that they work. If you are using IIS to host PHP, you may also want to install **FastCGI**, which improves performance of CGI applications in IIS. You can find a very detailed set of instructions for installing and configuring FastCGI and PHP at:

- IIS 6 - https://learn.iis.net/page.aspx/247/using-fastcgi-to-host-php-applications-on-iis-60/
- IIS 7 - https://learn.iis.net/page.aspx/246/using-fastcgi-to-host-php-applications-on-iis-7/

There are many other servers that can be configured to host PHP, and you are free to use any of those instead if you are more familiar with their functionality.

Finally, in terms of a development environment, you can use anything from a text editor such as Windows Notepad to a full-fledged PHP IDE such as PHP Designer to write your code.

PHP Starter Code

This article contains numerous samples of interacting with Bing Maps REST Services APIs using PHP. If you are already familiar with PHP, you may not need any assistance creating and setting up a PHP page in which to include your Bing Maps code. However, if you need assistance getting started, you may want to being with the following simple PHP page, which includes a form that asks for the user's Bing Maps Key, an address to be geocoded, and a zoom level for the map to be returned. Bing Maps Keys are discussed in the Authentication section below.

Listing 1 - PHP starter code for working with Bing Maps REST Services APIs

HTML

```
<html>
  <head>
    <title>Using PHP and Bing Maps REST Services APIs</title>
  </head>
  <body>
    <form action="BingMaps_REST_LocationsSample.php" method="post">
      Bing Maps Key: <input type="text" name="key"
     value="<?php echo (isset($_POST['key'])?$_POST['key']:'') ?>"><br>
        Street Address: <input type="text" name="address"
      value="<?php echo (isset($_POST['address'])?$_POST['address']:'') ?>">
<br>
          City: <input type="text" name="city"
      value="<?php echo (isset($_POST['city'])?$_POST['city']:'') ?>"><br>
            State: <input type="text" name="state"
       value="<?php echo (isset($_POST['state'])?$_POST['state']:'') ?>">
<br>
              Zip Code: <input type="text" name="zipcode"
      value="<?php echo (isset($_POST['zipcode'])?$_POST['zipcode']:'') ?>">
<br>
                <input type="submit" value="Submit">
    </form>
```

```
    <?php
  // Code goes here
    ?>
  </body>
</html>
```

We will use this simple page as a starting point for our Locations and Imagery API examples, building on it as necessary to show different options for searching.

Authentication

All Bing Maps REST Services APIs require authentication from the client each time they are called. In order to authenticate against any of the REST Services APIs, you will need a Bing Maps Key. For information about how to sign up for a Bing Maps Developer Account and get a Bing Maps Key, see Getting a Bing Maps Key.

When you send an HTTP request to one of the REST Services APIs, which we will discuss in the next section of this article, you must include the Bing Maps Key as a parameter. For example, you might send an HTTP request like the one shown in Listing 2.

Listing 2 - Authenticating using a Bing Maps key

```
url
```

```
http://dev.virtualearth.net/REST/v1/Locations/US/NY/10007/New York/291
Broadway?output=xml&key=yourKeyHere
```

In all of the examples in this article, you will see the Bing Maps Key sent to the REST Services APIs in this way.

Working with the Locations API

The Locations API allows you to geocode locations based on an address or query, or reverse geocode addresses based on a specified point. All three of these functions can be accessed simply by sending a specially formatted URI that includes parameters for the function. The Locations API can return results as either JSON or XML. For the purposes of this article, we will get results in XML format because we found it easier to work with using PHP's built in XML classes.

> ⓘ **Note**

You can find the complete sample described in this section in the **Code Samples** section at the end of this article.

Geocoding by Address

To geocode a location by address, you must access the Locations API using a specially formatted URI that includes the street address, city, state, and postal code of the location you are trying to access.

As of this writing, the Locations API supports structured address URIs for the following countries/regions:

- United States (US)
- United Kingdom (GB)
- Canada (CA)
- Germany (DE)
- France (FR)

If you want to geocode addresses in a country/region other than one of the ones mentioned above, you must use an unformatted query string as discussed in the next section of this article.The general URI format for geocoding by address using the Locations API is as follows:

The general URI format for geocoding by address using the Locations API is as follows:

Listing 3 - URI format for geocoding by address

```
url
```

```
http://dev.virtualearth.net/REST/v1/Locations/countryRegion/adminDistrict/postalCode/locality/addressLine?key=yourBingMapsKey
```

There are a few exceptions to this general URL format:

1. In the United States, you can leave out the postal code (zip code) if it is unknown.
2. In any of the supported countries/regions, you can replace an address value (adminDistrict, postalCode, locality, or addressLine) with a dash ("-") character if it is unknown.

Assuming you are using the PHP starter code we included as Listing 1 and have collected a Bing Maps key and address information from the user, you could use the following code to call the Locations API and geocode an address in the United States:

Listing 4 - Geocoding a location by address using REST Services Locations API

PHP

```
// URL of Bing Maps REST Services Locations API
$baseURL = "http://dev.virtualearth.net/REST/v1/Locations";

// Create variables for search parameters (encode all spaces by specifying
'%20' in the URI)
$key = $_POST['key'];
$country = "US";
$addressLine = str_ireplace(" ","%20",$_POST['address']);
$adminDistrict = str_ireplace(" ","%20",$_POST['state']);
$locality = str_ireplace(" ","%20",$_POST['city']);
$postalCode = str_ireplace(" ","%20",$_POST['zipcode']);

// Compose URI for Locations API request
$findURL =
$baseURL."/".$country."/".$adminDistrict."/".$postalCode."/".$locality."/"
 .$addressLine."?output=xml&key=".$key;
```

For each of the parameters accepted from the user, we use the PHP **str_ireplace** method to encode the spaces in the URL with the %20 character so that the URI is correctly formed. We then append all of the parameters together in a URI string, which will look something like this:

Listing 5 - REST Services Locations sample URI

url

```
http://dev.virtualearth.net/REST/v1/Locations/US/ny/10007/new%20york/291%20b
roadway?output=xml&key=yourBingMapsKey
```

Notice that the URI indicates output=xml. We include this parameter so that the response data is returned as XML instead of JSON, which is the default. If you obtain JSON format result data, you will need to either manually parse the JSON string using PHP string functions, or locate a JSON extension for PHP.

In order to easily extract data from the response, we can first use the PHP method **file_get_contents** to send the request and save the result to a string, and then create a **SimpleXMLElement** object using that string. **SimpleXMLElement** is part of the XML API included with PHP, and allows you to access XML elements and attributes using PHP syntax.

Once we have a **SimpleXMLElement** containing the response data, we can extract information from it by navigating through its structure. Listing 6 shows the code for obtaining the XML result data and pulling out the latitude and longitude coordinates of the geocoded address.

Listing 6 - Working with response data from the Locations API

PHP

```
// get the response from the Locations API and store it in a string
$output = file_get_contents($findURL);

// create an XML element based on the XML string
$response = new SimpleXMLElement($output);

// Extract data (e.g. latitude and longitude) from the results
$latitude =
 $response->ResourceSets->ResourceSet->Resources->Location->Point->Latitude;
$longitude =
 $response->ResourceSets->ResourceSet->Resources->Location->Point-
>Longitude;
```

The response from a geocode operation may, of course, include multiple results, in which case **ResourceSet** would be an indexed array. **ResourceSets** includes an element, **EstimatedTotal**, which you can use to easily determine the number of results. Each Resource, in addition to point location information, also contains a **BoundingBox** element that indicates northeast and southwest points for use in determining an optimal map view for the results, as well as additional EntityType elements that contain further information (usually a fully formatted Address) about the location that was geocoded. You can find complete documentation for the information returned from the Locations API at Location Data.

Geocoding by Query

If you don't want to provide all of the information required by the Address method, or you want to geocode a location in a country/region that does not have a supported URL, you can perform a geocode operation using a query instead. A query is an unstructured string, which might be something like "119 Spadina Ave, Toronto, Ontario".

The URI format for accessing the Locations API using a query is as follows:

Listing 7 - URI format for geocoding by query

url

```
http://dev.virtualearth.net/REST/v1/Locations/locationQuery?key=
{BingMapsKey}
```

Assuming you obtained a query string from the user (using a similar form field as the ones shown in Listing 1) you could construct a Locations API query request in PHP as shown in Listing 8.

Listing 8 - Geocoding a location by query using the REST Services Locations API

PHP

```
// Store the query in a PHP variable (assuming you obtained it from the
form)
$query = str_ireplace(" ","%20",$_POST['query']);

// Construct the final Locations API URI
$findURL = $baseURL."/".$query."?output=xml&key=".$key;

// get the response from the Locations API and store it in a string
$output = file_get_contents($findURL);

// create an XML element based on the XML string
$response = new SimpleXMLElement($output);

// Extract data (e.g. latitude and longitude) from the results
$latitude =
$response->ResourceSets->ResourceSet->Resources->Location->Point->Latitude;
$longitude =
$response->ResourceSets->ResourceSet->Resources->Location->Point->Longitude;
```

Notice that the code is almost identical to that used for accessing the Locations API using an address; the only difference is the structure of the URI.

Reverse Geocoding

You can use the Locations API to reverse geocode an address from a point as well. To reverse geocode, you provide a latitude and longitude representing a location on the map, and the Locations API will return location information (including formatted address) of whatever is at that location (street address or place). You will get an empty response (EstimatedTotal = 0) if Bing Maps is unable to find an address that matches the location you specified.

The URI format for a reverse geocode request is as follows:

Listing 9 - URI format for reverse geocoding by point

```
url
```

```
http://dev.virtualearth.net/REST/v1/Locations/point?key={BingMapsKey}
```

The *point* must be specified as a pair of latitude and longitude coordinates, separated by a comma. You could use the following PHP code to perform a reverse geocode request (we are hardcoding the point here, but generally this would come from a data source of some kind):

Listing 10 - Reverse geocoding using the Locations API

```
PHP
```

```
$point = "43.64,-79.39";
$revGeocodeURL = $baseURL."/".$point."?output=xml&key=".$key;

$rgOutput = file_get_contents($revGeocodeURL);
$rgResponse = new SimpleXMLElement($rgOutput);

$address = $rgResponse->ResourceSets->
 ResourceSet->Resources->Location->Address->FormattedAddress;
```

Note that the FormattedAddress element combines the street address, city, state, country/region, and postal code.

You can find complete information on the Locations API and its operations at Locations.

Working with the Imagery API

The Bing Maps REST Services Imagery API allows you to request a customized static map image based on information you specify in a URI. You can also request imagery metadata and map tile URLs. As with the Locations API, request URIs for the Imagery API must include parameters. For example, to request a static map, you must specify the center point of the map and/or any pushpins you want to appear on it, as well as other options such as map style, size, and zoom level.

The Imagery API actually allows you to request maps based on three main types of input:

- You can request a map by specifying a center point for the map (as a latitude/longitude pair);
- You can request a map using an unstructured query, in which case you will get a map that most closely matches the query (e.g. '119 Spadina Avenue, Toronto, ON');

or

- You can request a map that shows a specified map area (provided as a bounding box – two sets of coordinates representing the northeast and southwest corners of the area).

In all three of these cases, you can also provide one or more pushpin locations to be displayed on the map, as well as additional options.

We will focus on the first option (requesting a map by a center point) in this article; you can find the URI format and information about parameters for all three options at Imagery.

> ⓘ **Note**
>
> You can find the complete sample described in this section in the **Code Samples** section at the end of this article.

Generating a Map using the Imagery API

We will begin this example by assuming that you have used the Locations API to geocode an address, as described in the previous section of this article. You will need to obtain the latitude and longitude coordinates of the geocoded location, as demonstrated in the previous section of this article.

The general URI format for getting an image by specifying a center point is:

Listing 11 - General URI format for requesting a map from the Imagery API, using a center point

```
url
```

```
http://dev.virtualearth.net/REST/v1/Imagery/Map/imagerySet/centerPoint/zoomLevel=zoomLevel&mapSize=mapSize&pushpin=pushpin&mapLayer=mapLayer&key={BingMapsKey}
```

A brief description of the parameters for this URI follows:

- **imagerySet** – This part of the URL must be one of **Aerial**, **AerialWithLabels**, or **Road**. It indicates the style of map image that will be returned.
- **centerPoint** – A pair of coordinates representing the center of the map to be generated. Typically, you will obtain these coordinates from a data source or by

geocoding an address or place using the Locations API. Example: `45.8237,-120.1933`

- **zoomLevel** – An integer from 1-19 representing the zoom level for the generated map. This is required if generating a map by center point.
- **mapSize**– The width and height (separated by a comma) in pixels for the map to be generated.
- **mapLayer** – This optional parameter allows you to specify additional layers to appear on top of the map. Currently the only option available is **TrafficFlow**.
- **pushpin** – This parameter allows you to specify one or more pushpins to appear on the generated map. You can include up to 18 sets of pushpin parameters URI. A pushpin must include a latitude and longitude at which the pushpin will be placed, and can also include an iconStyle (see Pushpin Syntax and Icon Styles for a list of icon styles) and a label (two-character maximum). The format for a pushpin property is: `pushpin=latitude,longitude;iconStyle;label`. Example: `pushpin=47.620548,-122.34874;5;P1`

Assuming we already have a latitude and longitude available, the PHP code for generating a URI to access the Imagery API might look something like this:

Listing 12 - Requesting a map image from the Imagery API

PHP

```
// Save the base URL for the Imagery API to a string
$imageryBaseURL = "http://dev.virtualearth.net/REST/v1/Imagery/Map";

// Setup the parameters for the Imagery API request (using a center point)
$imagerySet = "Road";
$centerPoint = $latitude.",".$longitude;
$pushpin = $centerPoint.";4;ID";
$zoomLevel = "15";

// Display the image in the browser
echo "<img src='".$imageryURL =
$imageryBaseURL."/".$imagerySet."/".$centerPoint."/".$zoomLevel."?
pushpin=".$pushpin."&key=".$key."'>";
```

Notice that each time this page is loaded, a new image is requested from the Imagery API. The advantage of this is that the image will be updated if map data changes (e.g. streets are added/changed/removed). The disadvantage is that if the REST APIs are unavailable or the access point changes, the image will no longer load. In a real world scenario, you would likely want to cache a backup image in case of API unavailability.

The code in Listing 12 will produce a map image like figure 1 (assuming a geocoded address at 291 Broadway, New York, NY).

Figure1 - Results of retrieving an image using the Imagery API

Working with the Routes API

The Bing Maps Routes API is a REST web service that allows you to get routing information for two or more locations. As with the SOAP Routing service, the Routes API can provide routing information for both driving and walking routes, and is able to include additional layers such as traffic information in the response data.

A call to the Routes API requires a URI that specifies at least two, and up to twenty-five, waypoints. Waypoints can be identified by latitude and longitude coordinates, landmark names, or addresses. Waypoints are numbered (starting at 0) and the route is generated by using the waypoints in sequential order.

Calls to the Routes API can also include the following optional parameters:

- **optimize** – Optimize the route by **distance**, **time**, or **timeWithTraffic**. The default is **time**.
- **routePathOutput** – Set to **Points** to return a list of points along the route's path, for use in rendering the route on a map. The default is **None**.
- **distanceUnit** – Get distances in miles (**mi**) or kilometers (**km**). The default is **km**.
- **travelMode** – Get directions for **Driving** or **Walking**. The default is **Driving**.

> ⓘ Note
>
> You can find the complete sample described in this section in the **Code Samples** section at the end of this article.

Getting a Route using the Routes API

The URI format for getting a route from the Routes API is as follows:

Listing 13 - General URI format for accessing the Routes API

url

```
http://dev.virtualearth.net/REST/v1/Routes?
wayPoint.1=wayPoint1&waypoint.2=wayPoint2&wayPoint.n=wayPointn&optimize=opti
mize&routePathOutput=routePathOutput&distanceUnit=distanceUnit&key=
{BingMapsKey}
```

For this example, we will obtain a route from the Routes API that includes only two waypoints: an origin and a destination. We will specify both of these points as addresses, which we can obtain from the user using a simple form like the one shown in Listing 14.

Listing 14 - A simple PHP form that obtains a Bing Maps Key, origin address, and destination address from the user

HTML

```
<html>
  <head>
    <title>Using PHP and Bing Maps REST Services Routes API</title>
  </head>
  <body>
    <form action="BingMaps_REST_RoutingSample.php" method="post">
      Bing Maps Key: <input type="text" name="key" value=""<?php echo
(isset($_POST['key'])?$_POST['key']:'') ?>"><br>
        Origin: <input type="text" name="origin" value=""<?php echo
(isset($_POST['origin'])?$_POST['origin']:'') ?>"><br>
          Destination: <input type="text" name="destination" value=""<?php
echo (isset($_POST['destination'])?$_POST['destination']:'') ?>"><br>
            <input type="submit" value="Submit">
  </form>
    <?php
  // PHP code goes here
  ?>
  </body>
</html>
```

Once we have obtained the origin and destination addresses from the user, we can use those addresses to construct the URI required to call the Routes API. Listing 15 shows PHP code that constructs a Routes API call that:

- Provides driving directions
- Optimizes the route for time
- Provides distance units in km
- Returns a set of points representing the route

Listing 15 - Constructing a Routes API URI

PHP

```
// URL of Bing Maps REST Services Routes API;
$baseURL = "http://dev.virtualearth.net/REST/v1/Routes";

// Get the Bing Maps key from the user
$key = $_POST['key'];

// construct parameter variables for Routes call
$wayPoint0 = str_ireplace(" ","%20",$_POST['origin']);
$wayPoint1 = str_ireplace(" ","%20",$_POST['destination']);
$optimize = "time";
$routePathOutput = "Points";
$distanceUnit = "km";
$travelMode = "Driving";

// Construct final URL for call to Routes API
$routesURL =
  $baseURL."/".$travelMode."?wp.0=".$wayPoint0."&wp.1=".$wayPoint1
   ."&optimize=".$optimize."&routePathOutput=".$routePathOutput
   ."&distanceUnit=".$distanceUnit."&output=xml&key=".$key;
```

Note that the set of points returned when you specify **routePathOutput=points** is different (and in addition to) the maneuver points that are used to provide driving directions along the route. Whereas maneuver points represent only places along a route where the driver or walker must take action (e.g., take a turn), the **points** array includes all points required to draw the line representing the route on a map (including all curves in roads and other geographic features).

As we did in REF _Ref263769059 \h Working with the Locations API earlier in this article, we can use the **file_get_contents()** method and PHP XML library to call the Routes API using the URI we constructed and parse the return XML into an object that we can then manipulate in code. Listing 16 shows code that creates this XML object and, as an

example, obtains and prints the number of routes returned in the response from the object.

Listing 16 - Getting a response from the Routes API and working with the response as an XML object

PHP

```
// Get output from Routes API and convert to XML element using php_xml
$output = file_get_contents($routesURL);
$response = new SimpleXMLElement($output);

// Extract and print number of routes from response
$numRoutes = $response->ResourceSets->ResourceSet->EstimatedTotal;
echo "Number of routes found: ".$numRoutes."<br>";
```

Assuming the call produced a successful route, we can get the route instructions by looping through the itinerary items and pulling out the instruction text, as shown in Listing 17.

Listing 17 - Extracting and printing route instructions from Routes API response

PHP

```
// Extract and print route instructions from response
$itinerary =
  $response->ResourceSets->ResourceSet->Resources->Route->RouteLeg-
>ItineraryItem;

echo "<ol>";
for ($i = 0; $i < count($itinerary); $i++) {
  $instruction = $itinerary[$i]->Instruction;
  echo "<li>".$instruction."</li>";
}
echo "</ol>";
```

The code in Listing 17 will produce output something like figure 2.

Figure 2 - Displaying route information from a call to the Routes API

1. Depart Reade St toward Church St
2. Turn right onto Church St
3. Bear left onto 6th Ave / Avenue of the Americas
4. Bear right onto W Broadway
5. Turn right onto W Houston St
6. Arrive at 599 Broadway, New York, NY 10012-3235 on the right

You can find a complete description of the Routes API and its options at Routes.

Conclusions and Further Reading

This article has demonstrated how to use PHP to access the three Bing Maps REST Services APIs. We have shown how you can geocode and reverse geocode locations using the Locations API, generate customized static maps using the Imagery API, and create routes using the Routes API.

You can find full API documentation for the Bing Maps REST Services at: Bing Maps REST Services

Code Samples

The following sections provide the complete code samples described in this article.

Locations and Imagery Code Sample

Save this sample to a file named BingMaps_REST_LocationsSample.php. This sample geocodes specific address information or a query using the Locations API and then displays a static map using the Imagery API.

HTML

```
<html>
  <head>
    <title>Using PHP and Bing Maps REST Services APIs</title>
  </head>
  <body>
    <form action="BingMaps_REST_LocationsSampleOrig.php" method="post">
      Bing Maps Key: <input type="text" name="key" value="<?php echo
(isset($_POST['key'])?$_POST['key']:'') ?>"><br>
        Street Address: <input type="text" name="address" value="<?php echo
(isset($_POST['address'])?$_POST['address']:'') ?>"><br>
          City: <input type="text" name="city" value="<?php echo
(isset($_POST['city'])?$_POST['city']:'') ?>"><br>
            State: <input type="text" name="state" value="<?php echo
(isset($_POST['state'])?$_POST['state']:'') ?>"><br>
              Zip Code: <input type="text" name="zipcode" value="<?php echo
(isset($_POST['zipcode'])?$_POST['zipcode']:'') ?>"><br>
                Query: <input type="text" name="query" value="<?php echo
(isset($_POST['query'])?$_POST['query']:'') ?>"><br>
                  <input type="submit" value="Submit">
  </form>
<?php

if(isset($_POST['key']) &&
     ( isset($_POST['address']) || isset($_POST['city']) ||
isset($_POST['state']) || isset($_POST['zipcode']) || isset($_POST['query'])
))
```

```
  {
  // URL of Bing Maps REST Locations API;
    $baseURL = "http://dev.virtualearth.net/REST/v1/Locations";

  // Set key based on user input
  $key = $_POST['key'];

  if ($_POST['query']!= "")//if query value is provided, find location using
query
  {
   // Create URL to find a location by query
    $query = str_ireplace(" ","%20",$_POST['query']);
    $findURL = $baseURL."/".$query."?output=xml&key=".$key;
  }
  else //if query value is not provided, find location using specified US
address values
    // Create a URL to find a location by address
  {
    $country = "US";
    $addressLine = str_ireplace(" ","%20",$_POST['address']);
    $adminDistrict = str_ireplace(" ","%20",$_POST['state']);
    $locality = str_ireplace(" ","%20",$_POST['city']);
    $postalCode = str_ireplace(" ","%20",$_POST['zipcode']);
    // Construct final URL for call to Locations API
    $findURL =
$baseURL."/".$country."/".$adminDistrict."/".$postalCode."/".$locality."/".$
addressLine."?output=xml&key=".$key;
  }

   // Get output from URL and convert to XML element using php_xml

   $output = file_get_contents($findURL)
   $response = new SimpleXMLElement($output);

  // Extract and pring latitude and longitude coordinates from results
  $latitude = $response->ResourceSets->ResourceSet->Resources->Location-
>Point->Latitude;
  $longitude = $response->ResourceSets->ResourceSet->Resources->Location-
>Point->Longitude;

  echo "Latitude: ".$latitude."<br>";
  echo "Longitude: ".$longitude."<br>";

  // Display the location on a map using the Imagery API
  $imageryBaseURL = "http://dev.virtualearth.net/REST/v1/Imagery/Map";

  $imagerySet = "Road";
  $centerPoint = $latitude.",".$longitude;
  $pushpin = $centerPoint.";4;ID";
  $zoomLevel = "15";

  echo "<img src='".$imageryURL =
$imageryBaseURL."/".$imagerySet."/".$centerPoint."/".$zoomLevel."?
pushpin=".$pushpin."&key=".$key."'>";
```

```
}
else
{
  echo "<p>Please enter your Bing Maps key and complete all address fields
for a US address or the Query field, then click submit.</p>";
}
?>
</body>
</html>
```

Locations Reverse Geocode Sample

Save this sample to a file named BingMaps_REST_ReverseGeocodeSample.php. This sample shows how to use the Locations API to reverse-geocode a pair of latitude and longitude values.

HTML

```
<html>
  <head>
    <title>Using PHP and Bing Maps REST Services Locations API</title>
  </head>
  <body>
    <form action="BingMaps_REST_ReverseGeocodeSample.php" method="post">
      Bing Maps Key: <input type="text" name="key" value=""<?php echo
(isset($_POST['key'])?$_POST['key']:'') ?>"><br>
        Latitude: <input type="text" name="latitude" value=""<?php echo
(isset($_POST['latitude'])?$_POST['latitude']:'') ?>"><br>
          Longitude: <input type="text" name="longitude" value=""<?php echo
(isset($_POST['longitude'])?$_POST['longitude']:'') ?>"><br>
            <input type="submit" value="Submit">
  </form>
<?php
// Reverse geocode a location by point
if(isset($_POST['key']) && isset($_POST['latitude']) &&
isset($_POST['longitude']))
{

  // Set key based on user input
  $key = $_POST['key'];
  $latitude = $_POST['latitude'];
  $longitude = $_POST['longitude'];

  // URL of Bing Maps REST Locations API;
  $baseURL = "http://dev.virtualearth.net/REST/v1/Locations";
  $revGeocodeURL = $baseURL."/".$latitude.",".$longitude."?
output=xml&key=".$key;

   $rgOutput = file_get_contents($revGeocodeURL);
  $rgResponse = new SimpleXMLElement($rgOutput);
```

```
  $address= $rgResponse->ResourceSets->ResourceSet->Resources->Location-
>Address->FormattedAddress;
  echo $address;
  }
else
{
  echo "<p>Please enter your Bing Maps key and a latitude and longitude
pair, then click Submit.</p>";
}

?>
</body>
</html>
```

Routes Code Sample

Save this sample to a file named BingMaps_REST_RoutesSample.php. You will also need the BingMapsHelperFunctions code provided later in this section.

HTML

```
<html>
  <head>
    <title>Using PHP and Bing Maps REST Services Routes API</title>
  </head>
  <body>
    <form action="BingMaps_REST_RoutesSample.php" method="post">
      Bing Maps Key: <input type="text" name="key" value=""<?php echo
(isset($_POST['key'])?$_POST['key']:'') ?>"><br>
        Origin: <input type="text" name="origin" value=""<?php echo
(isset($_POST['origin'])?$_POST['origin']:'') ?>"><br>
          Destination: <input type="text" name="destination" value=""<?php
echo (isset($_POST['destination'])?$_POST['destination']:'') ?>"><br>
            <input type="submit" value="Submit">
  </form>
<?php

include 'BingMapsHelperFunctions.php';

if(isset($_POST['key']) && isset($_POST['origin']) &&
isset($_POST['destination']))
{
  // Set default map width and height
  $mapWidth = 300;
  $mapHeight = 300;

  // URL of Bing Maps REST Routes API;
  $baseURL = "http://dev.virtualearth.net/REST/v1/Routes";

  // Set key based on user input
  $key = $_POST['key'];
```

```
  // construct parameter variables for Routes call
  $wayPoint0 = str_ireplace(" ","%20",$_POST['origin']);
  $wayPoint1 = str_ireplace(" ","%20",$_POST['destination']);
  $optimize = "time";
  $routePathOutput = "Points";
  $distanceUnit = "km";
  $travelMode = "Driving";

  // Construct final URL for call to Routes API
  $routesURL = $baseURL."/".$travelMode."?
wp.0=".$wayPoint0."&wp.1=".$wayPoint1."&optimize=".$optimize."&routePathOutp
ut=".$routePathOutput."&distanceUnit=".$distanceUnit."&output=xml&key=".$key
;

  // Get output from API and convert to XML element using php_xml
  $output = file_get_contents($routesURL);
  $response = new SimpleXMLElement($output);

  // Extract and print number of routes from response
  $numRoutes = $response->ResourceSets->ResourceSet->EstimatedTotal;
  echo "Number of routes found: ".$numRoutes."<br>";

  // Extract and print route instructions from response
  $itinerary = $response->ResourceSets->ResourceSet->Resources->Route-
>RouteLeg->ItineraryItem;

  echo "<ol>";
  for ($i = 0; $i < count($itinerary); $i++) {
    $instruction = $itinerary[$i]->Instruction;
// While looping, construct the $maneuverPoints array for later use (note
casting to double)
$maneuverPoints[$i]->Latitude = (double) $itinerary[$i]->ManeuverPoint-
>Latitude;
$maneuverPoints[$i]->Longitude = (double) $itinerary[$i]->ManeuverPoint-
>Longitude;
echo "<li>".$instruction."</li>";
  }
  echo "</ol>";

}
else
{
  echo "<p>Please enter your Bing Maps key and complete all address fields,
then click submit.</p>";
}
?>
</body>
</html>
```

BingMapsHelperFunctions Code Sample

Save this code to a file named BingMapsHelperFunctions.php. This PHP code is referenced in the Routes code sample.

PHP

```php
<?php

function LatLongToPixel(&$x, &$y, $lat, $long, $clat, $clong, $zoom,
$mapWidth, $mapHeight)
{
$sinLatCenter = sin($clat * pi() / 180);
$pixelXCenter = ($clong + 180) / 360 * 256 * pow(2,$zoom);
$pixelYCenter = (0.5 - log((1 + $sinLatCenter) / (1-$sinLatCenter)) /
(4*pi())) * 256 * pow(2,$zoom);

$sinLat = sin($lat * pi() / 180);
$pixelX = ($long + 180) / 360 * 256 * pow(2,$zoom);
$pixelY = (0.5 - log((1 + $sinLat) / (1-$sinLat)) / (4*pi())) * 256 *
pow(2,$zoom);

$topLeftPixelX = $pixelXCenter - $mapWidth / 2;
$topLeftPixelY = $pixelYCenter - $mapHeight /2;
$x = $pixelX - $topLeftPixelX;
$y = $pixelY - $topLeftPixelY;
}

function haversineDistance($lat1, $lon1, $lat2, $lon2)
{
$radius = 6371;
$factor = pi() / 180;
$dLat = ($lat2-$lat1)*$factor;
$dLon = ($lon2-$lon1)*$factor;
$a = sin($dLat/2) * sin($dLat/2) + cos($lat1*$factor) * cos($lat2*$factor) *
sin($dLon/2) * sin($dLon/2);
$c = 2 * atan2(sqrt($a), sqrt(1-$a));
return $radius*$c;
}

function calculateView($points)
    {

  global $mapWidth, $mapHeight;

      $maxLat = -90;
      $minLat = 90;
      $maxLon = -180;
      $minLon = 180;

      $defaultScales = array(78.27152, 39.13576, 19.56788, 9.78394, 4.89197,
2.44598, 1.22299, 0.61150, 0.30575, 0.15287, .07644, 0.03822, 0.01911,
0.00955, 0.00478, 0.00239, 0.00119, 0.0006, 0.0003);

  // calculate bounding box for array of locations
  for($i = 0; $i < count($points); $i++)
```

```
      {
        if ($points[$i]->Latitude > $maxLat) $maxLat = $points[$i]-
>Latitude;
        if ($points[$i]->Latitude < $minLat) $minLat = $points[$i]-
>Latitude;
        if ($points[$i]->Longitude > $maxLon) $maxLon = $points[$i]-
>Longitude;
        if ($points[$i]->Longitude < $minLon) $minLon = $points[$i]-
>Longitude;
  }

  // calculate center coordinate of bounding box
  $centerLat = ($maxLat + $minLat) / 2;
  $centerLon = ($maxLon + $minLon) / 2;

  // create a Location object for the center point
  $centerPoint = array(
    'Latitude' => $centerLat,
'Longitude' => $centerLon
  );

  // want to calculate the distance in km along the center latitude between
the two longitudes
  $meanDistanceX = haversineDistance($centerLat, $minLon, $centerLat,
$maxLon);

  // want to calculate the distance in km along the center longitude between
the two latitudes
  $meanDistanceY = haversineDistance($maxLat, $centerLon, $minLat,
$centerLon) * 2;

  // calculate the X and Y scales
  $meanScaleValueX = $meanDistanceX / $mapWidth;
  $meanScaleValueY = $meanDistanceY / $mapHeight;

  // gets the largest scale value to work with
  if ($meanScaleValueX > $meanScaleValueY)
    $meanScale = $meanScaleValueX;
  else
    $meanScale = $meanScaleValueY;

      // initialize zoom level variable
  $zoom = 1;

  // calculate zoom level
  for ($i = 1; $i < 19; $i++) {
    if ($meanScale >= $defaultScales[$i]) {$zoom = $i; break;}
  }

  // return a BestView "object" with the center point and zoom level to use
  $bestView = array(
    'CenterPoint' => $centerPoint,
'Zoom' => $zoom
  );
```

```
  return $bestView;
    }

function imagelinethick($image, $x1, $y1, $x2, $y2, $color, $thick = 1)
    {
    /* this way it works well only for orthogonal lines
    imagesetthickness($image, $thick);
    return imageline($image, $x1, $y1, $x2, $y2, $color);
    */
    if ($thick == 1) {
        return imageline($image, $x1, $y1, $x2, $y2, $color);
    }
    $t = $thick / 2 - 0.5;
    if ($x1 == $x2 || $y1 == $y2) {
        return imagefilledrectangle($image, round(min($x1, $x2) - $t),
round(min($y1, $y2) - $t), round(max($x1, $x2) + $t), round(max($y1, $y2) +
$t), $color);
    }
    $k = ($y2 - $y1) / ($x2 - $x1); //y = kx + q
    $a = $t / sqrt(1 + pow($k, 2));
    $points = array(
        round($x1 - (1+$k)*$a), round($y1 + (1-$k)*$a),
        round($x1 - (1-$k)*$a), round($y1 - (1+$k)*$a),
        round($x2 + (1+$k)*$a), round($y2 - (1-$k)*$a),
        round($x2 + (1-$k)*$a), round($y2 + (1+$k)*$a),
    );
    imagefilledpolygon($image, $points, 4, $color);
    return imagepolygon($image, $points, 4, $color);
    }
?>
```

Accessing the Bing Spatial Data Services using PHP

Article • 05/22/2024

> **Note**
>
> **Bing Maps for Enterprise service retirement**
>
> Bing Maps for Enterprise is deprecated and will be retired. Free (Basic) account customers can continue to use Bing Maps for Enterprise services until June 30th, 2025. Enterprise account customers can continue to use Bing Maps for Enterprise services until June 30th, 2028. To avoid service disruptions, all implementations using Bing Maps for Enterprise REST APIs and SDKs will need to be updated to use Azure Maps by the retirement date that applies to your Bing Maps for Enterprise account type.
>
> Azure Maps is Microsoft's next-generation maps and geospatial services for developers. Azure Maps has many of the same features as Bing Maps for Enterprise, and more. To get started with Azure Maps, create a free Azure subscription and an Azure Maps account. For more information about azure Maps, see Azure Maps Documentation. For migration guidance, see Bing Maps Migration Overview.

This article will describe how to write a PHP application that can interact with the Bing Spatial Data Services Geocode Dataflow API.

The Bing Spatial Data Services is a RESTful web service. Representational State Transfer (REST) is an architecture for distributed systems. It follows a stateless client-server model, meaning that there is no memory (context) of past requests stored on the server between client requests. A RESTful web service is a collection of resources, stored under a central URL, which supports a set of operations all of which can be activated using HTTP methods (POST, GET, etc.).

The Bing Spatial Data Services includes a Geocode Dataflow API that allows you to "batch" geocode large sets of spatial data. Use of this API involves starting geocode "jobs" on large lists of spatial data, waiting for the job to complete, and then accessing the successfully (or unsuccessfully) geocoded locations when the job is complete. Spatial data can be uploaded in a variety of formats, including XML, CSV, tab-delimited and pipe-delimited files. These files can be accessed using HTTP GET requests that include a URI and parameters indicating what information is to be returned.

This article will describe how to construct PHP pages that interact with the Bing Spatial Data Services Geocode Dataflow API.

For a complete description of the Bing Spatial Data Services including an API reference, see the Bing Spatial Data Services documentation at Bing Spatial Data Services.

Setting Up Your Environment

Before continuing with this article, you should ensure that you have the correct software installed and environment setup to develop and host PHP pages that will connect to the Bing Spatial Data Services Geocode Dataflow API.

Required Software

To work with and run the sample applications in this article, you will need a PHP-enabled web server. To develop the samples, we used **WampServer** (https://www.wampserver.com/), a Windows-based development environment that includes Apache, PHP, and MySQL Database. It is easy to install and has everything you need to create and host PHP applications quickly and easily.

Microsoft IIS 6 or 7 can also be used to host PHP applications, and we tested several of the examples on it to confirm that they work. If you are using IIS to host PHP, you may also want to install **FastCGI**, which improves performance of CGI applications in IIS. You can find a very detailed set of instructions for installing and configuring FastCGI and PHP at:

- IIS 6 - https://learn.iis.net/page.aspx/247/using-fastcgi-to-host-php-applications-on-iis-60/
- IIS 7 - https://learn.iis.net/page.aspx/246/using-fastcgi-to-host-php-applications-on-iis-7/

To use the code in this article in the Windows environment, you will need to download the php_http extension. For more information about which download to choose, see the **Which version do I choose?** on the PHP for Windows download page: https://windows.php.net/download.

You will also need to activate the php_openssl extension. If you are using Wampserver, you can activate the php_openssl extension by selecting it from the WampServer taskbar menu under PHP->PHP extensions->php_openssl.

There are many other servers that can be configured to host PHP, and you are free to use any of those instead if you are more familiar with their functionality.

Finally, in terms of a development environment, you can use anything from a text editor such as Windows Notepad to a full-fledged PHP IDE such as PHP Designer to write your code.

PHP Starter Code

This article contains numerous samples of interacting with Bing Spatial Data Services Geocode Dataflow API using PHP. If you are already familiar with PHP, you may not need any assistance creating and setting up a PHP page in which to include your Bing Maps code. However, if you need assistance getting started, you may want to begin with the following simple PHP structure which includes a Bing Maps Key that will be discussed in the following section.

Listing 1 - PHP starter code for working with Bing Spatial Data Services Geocode Dataflow API

HTML

```
<<html>
  <head><title>Using the Bing Spatial Data Geocode Dataflow API</title>
</head>
  <body>
    <?php
       $key = "yourKeyHere";
  // Additional code goes here
    ?>
  </body>
</html>
```

Authentication

All Bing Spatial Data Services require authentication from the client each time they are called. In order to authenticate against the Bing Spatial Data Services Geocode Dataflow API, you will need a Bing Maps Key. For information about how to sign up for a Bing Maps Developer Account and get a Bing Maps Key, see Getting a Bing Maps Key.

When you send an HTTP request to the Geocode Dataflow API, which we will discuss in the next section of this article, you must include the Bing Maps Key as a parameter. For example, you might send an HTTP request like the one shown in Listing 2.

Listing 2 - Authenticating using a Bing Maps Key

url

```
http://spatial.virtualearth.net/REST/v1/Dataflows/Geocode?
description=description&input=input&key=YourBingMapsKeyHere
```

In all of the examples in this article, you will see the Bing Maps Key sent to the Bing Spatial Data Services Geocode Dataflow API in this way.

Working with the Geocode Dataflow API

The Geocode Dataflow API is a REST web service that allows you to pass in a list of addresses to be geocoded. The API will attempt to geocode each of the requests and, when finished, provide a link to two sets of data: one set of data lists the results of the successfully geocoded addresses, and another lists any addresses that were not geocoded successfully. You can provide this data in XML, CSV, tab-delimited or pipe-delimited formats. The geocoded data is returned in the same format. As explained below, the Geocode Dataflow API requires that you include additional information in your HTTP request beyond the URI with parameters.

Working with the Geocode Dataflow API involves the following steps:

1. Start the batch geocode by sending the addresses to be geocoded using the following URI format:

   Listing 3 - URI format for starting a geocode job with the Geocode Dataflow API

   ```
   url
   ```

   ```
   http://spatial.virtualearth.net/REST/v1/Dataflows/Geocode?
   description=description&input=input&key={BingMapsKey}
   ```

   This call to the Geocode Dataflow API will create a geocode job, which the API will begin processing. You may have up to 10 simultaneously running geocode jobs for a single Bing Maps Key.

2. Check the status of your job in progress using the following URI format:

   Listing 4 - URI format for checking the status of a geocode job

   ```
   url
   ```

   ```
   http://spatial.virtualearth.net/REST/v1/Dataflows/Geocode/JobId?key=
   {BingMapsKey}
   ```

The *JobId* for a geocode job is returned as part of the response when you initially create the job. The status of a geocode job will start out as "Pending" and eventually change to "Completed" when processing is complete.

3. When the status of your geocode job is set to "Completed", the result data from the status check in listing 4 will include two links:

   a. The **succeeded** link will provide a list of result data for all addresses or places that were successfully geocoded.

   b. The **failed** link will provide a list of addresses or places that could not be successfully geocoded.

Example: Creating a Geocode Job and Obtaining Results

We will now walk through a simple example of how to create a geocode job, check on its status, and obtain results when it has completed.

We will start with the following data file, which consists of three correctly formatted XML blocks describing three addresses that we want to geocode. To use this data, save it in a file named geocodeFeed.xml.

Listing 5 - Sample XML file to be used as input for the Geocode Dataflow API

XML

```
<GeocodeFeed>
  <GeocodeEntity Id="001"

xmlns="http://schemas.microsoft.com/search/local/2010/5/geocode">
    <GeocodeRequest Culture="en US">
      <Address AddressLine="291 Broadway" AdminDistrict="NY" Locality="New
York"
               PostalCode="10007" />
    </GeocodeRequest>
  </GeocodeEntity>
  <GeocodeEntity Id="002"

xmlns="http://schemas.microsoft.com/search/local/2010/5/geocode">
    <GeocodeRequest Culture="en US">
      <Address AddressLine="599 Broadway" AdminDistrict="NY" Locality="New
York" />
      <ConfidenceFilter MinimumConfidence="High" xmlns="" />
    </GeocodeRequest>
  </GeocodeEntity>
  <GeocodeEntity Id="003"

xmlns="http://schemas.microsoft.com/search/local/2010/5/geocode">
    <GeocodeRequest Culture="en US" Query="Empire State Building" />
```

```
  </GeocodeEntity>
</GeocodeFeed>
```

The Geocode Dataflow API accepts input in the following formats:

- XML (as shown in Listing 5)
- Comma-separated values (CSV)
- Pipe-delimited
- Tab-delimited

You can get complete details on the data structure required by the Geocode Dataflow API for all these formats at Data Schema v1.0.

To begin a geocode job, we can use the following PHP code:

Listing 6 - Starting a geocode job using the Geocode Dataflow API

PHP

```
$key = "Insert Your Bing Maps Key Here";
$url = "http://spatial.virtualearth.net/REST/v1/Dataflows/Geocode?
description=MyJob&input=xml&output=xml&key=".$key;

// STEP 1 - Create a geocode job

// Get the contents of an XML data file
$myfile = "geocodefeed.xml";
$data = file_get_contents($myfile);

// Call custom function to generate an HTTP request and get back an HTTP
response
$response = do_post_request($url, $data);

// This function constructs and sends an HTTP request with a provided URL
and data, and returns an HTTP response object
// This function uses the php_http extension
function do_post_request($url, $data, $optional_headers = null)
{
  $request = new HttpRequest($url, HttpRequest::METH_POST);
  $request->setBody($data);
  $response = $request->send();
  return $response->getBody();
}
```

The sample code in Listing 6 includes a custom function, **do_post_request**, which generates and sends an HTTP request using the classes provided by the **php_http**

extension. If this extension is not included in your version of PHP, you can download it from `https://downloads.php.net/pierre/`.

The $response object returned in Listing 6 is an HTTP response. You can pull information out of it by extracting its body (which is in XML format) and then using the PHP XML classes to extract what you need, as shown in Listing 7.

Listing 7 - Working with response data from the Geocode Dataflow request to start a geocode job

PHP

```
// Convert the response body into an XML element so we can extract data
$responseBody = new SimpleXMLElement($response);

// Get data (such as job id, status description, and job status) from the
response
$statusDescription = $responseBody->StatusDescription;
$jobId = $responseBody->ResourceSets->ResourceSet->Resources->DataflowJob-
>Id;
$jobStatus = $responseBody->ResourceSets->ResourceSet->Resources-
>DataflowJob->Status;

echo "Job Created:<br>";
echo " Request Status: ".$statusDescription."<br>";
echo " Job ID: ".$jobId."<br>";
echo " Job Status: ".$jobStatus."<br><br>";
```

Once you have started one or more geocode jobs, you can check on their statuses using the URL described in the previous section. You will need a JobId in order to check the status of a job, which you can obtain using the code from Listing 7. For example, the following PHP code checks the status of a specific job, prints the job description and status, and repeats until the status of the job is "Completed" (instead of "Pending").

Listing 8 - Checking the status of your geocode job

PHP

```
// Call the API to determine the status of all geocode jobs associated with
a Bing Maps key
echo "Checking status until complete...<br>";
while ($jobStatus != "Completed")
{
  // Wait 5 seconds, then check the job's status
  sleep(5);

  // Construct the URL to check the job status, including the jobId
  $checkUrl =
"http://spatial.virtualearth.net/REST/v1/Dataflows/Geocode/".$jobId."?
```

```
output=xml&key=".$key;
  $checkResponse = file_get_contents($checkUrl);
  $responseBody = new SimpleXMLElement($checkResponse);

  // Get and print the description and current status of the job
  $jobDesc = $responseBody->ResourceSets->ResourceSet->Resources-
>DataflowJob->Description;
  $jobStatus = $responseBody->ResourceSets->ResourceSet->Resources-
>DataflowJob->Status;

  echo $jobDesc." - ".$jobStatus."<br>";
}
```

The code in Listing 8 might produce output something like Figure 1.

Figure 1 - Output from code to check status of geocode job

1. Depart Reade St toward Church St
2. Turn right onto Church St
3. Bear left onto 6th Ave / Avenue of the Americas
4. Bear right onto W Broadway
5. Turn right onto W Houston St
6. Arrive at 599 Broadway, New York, NY 10012-3235 on the right

Once you have a job that has completed its run, you can obtain the URL to the successfully geocoded locations (as well as, if needed, the URL for the list of unsuccessful items) from the XML. For example, you could use the PHP code in Listing 9 to obtain the URL to the successfully geocoded results for the geocode job.

Listing 9 - Obtaining the URL to the successfully geocoded results for a geocode job

PHP

```
//Iterate through the links provided with the first geocode job and extract
the 'succeeded' link
$Links = $responseBody->ResourceSets->ResourceSet->Resources->DataflowJob-
>Link;
foreach ($Links as $Link) {
  if ($Link['name'] == "succeeded")
  {
     $successUrl = $Link;
     break;
  }
}
```

Once you have the URL for the successfully geocoded results, you can append your Bing Maps Key to it and send it to the Geocode Dataflow API in order to access the results.

Listing 10 - Getting the successfully geocoded results of a job

PHP

```
// Access the URL for the successful requests, and convert response to an
XML element
$successUrl .= "?output=xml&key=".$key;
$successResponse = file_get_contents($successUrl);
$successResponseBody = new SimpleXMLElement($successResponse);
```

In Listing 10 we (yet again) convert the results from the Geocode Dataflow API call into an XML element so that we can use the PHP XML API to extract results from it. For example, if you wanted to iterate through all of the successfully geocoded results and print out the geocoded addresses and their coordinates (latitude and longitude), you could use the code shown in Listing 11.

Listing 11 - Extracting data from successful geocode job results

PHP

```
// Loop through the geocoded results and output addresses and lat/long
coordinates
foreach ($successResponseBody->GeocodeEntity as $entity) {
  echo $entity->GeocodeResponse->Address['FormattedAddress'],"<br>";
  if (!$entity->GeocodeResponse->RooftopLocation) {
    echo $entity->GeocodeResponse->InterpolatedLocation['Longitude'].", ";
    echo $entity->GeocodeResponse->InterpolatedLocation['Latitude']."<br>";
  }
  else {
    echo $entity->GeocodeResponse->RooftopLocation['Longitude'].", ";
    echo $entity->GeocodeResponse->RooftopLocation['Latitude']."<br>";
  }
}
```

Note that in Listing 11, we need to check which type of location has been provided with each result. Some results may include Rooftop Location coordinates, while others will have Interpolated Location coordinates. The code in Listing 11 prefers Rooftop Location coordinates, and only prints out Interpolated Location coordinates it the Rooftop ones do not exist.You can find complete information on the Geocode Dataflow API, as well as additional data samples and walkthroughs, in the Bing Spatial Data Services documentation at Bing Spatial Data Services.

Conclusions and Further Reading

This article has demonstrated how to use PHP to access the Bing Spatial Data Services Geocode Dataflow API.

You can find full API documentation for the Bing Spatial Data Services at: Bing Spatial Data Services

Complete PHP Sample Code

The following is the complete code sample discussed in this article. This sample runs off the data provided in Listing 5. You must also replace the placeholder text with your Bing Maps Key.

PHP

```
<html>
  <head>
    <title>Using the Bing Spatial Data Geocode Dataflow API</title>
  </head>
  <body>
    <?php

$key = "Insert Bing Maps Key Here";
$url = "http://spatial.virtualearth.net/REST/v1/Dataflows/Geocode?
description=MyJob&input=xml&output=xml&key=".$key;

// STEP 1 - Create a geocode job

// Get the contents of an XML data file
$myfile = "geocodefeed.xml";
$data = file_get_contents($myfile);

// Call custom function to generate an HTTP request and get back an HTTP
response
$response = do_post_request($url, $data);

// This function constructs and sends an HTTP request with a provided URL
and data, and returns an HTTP response object
// This function uses the php_http extension
function do_post_request($url, $data, $optional_headers = null) {
  $request = new HttpRequest($url, HttpRequest::METH_POST);
  $request->setBody($data);
  $response = $request->send();
  return $response->getBody();
}

// Convert the response body into an XML element so we can extract data
$responseBody = new SimpleXMLElement($response);

// Get data (such as job id, status description, and job status) from the
response
$statusDescription = $responseBody->StatusDescription;
```

```
$jobId = $responseBody->ResourceSets->ResourceSet->Resources->DataflowJob->Id;
$jobStatus = $responseBody->ResourceSets->ResourceSet->Resources->DataflowJob->Status;

echo "Job Created:<br>";
echo " Request Status: ".$statusDescription."<br>";
echo " Job ID: ".$jobId."<br>";
echo " Job Status: ".$jobStatus."<br><br>";

// STEP 2 - Get the status of geocode job(s)

// Call the API to determine the status of all geocode jobs associated with a Bing Maps key
echo "Checking status until complete...<br>";
while ($jobStatus != "Completed") {

  // Wait 5 seconds, then check the job's status
  sleep(5);

  // Construct the URL to check the job status, including the jobId
  $checkUrl = "http://spatial.virtualearth.net/REST/v1/Dataflows/Geocode/".$jobId."?output=xml&key=".$key;
  $checkResponse = file_get_contents($checkUrl);
  $responseBody = new SimpleXMLElement($checkResponse);

  // Get and print the description and current status of the job
  $jobDesc = $responseBody->ResourceSets->ResourceSet->Resources->DataflowJob->Description;
  $jobStatus = $responseBody->ResourceSets->ResourceSet->Resources->DataflowJob->Status;

  echo $jobDesc." - ".$jobStatus."<br>";

}

// STEP 3 - Obtain results from a successfully geocoded set of data

//Iterate through the links provided with the first geocode job and extract the 'succeeded' link
$Links = $responseBody->ResourceSets->ResourceSet->Resources->DataflowJob->Link;
foreach ($Links as $Link) {
  if ($Link['name'] == "succeeded")
  {
    $successUrl = $Link;
    break;
  }
}

// Access the URL for the successful requests, and convert response to an XML element
$successUrl .= "?output=xml&key=".$key;
$successResponse = file_get_contents($successUrl);
```

```
$successResponseBody = new SimpleXMLElement($successResponse);

// Loop through the geocoded results and output addresses and lat/long
coordinates
foreach ($successResponseBody->GeocodeEntity as $entity) {
  echo $entity->GeocodeResponse->Address['FormattedAddress'],"<br>";
  if (!$entity->GeocodeResponse->RooftopLocation) {
    echo $entity->GeocodeResponse->InterpolatedLocation['Longitude'].", ";
    echo $entity->GeocodeResponse->InterpolatedLocation['Latitude']."<br>";
  }
  else {
    echo $entity->GeocodeResponse->RooftopLocation['Longitude'].", ";
    echo $entity->GeocodeResponse->RooftopLocation['Latitude']."<br>";
  }
}

?>

  </body>
</html>
```

Bing Maps Tile System

Article • 05/22/2024

> **Note**
>
> **Bing Maps for Enterprise service retirement**
>
> Bing Maps for Enterprise is deprecated and will be retired. Free (Basic) account customers can continue to use Bing Maps for Enterprise services until June 30th, 2025. Enterprise account customers can continue to use Bing Maps for Enterprise services until June 30th, 2028. To avoid service disruptions, all implementations using Bing Maps for Enterprise REST APIs and SDKs will need to be updated to use Azure Maps by the retirement date that applies to your Bing Maps for Enterprise account type.
>
> Azure Maps is Microsoft's next-generation maps and geospatial services for developers. Azure Maps has many of the same features as Bing Maps for Enterprise, and more. To get started with Azure Maps, create a free Azure subscription and an Azure Maps account. For more information about azure Maps, see Azure Maps Documentation. For migration guidance, see Bing Maps Migration Overview.

Bing Maps provides a world map that users can directly manipulate to pan and zoom. To make this interaction as fast and responsive as possible, we chose to pre-render the map at many different levels of detail, and to cut each map into tiles for quick retrieval and display. This document describes the projection, coordinate systems, and addressing scheme of the map tiles, which collectively are called the Bing Maps Tile System.

Map Projection

To make the map seamless, and to ensure that aerial images from different sources line up properly, we have to use a single projection for the entire world. We chose to use the **Mercator projection**, which looks like this:

Although the Mercator projection significantly distorts scale and area (particularly near the poles), it has two important properties that outweigh the scale distortion:

1. It's a **conformal** projection, which means that it preserves the shape of relatively small objects. This is especially important when showing aerial imagery, because we want to avoid distorting the shape of buildings. Square buildings should appear square, not rectangular.

2. It's a **cylindrical** projection, which means that north and south are always straight up and down, and west and east are always straight left and right.

Since the Mercator projection goes to infinity at the poles, it doesn't actually show the entire world. Using a square aspect ratio for the map, the maximum latitude shown is approximately 85.05 degrees.

To simplify the calculations, we use the spherical form of this projection, not the ellipsoidal form. Since the projection is used only for map display, and not for displaying numeric coordinates, we don't need the extra precision of an ellipsoidal projection. The spherical projection causes approximately 0.33% scale distortion in the Y direction, which is not visually noticeable.

Ground Resolution and Map Scale

In addition to the projection, the ground resolution or map scale must be specified in order to render a map. At the lowest level of detail (Level 1), the map is 512 x 512 pixels. At each successive level of detail, the map width and height grow by a factor of 2: Level 2 is 1024 x 1024 pixels, Level 3 is 2048 x 2048 pixels, Level 4 is 4096 x 4096 pixels, and so on. In general, the width and height of the map (in pixels) can be calculated as:

$$\texttt{map width = map height = 256 * 2}^{\text{level}}\ \texttt{pixels}$$

The **ground resolution** indicates the distance on the ground that's represented by a single pixel in the map. For example, at a ground resolution of 10 meters/pixel, each pixel represents a ground distance of 10 meters. The ground resolution varies depending on the level of detail and the latitude at which it's measured. Using an earth radius of 6378137 meters, the ground resolution (in meters per pixel) can be calculated as:

```
ground resolution = cos(latitude * pi/180) * earth circumference / map width
```

```
= (cos(latitude * pi/180) * 2 * pi * 6378137 meters) / (256 * 2^level pixels)
```

The **map scale** indicates the ratio between map distance and ground distance, when measured in the same units. For instance, at a map scale of 1 : 100,000, each inch on the map represents a ground distance of 100,000 inches. Like the ground resolution, the map scale varies with the level of detail and the latitude of measurement. It can be calculated from the ground resolution as follows, given the screen resolution in dots per inch, typically 96 dpi:

```
map scale = 1 : ground resolution * screen dpi / 0.0254 meters/inch
```

```
= 1 : (cos(latitude * pi/180) * 2 * pi * 6378137 * screen dpi) / (256 * 2^level * 0.0254)
```

This table shows each of these values at each level of detail, **as measured at the Equator**. (Note that the ground resolution and map scale also vary with the latitude, as shown in the equations above, but not shown in the table below.)

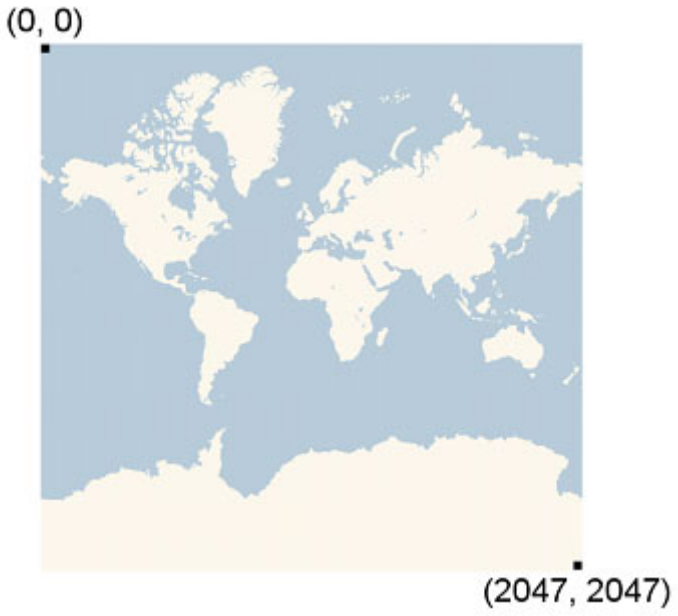
Expand table

Level of Detail	Map Width and Height (pixels)	Ground Resolution (meters / pixel)	Map Scale (at 96 dpi)
1	512	78,271.5170	1 : 295,829,355.45
2	1,024	39,135.7585	1 : 147,914,677.73
3	2,048	19,567.8792	1 : 73,957,338.86
4	4,096	9,783.9396	1 : 36,978,669.43
5	8,192	4,891.9698	1 : 18,489,334.72
6	16,384	2,445.9849	1 : 9,244,667.36
7	32,768	1,222.9925	1 : 4,622,333.68

Level of Detail	Map Width and Height (pixels)	Ground Resolution (meters / pixel)	Map Scale (at 96 dpi)
8	65,536	611.4962	1 : 2,311,166.84
9	131,072	305.7481	1 : 1,155,583.42
10	262,144	152.8741	1 : 577,791.71
11	524,288	76.4370	1 : 288,895.85
12	1,048,576	38.2185	1 : 144,447.93
13	2,097,152	19.1093	1 : 72,223.96
14	4,194,304	9.5546	1 : 36,111.98
15	8,388,608	4.7773	1 : 18,055.99
16	16,777,216	2.3887	1 : 9,028.00
17	33,554,432	1.1943	1 : 4,514.00
18	67,108,864	0.5972	1 : 2,257.00
19	134,217,728	0.2986	1 : 1,128.50
20	268,435,456	0.1493	1 : 564.25
21	536,870,912	0.0746	1 : 282.12
22	1,073,741,824	0.0373	1 : 141.06
23	2,147,483,648	0.0187	1 : 70.53

Pixel Coordinates

Having chosen the projection and scale to use at each level of detail, we can convert geographic coordinates into pixel coordinates. Since the map width and height is different at each level, so are the pixel coordinates. The pixel at the upper-left corner of the map always has pixel coordinates (0, 0). The pixel at the lower-right corner of the map has pixel coordinates (width-1, height-1), or referring to the equations in the previous section, ($256 * 2^{level}–1$, $256 * 2^{level}–1$). For example, at level 3, the pixel coordinates range from (0, 0) to (2047, 2047), like this:



Given latitude and longitude in degrees, and the level of detail, the pixel XY coordinates can be calculated as follows:

```
sinLatitude = sin(latitude * pi/180)
```

```
pixelX = ((longitude + 180) / 360) * 256 * 2
```
level

```
pixelY = (0.5 – log((1 + sinLatitude) / (1 – sinLatitude)) / (4 * pi)) * 256 * 2
```
level

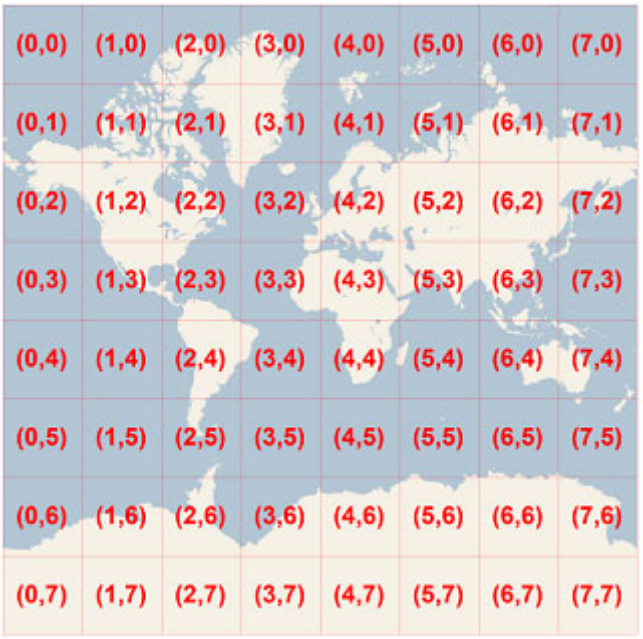
The latitude and longitude are assumed to be on the WGS 84 datum. Even though Bing Maps uses a spherical projection, it's important to convert all geographic coordinates into a common datum, and WGS 84 was chosen to be that datum. The longitude is assumed to range from -180 to +180 degrees, and the latitude must be clipped to range from -85.05112878 to 85.05112878. This avoids a singularity at the poles, and it causes the projected map to be square.

Tile Coordinates and Quadkeys

To optimize the performance of map retrieval and display, the rendered map is cut into tiles of 256 x 256 pixels each. As the number of pixels differs at each level of detail, so does the number of tiles:

```
map width = map height = 2
```
level
```
tiles
```

Each tile is given XY coordinates ranging from (0, 0) in the upper left to (2^{level}–1, 2^{level}–1) in the lower right. For example, at level 3 the tile coordinates range from (0, 0) to (7, 7) as follows:



Given a pair of pixel XY coordinates, you can easily determine the tile XY coordinates of the tile containing that pixel:

```
tileX = floor(pixelX / 256)
```

```
tileY = floor(pixelY / 256)
```

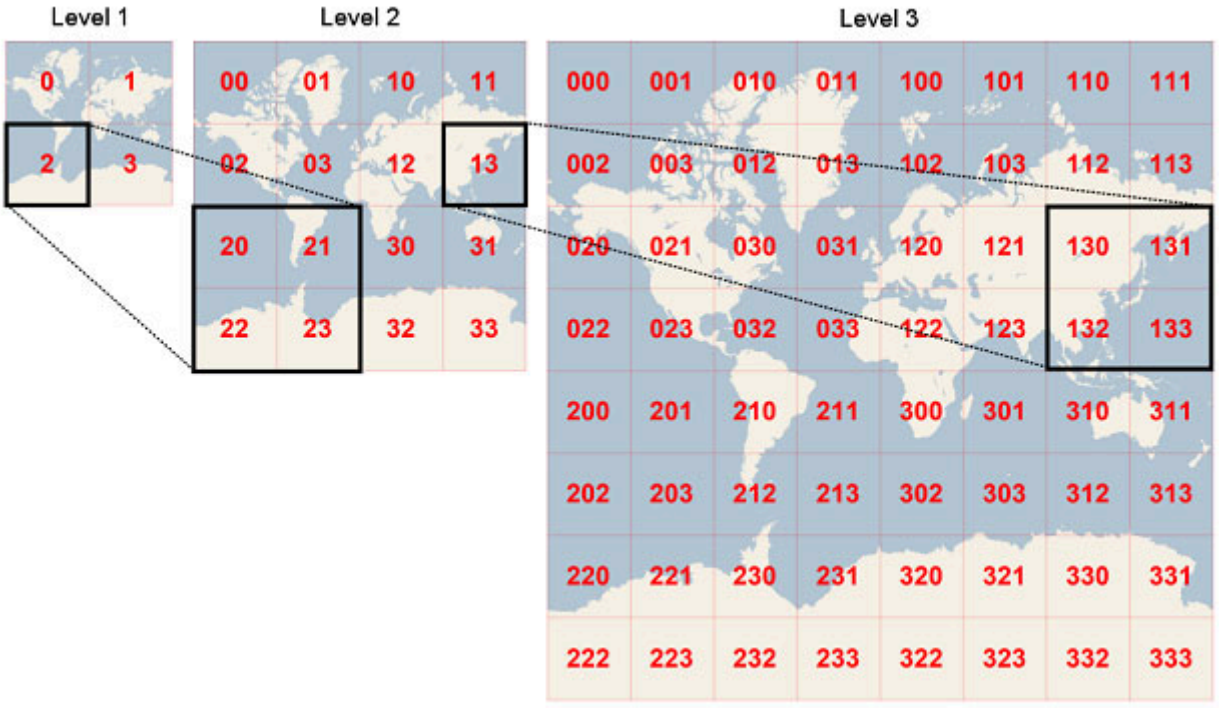
To optimize the indexing and storage of tiles, the two-dimensional tile XY coordinates are combined into one-dimensional strings called quadtree keys, or “quadkeys” for short. Each quadkey uniquely identifies a single tile at a particular level of detail, and it can be used as an key in common database B-tree indexes. To convert tile coordinates into a quadkey, the bits of the Y and X coordinates are interleaved, and the result is interpreted as a base-4 number (with leading zeros maintained) and converted into a string. For instance, given tile XY coordinates of (3, 5) at level 3, the quadkey is determined as follows:

`tileX = 3 = 011`$_2$

`tileY = 5 = 101`$_2$

`quadkey = 100111`$_2$ `= 213`$_4$ `= “213”`

Quadkeys have several interesting properties. First, the length of a quadkey (the number of digits) equals the level of detail of the corresponding tile. Second, the quadkey of any tile starts with the quadkey of its parent tile (the containing tile at the previous level). As shown in the example below, tile 2 is the parent of tiles 20 through 23, and tile 13 is the parent of tiles 130 through 133:



Finally, quadkeys provide a one-dimensional index key that usually preserves the proximity of tiles in XY space. In other words, two tiles that have nearby XY coordinates usually have quadkeys that are relatively close together. This is important for optimizing database performance, because neighboring tiles are usually requested in groups, and it's desirable to keep those tiles on the same disk blocks, in order to minimize the number of disk reads.

Sample Code

The following sample C# code illustrates how to implement the functions described in this document. These functions can be easily translated into other programming languages as needed.

CSharp

```
//---------------------------------------------------------------------------
----
// <copyright company="Microsoft">
//     Copyright (c) 2006-2009 Microsoft Corporation.  All rights reserved.
// </copyright>
//---------------------------------------------------------------------------
----

using System;
using System.Text;

namespace Microsoft.MapPoint
{
    static class TileSystem
```

```
    {
        private const double EarthRadius = 6378137;
        private const double MinLatitude = -85.05112878;
        private const double MaxLatitude = 85.05112878;
        private const double MinLongitude = -180;
        private const double MaxLongitude = 180;

        /// <summary>
        /// Clips a number to the specified minimum and maximum values.
        /// </summary>
        /// <param name="n">The number to clip.</param>
        /// <param name="minValue">Minimum allowable value.</param>
        /// <param name="maxValue">Maximum allowable value.</param>
        /// <returns>The clipped value.</returns>
        private static double Clip(double n, double minValue, double
maxValue)
        {
            return Math.Min(Math.Max(n, minValue), maxValue);
        }

        /// <summary>
        /// Determines the map width and height (in pixels) at a specified
level
        /// of detail.
        /// </summary>
        /// <param name="levelOfDetail">Level of detail, from 1 (lowest
detail)
        /// to 23 (highest detail).</param>
        /// <returns>The map width and height in pixels.</returns>
        public static uint MapSize(int levelOfDetail)
        {
            return (uint) 256 << levelOfDetail;
        }

        /// <summary>
        /// Determines the ground resolution (in meters per pixel) at a
specified
        /// latitude and level of detail.
        /// </summary>
        /// <param name="latitude">Latitude (in degrees) at which to measure
the
        /// ground resolution.</param>
        /// <param name="levelOfDetail">Level of detail, from 1 (lowest
detail)
        /// to 23 (highest detail).</param>
        /// <returns>The ground resolution, in meters per pixel.</returns>
        public static double GroundResolution(double latitude, int
levelOfDetail)
        {
            latitude = Clip(latitude, MinLatitude, MaxLatitude);
            return Math.Cos(latitude * Math.PI / 180) * 2 * Math.PI *
EarthRadius / MapSize(levelOfDetail);
        }

        /// <summary>
```

```
        /// Determines the map scale at a specified latitude, level of
detail,
        /// and screen resolution.
        /// </summary>
        /// <param name="latitude">Latitude (in degrees) at which to measure
the
        /// map scale.</param>
        /// <param name="levelOfDetail">Level of detail, from 1 (lowest
detail)
        /// to 23 (highest detail).</param>
        /// <param name="screenDpi">Resolution of the screen, in dots per
inch.</param>
        /// <returns>The map scale, expressed as the denominator N of the
ratio 1 : N.</returns>
        public static double MapScale(double latitude, int levelOfDetail,
int screenDpi)
        {
            return GroundResolution(latitude, levelOfDetail) * screenDpi /
0.0254;
        }

        /// <summary>
        /// Converts a point from latitude/longitude WGS-84 coordinates (in
degrees)
        /// into pixel XY coordinates at a specified level of detail.
        /// </summary>
        /// <param name="latitude">Latitude of the point, in degrees.
</param>
        /// <param name="longitude">Longitude of the point, in degrees.
</param>
        /// <param name="levelOfDetail">Level of detail, from 1 (lowest
detail)
        /// to 23 (highest detail).</param>
        /// <param name="pixelX">Output parameter receiving the X coordinate
in pixels.</param>
        /// <param name="pixelY">Output parameter receiving the Y coordinate
in pixels.</param>
        public static void LatLongToPixelXY(double latitude, double
longitude, int levelOfDetail, out int pixelX, out int pixelY)
        {
            latitude = Clip(latitude, MinLatitude, MaxLatitude);
            longitude = Clip(longitude, MinLongitude, MaxLongitude);

            double x = (longitude + 180) / 360;
            double sinLatitude = Math.Sin(latitude * Math.PI / 180);
            double y = 0.5 - Math.Log((1 + sinLatitude) / (1 - sinLatitude))
/ (4 * Math.PI);

            uint mapSize = MapSize(levelOfDetail);
            pixelX = (int) Clip(x * mapSize + 0.5, 0, mapSize - 1);
            pixelY = (int) Clip(y * mapSize + 0.5, 0, mapSize - 1);
        }

        /// <summary>
        /// Converts a pixel from pixel XY coordinates at a specified level
```

```
of detail
        /// into latitude/longitude WGS-84 coordinates (in degrees).
        /// </summary>
        /// <param name="pixelX">X coordinate of the point, in pixels.
</param>
        /// <param name="pixelY">Y coordinates of the point, in pixels.
</param>
        /// <param name="levelOfDetail">Level of detail, from 1 (lowest
detail)
        /// to 23 (highest detail).</param>
        /// <param name="latitude">Output parameter receiving the latitude
in degrees.</param>
        /// <param name="longitude">Output parameter receiving the longitude
in degrees.</param>
        public static void PixelXYToLatLong(int pixelX, int pixelY, int
levelOfDetail, out double latitude, out double longitude)
        {
            double mapSize = MapSize(levelOfDetail);
            double x = (Clip(pixelX, 0, mapSize - 1) / mapSize) - 0.5;
            double y = 0.5 - (Clip(pixelY, 0, mapSize - 1) / mapSize);

            latitude = 90 - 360 * Math.Atan(Math.Exp(-y * 2 * Math.PI)) /
Math.PI;
            longitude = 360 * x;
        }

        /// <summary>
        /// Converts pixel XY coordinates into tile XY coordinates of the
tile containing
        /// the specified pixel.
        /// </summary>
        /// <param name="pixelX">Pixel X coordinate.</param>
        /// <param name="pixelY">Pixel Y coordinate.</param>
        /// <param name="tileX">Output parameter receiving the tile X
coordinate.</param>
        /// <param name="tileY">Output parameter receiving the tile Y
coordinate.</param>
        public static void PixelXYToTileXY(int pixelX, int pixelY, out int
tileX, out int tileY)
        {
            tileX = pixelX / 256;
            tileY = pixelY / 256;
        }

        /// <summary>
        /// Converts tile XY coordinates into pixel XY coordinates of the
upper-left pixel
        /// of the specified tile.
        /// </summary>
        /// <param name="tileX">Tile X coordinate.</param>
        /// <param name="tileY">Tile Y coordinate.</param>
        /// <param name="pixelX">Output parameter receiving the pixel X
coordinate.</param>
        /// <param name="pixelY">Output parameter receiving the pixel Y
coordinate.</param>
```

```
        public static void TileXYToPixelXY(int tileX, int tileY, out int
pixelX, out int pixelY)
        {
            pixelX = tileX * 256;
            pixelY = tileY * 256;
        }

        /// <summary>
        /// Converts tile XY coordinates into a QuadKey at a specified level
of detail.
        /// </summary>
        /// <param name="tileX">Tile X coordinate.</param>
        /// <param name="tileY">Tile Y coordinate.</param>
        /// <param name="levelOfDetail">Level of detail, from 1 (lowest
detail)
        /// to 23 (highest detail).</param>
        /// <returns>A string containing the QuadKey.</returns>
        public static string TileXYToQuadKey(int tileX, int tileY, int
levelOfDetail)
        {
            StringBuilder quadKey = new StringBuilder();
            for (int i = levelOfDetail; i > 0; i--)
            {
                char digit = '0';
                int mask = 1 << (i - 1);
                if ((tileX & mask) != 0)
                {
                    digit++;
                }
                if ((tileY & mask) != 0)
                {
                    digit++;
                    digit++;
                }
                quadKey.Append(digit);
            }
            return quadKey.ToString();
        }

        /// <summary>
        /// Converts a QuadKey into tile XY coordinates.
        /// </summary>
        /// <param name="quadKey">QuadKey of the tile.</param>
        /// <param name="tileX">Output parameter receiving the tile X
coordinate.</param>
        /// <param name="tileY">Output parameter receiving the tile Y
coordinate.</param>
        /// <param name="levelOfDetail">Output parameter receiving the level
of detail.</param>
        public static void QuadKeyToTileXY(string quadKey, out int tileX,
out int tileY, out int levelOfDetail)
        {
            tileX = tileY = 0;
            levelOfDetail = quadKey.Length;
            for (int i = levelOfDetail; i > 0; i--)
```

```
            {
                int mask = 1 << (i - 1);
                switch (quadKey[levelOfDetail - i])
                {
                    case '0':
                        break;

                    case '1':
                        tileX |= mask;
                        break;

                    case '2':
                        tileY |= mask;
                        break;

                    case '3':
                        tileX |= mask;
                        tileY |= mask;
                        break;

                    default:
                        throw new ArgumentException("Invalid QuadKey digit
sequence.");
                }
            }
        }
    }
}
```

About the Author

Joe Schwartz is a software architect for Bing Maps.

Create a Custom Map URL

Article • 07/23/2024

If you want to email someone a map URL or embed a map into your website, you can get the map you need in most cases by clicking **Share** at https://bing.com/maps. From the **Share** dialog box, you can click **Customize and Preview** to further customize your map URL. However, there may be times when you want more control or options such as search results and items in your places list. In this case, you can use the information in this article to build your own Bing Maps URL. `

Getting Started

To create your map link, start with the base URL for Bing Maps shown below, and then add parameters to customize the map. Parameters specify things like the map center point, zoom level, map view (area that you want your map to display), search results and more. Available parameters and examples are provided in the sections below.

1. To build your own URL, start with the base map URL:

   `https://bing.com/maps/default.aspx`

2. Add a question mark (?):

   `https://bing.com/maps/default.aspx?`

3. Add the first parameter that you want to use, and then set the value of the parameter by using an equal sign (=):

   This example sets the centerpoint of the map.

   `https://bing.com/maps/default.aspx?cp=47.677797~-122.122013`

General parameters

The following are some common parameters that customize your map.

Expand table

Parameter	Definition	Example	Details
cp	center point	cp=47.677797~-122.122013	Defines where the center of the map should be. Use the following format for the cp parameter: `Latitude~Longitude` Both values must be expressed in decimal degrees. Latitude and longitude are commonly presented in decimal degrees as two numbers, such as -47.677797 (latitude) and -122.122013 (longitude).
lvl	zoom level	lvl=5	Defines the zoom level of the map view. Valid values are 1-20. This parameter is ignored when a search parameter, such as **ss** or **where1**, is specified. A table of search parameters is provided below.
style	map view	style=r	Defines the map view. Valid values for this parameter include: - **a**: Display an aerial view of the map. - **r**: Display a road view of the map. - **h**: Display an aerial view of the map with labels. - **o**: Use this value to display a bird's eye (oblique) view of the map. - **b**: Display a bird's eye (oblique) with labels view of the map.
scene	scene ID reference	scene=3715328	Specifies the ID of the bird's eye (oblique) image tile to display. You can use this parameter with a **lvl** value of 1 or 2 and a **dir** value to view different formats of the map image.
dir	direction	dir=180	Specifies the direction that the camera is pointing in degrees. Valid values are 0 for North, 90 for East, 180 for South, and 270 for West.
trfc	traffic	trfc=1	Specifies whether traffic information is included on the map. Omitting the trfc parameter produces the same results as trfc=0

Examples

To open Bing Maps with the map centered on a specific location with a zoom level of 12, and the map view set to the road map view:

```
url

https://bing.com/maps/default.aspx?cp=43.901683~-69.522416&lvl=12&style=r
```

To open Bing Maps with the map centered on a specific location, the map view set to bird's eye (oblique) image, and a specified zoom level and scene:

```
url

https://bing.com/maps/default.aspx?cp=37.814692~-122.477339&style=o&lvl=1&dir=0&scene=1140291
```

To open Bing Maps with the map centered on a specific location, the map view set to bird's eye (oblique), and with a traffic overlay displayed:

```
url

https://bing.com/maps/default.aspx?cp=47.621065~-122.352534&style=o&lvl=14&trfc=1
```

Search parameters

To create a map that displays specific search results, use the following parameters.

Expand table

Parameter	Definition	Example	Details
where1	location	where1=1 Microsoft Way, Redmond, WA	Defines a location to center the map based on a specific address or a place name. The text is the same text that you type in the upper search box in Bing Maps to search for a specific address or place name.
ss	search type search	ss=coffee	Defines the searches that you want to display. Use this parameter to display search results for a business.

Search Parameter Modifiers

Use the following search parameter modifiers with the **ss** parameter to specify how results are displayed.

Expand table

Modifier	Definition	Example	Details
sst	search parameter modifier	sst.1	You can append the **sst** parameter to a search string using a tilde (~) to specify how the search results are sorted. Use the following values with the sst parameter to specify the sort option. **0**: Relevance **1**: Distance **2**: Rating
pg	search parameter modifier	pg.2	You can append the **pg** parameter to a search string using a tilde (~) to specify which page of results to display.

Examples

Open Bing Maps to a specific address:

```
url

https://bing.com/maps/default.aspx?
rtp=adr.One%20Microsoft%20Way,%20Redmond,%20WA%2098052~pos.45.23423_-122.1232_MyPlace&rtop=0~1~0
```

Open Bing Maps with a business search:

```
url
```

```
https://bing.com/maps/default.aspx?ss=yp.Pizza~sst.1~pg.2
```

Driving directions parameters

To create a map that displays directions from a specific start and end point, use the following parameters.

Expand table

Parameter	Definition	Example	Details
rtp	route	Rtp=adr.Seattle,WA~adr.One%20Microsoft%20Way,Redmond,WA	Defines the start and end of a route to draw on the map, each separated by a tilde (~). Each of the waypoints is defined by either a pos (position) or adr (address) identifier. These identifiers are described in the table below. A complete route contains at least two waypoints. For example, a route with two waypoints is defined by the following: `rtp="A"~"B"` You can also specify an incomplete route. For example, you can define only the start of a route: `rtp="A"~` Or, you can enter only the end of a route: `rtp=~"B"` If you provide only one waypoint, the driving directions panel is displayed with the provided waypoint, but no route is drawn.
rtop	route options	rtop=0~1~0	Defines options for the route. There are three sets of options, each separated by a tilde (~) The first option specifies how the route is chosen. The second option specifies whether traffic is displayed. The third option must be set to 0 if specified You can choose to not specify a value and get the default value of 0, but you must preserve the order of the values with the tilde (~) separators. For example, the follow examples are all valid values: 0~1~, ~1~0, 1~~0, and 1~~. The default value of 0~0~0 (quickest without traffic) will be used if this parameter is not specified. Options for how the route is chosen (first option): **0**: Quickest time [default] **1**: Shortest distance Options for displaying traffic (second option): **0**: Traffic is not displayed [default] **1**: Traffic is displayed
mode	mode	mode=D	Defines the mode of transportation. Use the following values: - **D**: Driving - **T**: Transit - **W**: Walking
limit	limit	limit=D	Defines whether the route is limited by the arrival or departure time or chooses the last possible train that day. Use the following values: - **D**: Depart at

Parameter	Definition	Example	Details
			- **A**: Arrive by - **LT**: Last train (Japanese market only)
time	time	time=201009302015	Defines the time for which the Depart or Arrive limit is based upon and based on the 24-hour timekeeping system. Example represents Sept 30, 2010 at 8:15pm. Use the following format: `YYYYMMDDhhmm`

Rtp Parameter Identifiers

Use the following identifiers with the **rtp** parameter to specify the endpoints of a route.

Expand table

Identifier for rtp	Definition	Example	Details
pos	position	rtp=pos.42.2_-122.3~pos.55.2_-127.0	Defines a waypoint as a specific position on the map. Use the following format: `rtp=pos.latitude_longitude_name`
adr	address	rtp=adr.Seattle,WA~adr.One%20Microsoft%20Way,Redmond,WA	Defines a waypoint as an address. Use the following format: `rtp=adr.address` Make sure your replace blank spaces in the address with the encoded string %20.

You can use any of the parameters listed in the previous tables to specify your waypoints.

The destination is specified using the **rtp** and **pos** parameters described above. Make sure that you specify only the end location, for example rtp=~pos.45.21_-123.2.

Example

Open Bing Maps and show a route from a specific address to a specific point

url

```
https://bing.com/maps/default.aspx?
rtp=adr.One%20Microsoft%20Way,Redmond,WA%2098052~pos.45.23423_-122.1232_MyPlace&rtop=0~1~0
```

Open Bing Maps and show the driving directions panel with a start address only

url

```
https://bing.com/maps/default.aspx?rtp=adr.Seattle,WA
```

Open Bing Maps and show driving directions to a fictitious pizza restaurant in Redmond, WA

url

```
https://bing.com/maps/default.aspx?rtp=~pos.rycz2z4tpkxm_555%20NE%2055th%20St,Redmond,WA_Pizza%20Parlor_425-555-5555
```

Collections editor and collections parameters

To create a map that displays information from the collections editor or a specific collection, use the following parameters.

Expand table

Parameter	Definition	Example	Details
sp	collections editor	sp=adr.One%20Microsoft%20Way,Redmond,WA	Defines a specific entity, address, or pin to add to the map. Collections editor items are defined as a category and value, separated by a period. There are five categories: **adr**, **point**, **polyline**, **polygon**, and **yp**. These are described in the next table. Separate multiple items with a tilde (~). If an item contains a tilde, make sure the tilde is encoded as %7E.
cid	collection ID	cid=15A41C376	Specifies the collection that you want to display by using the ID assigned to that collection. For the collection ID parameter, use the following format: `cid=collection ID`

Collections Categories

Use the following formats to add values to a collection.

Expand table

Category	Definition	Details
adr	address	Specifies an address to add to the collections editor. For the address, the value can be the address string, the address string and title, or the a sp=adr.addressStringsp=adr.addressString_titlesp=adr.addressString_title_description Make sure that the addresses you provide are as specific as possible.
point	point	Specifies a point to display on the map. For points, the value includes the latitude, longitude, title, notes, a reference URL, and a photo URL, ea sp=point.latitude_longitude_titleString_notesString_linkURL_photoURL
polyline	polyline	Specifies a polyline on the map by specifying a set of points. For polylines, the value includes a set of latitude and longitude points, a title, not color, fill color, line weight, line style, dash style, and the latitude and longitude of the label, each separated by an underscore (_). sp=polyline.lat1_long1_lat2_long2_..._titleString_notesString_linkURL_photoURL_strokeColor_fillColor_strokeWeight_ strokeStyle_strokeDashSty Fill color and stroke color are each specified as hexadecimal RGB values, such as #00ff00. Stroke weight is specified as a pixel value, such as 4px. Stroke style includes the following values: Single, ThinThin, ThickThin, ThinThick, ThickBetweenThin. Stroke dash style includes the following values: Solid, ShortDash, ShortDot, ShortDashDot, ShortDashDotDot, Dot, Dash, LongDash, DashDot,
polygon	polygon	Specifies a polygon to draw on the map by using the lat/long positions of its vertices. For polygons, the value includes pairs of latitude and lo URL, photo URL, line color, fill color, line weight, line style, dash style, and lat/long position of the label, each separated by an underscore (_). sp=polyline.lat1_long1_lat2_long2_lat3_long3_..._titleString_notesString_linkURL_photoURL_strokeColor_fillColor_strokeWeight_strokeStyle_str Fill color and stroke color are each specified as hexadecimal RGB values, such as #00ff00. Stroke weight is specified as a pixel value, such as 4px. Stroke style includes the following values: Single, ThinThin, ThickThin, ThinThick, ThickBetweenThin. Stroke dash style includes the following values: Solid, ShortDash, ShortDot, ShortDashDot, ShortDashDotDot, Dot, Dash, LongDash, DashDot,

Examples

Open Bing Maps and add the address "1 Microsoft Way, Redmond, WA 98052" to the collections editor

url

```
https://bing.com/maps/default.aspx?sp=adr.1%20Microsoft%20Way%2C%20Redmond%2C%20WA%2098052
```

Open Bing Maps and add a polyline to the collections editor:

url

```
https://bing.com/maps/default.aspx?
sp=Polyline.47.68_-122.12_48.68_-123.12_49.68_-122.12_LINE_some%20notes_http://bing.com__%2300ff00__4px_Single_Solid
```

> **Note**
>
> The parameters described in this topic are subject to change.

Custom Map Styles in Bing Maps

Article • 05/22/2024

> **Note**
>
> **Bing Maps for Enterprise service retirement**
>
> Bing Maps for Enterprise is deprecated and will be retired. Free (Basic) account customers can continue to use Bing Maps for Enterprise services until June 30th, 2025. Enterprise account customers can continue to use Bing Maps for Enterprise services until June 30th, 2028. To avoid service disruptions, all implementations using Bing Maps for Enterprise REST APIs and SDKs will need to be updated to use Azure Maps by the retirement date that applies to your Bing Maps for Enterprise account type.
>
> Azure Maps is Microsoft's next-generation maps and geospatial services for developers. Azure Maps has many of the same features as Bing Maps for Enterprise, and more. To get started with Azure Maps, create a free Azure subscription and an Azure Maps account. For more information about azure Maps, see Azure Maps Documentation. For migration guidance, see Bing Maps Migration Overview.

Depending on which API or service you are using there are two different ways to specify a custom style in Bing Maps. The Bing Maps V8 web and Windows 10 UWP controls use a JSON style object, while the Bing Maps REST services and map tiles use a formatted string.

The JSON style schema is very comprehensive. The Windows 10 UWP map control makes full use of the schema while the Bing Maps V8 Web control and REST services support a subset of the schema. A JSON style created for the Windows 10 UWP map control will work with the V8 Web Control and vice-versa. Any unsupported style settings will simply be ignored.

This documentation will focus on custom map styles in Bing Maps V8 Web Control and the Bing Maps REST services. Documentation on using custom map styles in the Windows 10 UWP control can be found here.

Custom Map Styles in the Bing Maps V8 Web Control

The Bing Maps V8 control has a new map option called `customMapStyle` which can be set when loading the map. This property expects a JSON map **ICustomMapStyle** object. Here is an example of how to add a custom map style to Bing Maps V8.

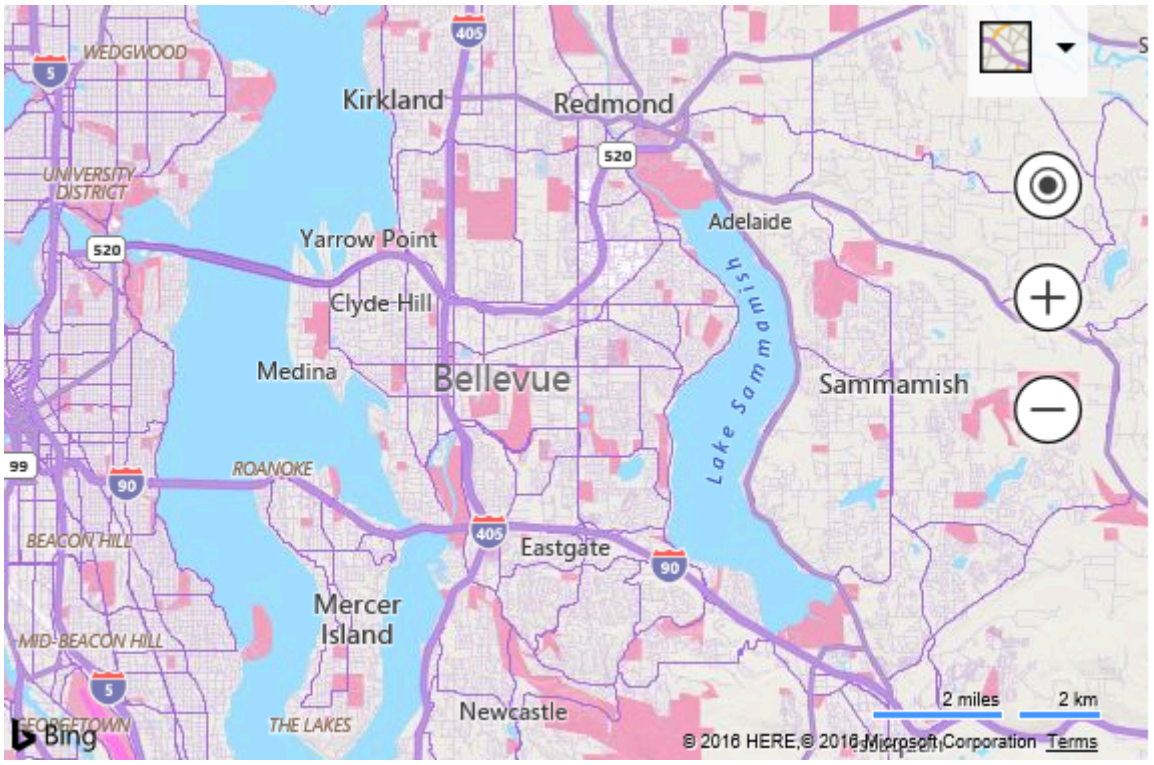
HTML

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
    <meta charset="utf-8" />
    <script type='text/javascript'>
        var map;

        var myStyle = {
            "elements": {
                "water": { "fillColor": "#a1e0ff" },
                "waterPoint": { "iconColor": "#a1e0ff" },
                "transportation": { "strokeColor": "#aa6de0" },
                "road": { "fillColor": "#b892db" },
                "railway": { "strokeColor": "#a495b2" },
                "structure": { "fillColor": "#ffffff" },
                "runway": { "fillColor": "#ff7fed" },
                "area": { "fillColor": "#f39ebd" },
                "political": { "borderStrokeColor": "#fe6850",
"borderOutlineColor": "#55ffff" },
                "point": { "iconColor": "#ffffff", "fillColor": "#FF6FA0",
"strokeColor": "#DB4680" },
                "transit": { "fillColor": "#AA6DE0" }
            },
            "version": "1.0"
        };

        function GetMap()
        {
            map = new Microsoft.Maps.Map('#myMap', {
                credentials: 'Your Bing Maps Key',
                customMapStyle: myStyle
            });
        }
    </script>
    <script type='text/javascript'
src='http://www.bing.com/api/maps/mapcontrol?callback=GetMap' async defer>
</script>
</head>
<body>
    <div id="myMap"></div>
</body>
</html>
```

Running this code will produce a map that looks like this:



> **Tip:** TypeScript definitions for the Bing Maps V8 Web Control do include definitions for custom style schema. You can access the Bing Maps TypeScript definitions here.

Custom Map Styles in the REST and Tile Services

Custom map styles can be used with the Bing Maps REST imagery service and when directly accessing Bing Maps tiles in a supported manner. To add a custom map style, a formatted string version of the style can be added as a URL parameter to the image/tile request. The URL parameter that can be added to the request is `&style=` or `&st=`.

The style string has the following format:

> [elementName]|[styleParam1]:[value];[styleParam2]:[value];

The above format is for a single element in the custom map style. You can combine multiple elements and the settings value by joining them with an underscore (_). The element name can either be the full element name (i.e. road) or the short form version (i.e. rd).

For example, take the following JSON style which colors water areas red and their labels green, makes roads blue, and sets the global land color to white:

```
JSON
```

```
{
    "elements": {
        "water": {
            "fillColor": "#FF0000",
            "labelColor": "#00FF00"
        },
        "road": { "fillColor": "#0000FF" }
    },
    "settings": { "landColor": "#FFFFFF" },
    "version": "1.0"
}
```

The string formatted version of this style looks like this (long form):

url

```
water|fillColor:FF0000;labelColor:00FF00_road|fillColor:0000FF_global|landCo
lor:FFFFFF
```

Here is the same style using the short form version:

url

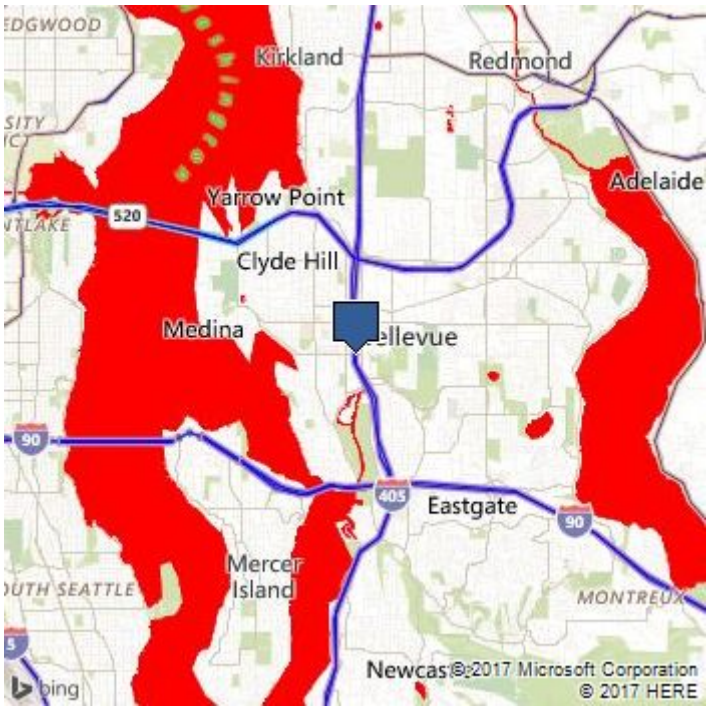
```
wt|fc:FF0000;lbc:00FF00_rd|fc:0000FF_g|landColor:FFFFFF
```

This can then be appended to a REST Static Image request or a tile URL. For example:

url

```
http://dev.virtualearth.net/REST/V1/Imagery/Map/Road/Bellevue%20Washington?
&key=
[YOUR_BING_MAPS_KEY]&st=wt|fc:FF0000;lbc:00FF00_rd|fc:0000FF_g|landColor:FFF
FFF
```

Here is the image this request would return:



If the style is too long for a URL, when using the REST imagery service, the style can be passed in using a POST request. The POST data object format is: `style=[Your custom style]`

Style Objects

A map style sheet consists primarily of entries and properties on those entries that you can override to customize the appearance.

Map style sheets can be created interactively using the Map Style Sheet Editor application.

Known Limitation

The following are some known limitations of custom map styles.

- Custom styles are not supported for all country/region maps. Custom maps styles require the use of vector map data. Some countries/regions have strict regulations around vector map data which prevents the Bing Maps team from using it to create custom map styles on the fly. This effects China, South Korea, and Japan currently. If you zoom into one of these countries/regions, you will see the default map tile style appear for these countries/regions while surrounding areas will use the custom map style.

Bing Maps V8

- If labels are styled, the map will go into `liteMode` as vector label styling is not currently supported.

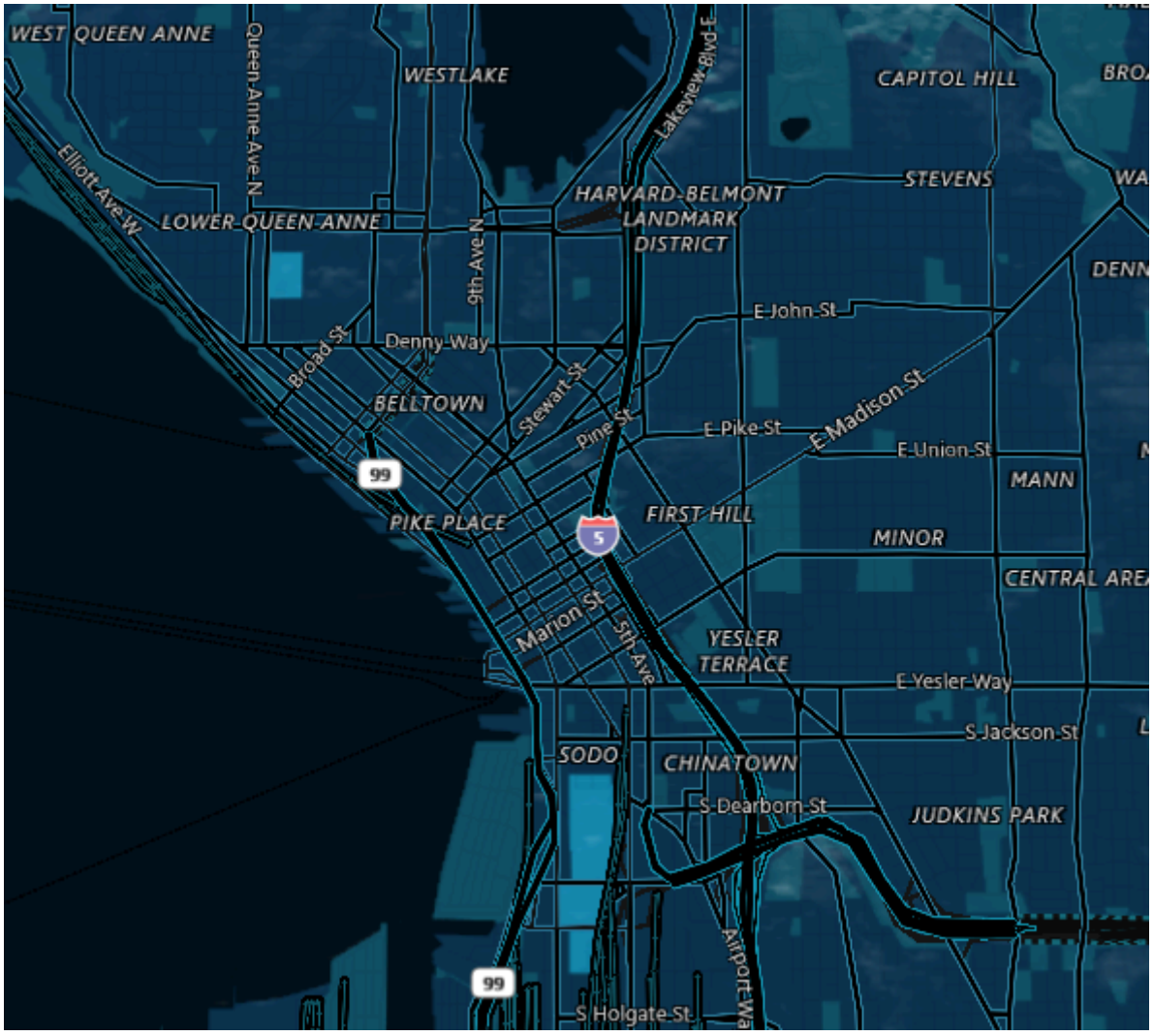
REST/Tile Services

- When using formatted string styles, do not end the style with a semi-colon (;).
- When an invalid style is passed to the REST service, a blue image is returned.

Custom Map Style Samples

Here are some sample custom map styles.

Midnight Commander Style



JSON

```
{
    "version": "1.0",
    "settings": {
        "landColor": "#0B334D"
```

```
    },
    "elements": {
        "mapElement": {
            "labelColor": "#FFFFFF",
            "labelOutlineColor": "#000000"
        },
        "political": {
            "borderStrokeColor": "#144B53",
            "borderOutlineColor": "#00000000"
        },
        "point": {
            "iconColor": "#0C4152",
            "fillColor": "#000000",
            "strokeColor": "#0C4152"
        },
        "transportation": {
            "strokeColor": "#000000",
            "fillColor": "#000000"
        },
        "highway": {
            "strokeColor": "#158399",
            "fillColor": "#000000"
        },
        "controlledAccessHighway": {
            "strokeColor": "#158399",
            "fillColor": "#000000"
        },
        "arterialRoad": {
            "strokeColor": "#157399",
            "fillColor": "#000000"
        },
        "majorRoad": {
            "strokeColor": "#157399",
            "fillColor": "#000000"
        },
        "railway": {
            "strokeColor": "#146474",
            "fillColor": "#000000"
        },
        "structure": {
            "fillColor": "#115166"
        },
        "water": {
            "fillColor": "#021019"
        },
        "area": {
            "fillColor": "#115166"
        }
    }
}
```

REST Style

```
```

url

```
me|lbc:ffffff;loc:000000_pl|bsc:144b53;boc:00000000_pt|ic:0c4152;fc:000000;sc:0c4152_trs|sc:000000;fc:000000_hg|sc:158399;fc:000000_cah|sc:158399;fc:000000_ard|sc:157399;fc:000000_mr|sc:157399;fc:000000_rl|sc:146474;fc:000000_str|fc:115166_wt|fc:021019_ar|fc:115166_g|lc:0b334d
```

Countries/Regions Only Style

JSON

```
{
    "version": "1.0",
    "elements": {
        "mapElement": {
            "labelVisible": false
        },
        "area": {
            "visible": false
        },
        "transportation": {
            "visible": false
```

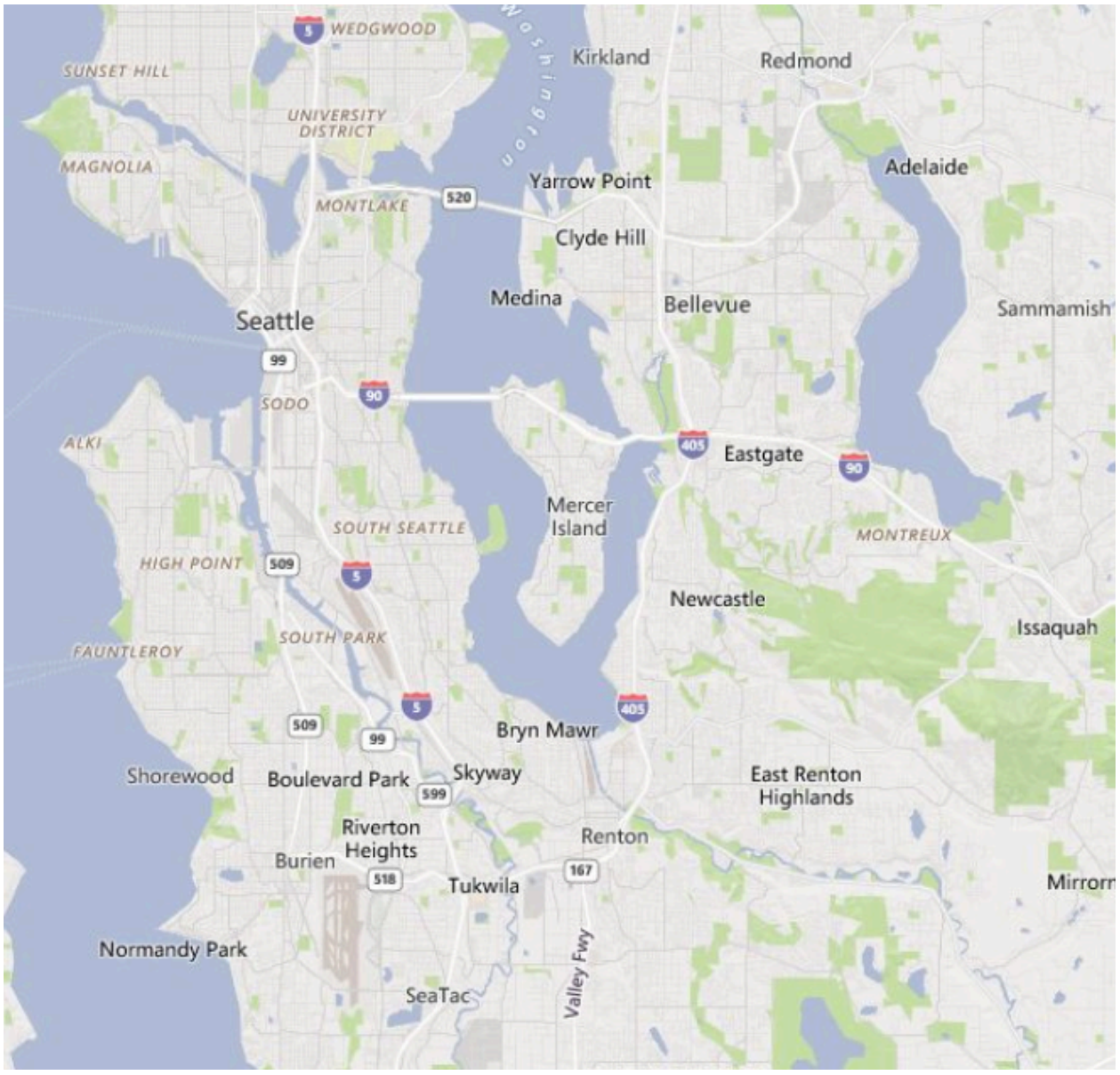
```
        },
        "countryRegion": {
            "borderStrokeColor": "#444444",
            "borderOutlineColor": "#00000000",
            "fillColor": "#888888",
            "visible": true
        },
        "adminDistrict": {
            "borderVisible": false
        },
        "water": {
            "fillColor": "#FFFFFF"
        },
        "point": {
            "visible": false
        }
    }
}
```

REST Style

url

```
me|lv:0_ar|v:0_trs|v:0_cr|bsc:444444;boc:00000000;fc:888888;v:1_ad|bv:0_wt|fc:ffffff_pt|v:0
```

Faded Map Style



JSON


```
{
    "version": "1.0",
    "settings": {
        "landColor": "#e7e6e5",
        "shadedReliefVisible": false
    },
    "elements": {
        "vegetation": {
            "fillColor": "#c5dea2"
        },
        "naturalPoint": {
            "visible": false,
            "labelVisible": false
        },
        "transportation": {
            "labelOutlineColor": "#ffffff",
            "fillColor": "#ffffff",
            "strokeColor": "#d7d6d5"
        },
        "water": {
```

```
            "fillColor": "#b1bdd6",
            "labelColor": "#ffffff",
            "labelOutlineColor": "#9aa9ca"
        },
        "structure": {
            "fillColor": "#d7d6d5"
        },
        "indigenousPeoplesReserve": {
            "visible": false
        },
        "military": {
            "visible": false
        }
    }
}
```

REST Style

url

```
vg|fc:c5dea2_np|v:0;lv:0_trs|loc:ffffff;fc:ffffff;sc:d7d6d5_wt|fc:b1bdd6;lbc
:ffffff;loc:9aa9ca_str|fc:d7d6d5_ipr|v:0_ima|v:0_g|lc:e7e6e5;srv:0
```

Related Resources

- Load Map with Custom Style Sample
- Simple Style Editor

Geocoding Japanese Addresses

Article • 05/22/2024

> **Note**
>
> **Bing Maps for Enterprise service retirement**
>
> Bing Maps for Enterprise is deprecated and will be retired. Free (Basic) account customers can continue to use Bing Maps for Enterprise services until June 30th, 2025. Enterprise account customers can continue to use Bing Maps for Enterprise services until June 30th, 2028. To avoid service disruptions, all implementations using Bing Maps for Enterprise REST APIs and SDKs will need to be updated to use Azure Maps by the retirement date that applies to your Bing Maps for Enterprise account type.
>
> Azure Maps is Microsoft's next-generation maps and geospatial services for developers. Azure Maps has many of the same features as Bing Maps for Enterprise, and more. To get started with Azure Maps, create a free Azure subscription and an Azure Maps account. For more information about azure Maps, see Azure Maps Documentation. For migration guidance, see Bing Maps Migration Overview.

Japanese geocoding is complex because addresses can be expressed using four different character sets – three native sets (Kanji, Hiragana, Katakana) as well as the Latin (western) alphabet. Also, Japan uses the Japanese address system which is different from the western address system. The Bing Maps REST Services and Bing Spatial Data Services offer flexibility to handle the custom needs of Japanese address geocoding and support these key features:

- **Use of Kanji and Kana (Katakana and Hiragana) character sets**: You can specify Japanese addresses in Kanji, Katakana or Hiragana or a combination of any of these. When you reverse-geocode, the response is returned in either Katakana or Hiragana. See **Japanese and Latin (Hepburn Romanization) Character Sets** below for more details
- **Use of Hepburn Romanization Latin character set and variants**: You can specify Japanese addresses using the Hepburn character set that uses Latin characters. Some variants are also supported. See the **Hepburn Romanization (Latin character set) and its variants** section below for more details.
- **Support for western and Japanese address systems**: You can specify a Japanese address using either the Japanese or western address system. See **Japanese and Western Address Systems** below for more details.
- **Support for addresses with mixed character sets**: You can specify a Japanese address using a combination of characters sets. For example, you can specify an address specified using both Kanji and Latin (Hepburn) characters.
- **Support for addresses that do not follow standard Japanese address conventions**: You can geocode non-standard Japanese addresses. See **Custom address support** below for more details.
- **Support for the historical Kyoto Toorina address system**: You can geocode addresses using the Kyoto Toorina address system.

The Bing Maps APIs are designed to look for and handle the complexity of Japanese geocoding, and will return results for a great variety of situations. However, note that there can be situations where due to the complexity, geocoding may not be successful.

Geocode using the Bing Maps REST Services Locations API

You can use the Locations (part of the Bing Maps REST Services) to geocode Japanese addresses. Each address component can be specified as a separate URL parameter or you can specify the entire address string.

> **Important**
>
> Japanese address values in native character sets (Kanji, Katakana, Hiragana) must be base-64 encoded UTF-8 strings.

Geocode using URL address parameters

If you know the parsed address, you can specify each component separately using the following URL template and the URL parameters in the table. This type of request will give the best result because the address string does not need to be parsed.

url

```
https://dev.virtualearth.net/REST/v1/Locations?countryRegion=countryRegion&adminDistrict=adminDistrict&locality=
```

```
locality&postalCode=postalCode&addressLine=addressLine&culture=ja&key=YourBingMapsKey
```

Address URL parameters with Japanese examples

The following table lists the address parameters that you can set in the URL

Expand table

URL parameter	Japanese address examples
CountryRegion	CountryRegion=JP
PostalCode	PostalCode=〒100-0000 PostalCode=100-0000 **Note**: The symbol 〒 is optional even though it is required by the Japanese addressing system.
AdminDistrict	AdminDistrict=東京都 : Tokyo prefecture AdminDistrict=北海道 : Hokkaido prefecture AdminDistrict=大阪府 : Osaka prefecture AdminDistrict=福岡県 : Other prefectures **Note**: It is acceptable to include Japan as part of the AdminDistrict by putting it in front of the prefecture. For example: AdminDistrict=日本、東京都.
Locality	Locality=港区 : Tokyo and other special designated cities Locality=西多摩郡 : Prefectures with gun Locality=那覇市 : Other municipals
AddressLine	港区2: Address within the locality
Culture [**REQUIRED**]	Culture=ja OR c=ja This parameter must be included in all Japanese geocoding requests.

Geocode using a single query string

If it's not possible or convenient to separate out the components of the address, you can specify a single address string in the request:

```
url

https://dev.virtualearth.net/REST/v1/Locations?q=addressString&culture=ja&key=YourBingMapsKey
```

Reverse-geocode

To reverse-geocode a Japanese address, specify the latitude and longitude of the location in the request.

```
url

https://dev.virtualearth.net/REST/v1/Locations/latitudeIinDegrees,longitudeInDegrees?key=YourBingMapsKey
```

Japanese Geocode and Reverse-Geocode URL Examples

The following examples show how to use these URLs to geocode Japanese addresses. Before trying out these examples, make sure you replace the placeholder YourBingMapsKey with your Bing Maps Key. These URLs will produce an XML response because the output parameter "o=xml" is specified. If you do not specify the output parameter, a JSON response is returned.

Expand table

Request type	Address values, strings or coordinates	Example URL
Geocode an address using URL address parameters	AdminDistrict = 東京都 Locality = 港区 AddressLine = 港南２－１６－３	https://dev.virtualearth.net/REST/v1/Locations?countryRegion=JP&adminDistrict=%E6%9D%B1%E4%BA%AC%E9%83%BD&locality=%e6%b8%af%e5%8c%ba&addressLine=%e6%b8
Geocode an address using a single address string	Query = 〒108-0075東京都港区港南２－１６－３	https://dev.virtualearth.net/REST/v1/Locations?query=%e3%80%92108%2d0075%e6%9d%b1%e4%ba%ac%e9%83%bd%e6%b8%af%e5%8c%ba%e6%b8%af%e5%8d%97%ef%bc%92
Geocode a postal code using URL address parameters	PostalCode = 108-0075	https://dev.virtualearth.net/REST/v1/Locations?countryRegion=JP&postalCode=108-0075&o=xml&key=YourBingMapsKey&c=ja
Geocode a postal code using a single address string	Query = 〒108-0075	https://dev.virtualearth.net/REST/v1/Locations?countryRegion=JP&postalCode=%e3%80%92108%2d0075&o=xml&key=YourBingMap
Reverse-geocode a latitude and longitude	Latitude=35 Longitude=139	https://dev.virtualearth.net/REST/v1/Locations/35,139?o=xml&key=YourBingMapsKey

Geocode Response

When you make a geocode request using the Locations API URLs and a location is found that matches the address, a response is returned with the latitude and longitude, a confidence and other information. The following is an example of an XML response returned by a Locations API geocode request. For more information about the fields returned in a Locations API response and the equivalent JSON response format, see Location Data.

Japanese address information in the response is returned using Japanese native character sets only (Kanji, Katakana, Hiragana). Latin character representations of Japanese addresses are not returned in the response.

XML

```
This XML file does not appear to have any style information associated with it. The document tree is shown below.
<Response xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="https://www.w3.org/2001/XMLSchema"
xmlns="https://schemas.microsoft.com/search/local/ws/rest/v1">
  <Copyright>
    Copyright © 2013 Microsoft and its suppliers. All rights reserved. This API cannot be accessed and the content and any
results may not be used, reproduced or transmitted in any manner without express written permission from Microsoft
Corporation.
  </Copyright>
  <BrandLogoUri>
    https://dev.virtualearth.net/Branding/logo_powered_by.png
  </BrandLogoUri>
  <StatusCode>200</StatusCode>
  <StatusDescription>OK</StatusDescription>
  <AuthenticationResultCode>ValidCredentials</AuthenticationResultCode>
  <TraceId>
    c6b9f2cdc5bf4145873fe1b60616600c
  </TraceId>
  <ResourceSets>
    <ResourceSet>
      <EstimatedTotal>1</EstimatedTotal>
      <Resources>
```

```
        <Location>
          <Name>108-0075 東京都港区港南２丁目１６ - ３</Name>
          <Point>
            <Latitude>35.6265754699707</Latitude>
            <Longitude>139.74099731445313</Longitude>
          </Point>
          <BoundingBox>
            <SouthLatitude>35.623880524118341</SouthLatitude>
            <WestLongitude>139.73768180735172</WestLongitude>
            <NorthLatitude>35.629270415823065</NorthLatitude>
            <EastLongitude>139.74431282155447</EastLongitude>
          </BoundingBox>
          <EntityType>Address</EntityType>
          <Address>
            <AddressLine>港南２丁目１６ - ３</AddressLine>
            <AdminDistrict>東京都</AdminDistrict>
            <CountryRegion>日本</CountryRegion>
            <FormattedAddress>108-0075 東京都港区港南２丁目１６ - ３</FormattedAddress>
            <Locality>港区</Locality>
            <PostalCode>108-0075</PostalCode>
          </Address>
          <Confidence>High</Confidence>
          <MatchCode>Good</MatchCode>
          <GeocodePoint>
            <Latitude>35.6265754699707</Latitude>
            <Longitude>139.74099731445313</Longitude>
            <CalculationMethod>Rooftop</CalculationMethod>
            <UsageType>Display</UsageType>
          </GeocodePoint>
        </Location>
      </Resources>
    </ResourceSet>
  </ResourceSets>
</Response>
```


Geocoding with Bing Spatial Data Services

You can geocode a set of addresses using the Bing Spatial Data Services using the Geocode Dataflow API. Like the Locations API, the Geocode Dataflow API allows you to geocode a Japanese address by identifying its components or by specifying a single address string. The Geocode Dataflow data schema includes the following input values.

Values

```
Id
GeocodeRequest/Culture
GeocodeRequest/Query
GeocodeRequest/Address/AddressLine
GeocodeRequest/Address/AdminDistrict
GeocodeRequest/Address/CountryRegion
GeocodeRequest/Address/AdminDistrict2
GeocodeRequest/Address/FormattedAddress
GeocodeRequest/Address/Locality
GeocodeRequest/Address/PostalCode
GeocodeRequest/Address/PostalTown
GeocodeRequest/ConfidenceFilter/MinimumConfidence
```

To specify individual components, use the same address definitions as defined above for the Locations API (such as GeocodeRequest/Address/AddressLine and GeocodeRequest/Address/AdminDistrict). To specify a single address query string use GeocodeRequest/Query. For more information about this data schema and how to use the Geocode Dataflow, see Geocode Dataflow API, Data Schema v2.0, and Walkthrough.

Japanese and Western Address Systems

The Japanese address system formats addresses in the opposite order of western addresses. In the western address system, you begin with the address details and then proceed to the larger categories such as city and state and postal code. In the Japanese system, the larger categories are listed first and finish with the address details. The Bing Maps REST Services and the Bing Spatial Data Services support geocoding for addresses that use either the Japanese or western address system for all character sets. For example, you can specify an address using Kanji in the western address system or you can specify an address in Latin characters using the Japanese address system and the API will detect the order. For best results, specify the values in the exact order shown below for each system.

Japanese Address System

(postcode}{prefecture){city/municipality/ward}{location in city/municipality/ward}{address details)

Example: 〒108-0075東京都港区港南2-16-3

Using Latin characters: 108-0075 Tokyo-to Minato-ku Konan 2-16-3

Western Address System

(address details)(location in city/municipality/ward) (city/municipal/ward)(preflecture)(postcode){country}

Example: 2-16-3 Konan Minato-ku, Tokyo 108-0075 Japan

Japanese and Latin (Hepburn/Hebon) Character Sets

When you geocode a Japanese address, you can specify the address using any of the native character sets -- Kanji, Hiragana, Katakana – as well as Hepburn Romanization. Hepburn Romanization, also known as Hebon, is a system for writing the Japanese language using the Latin alphabet.

You can also geocode an address that uses more than one characters set. For example, you can specify different parts of an address in Kanji and the Latin Hepburn system or Kanji and Katakana and still geocode the address. The following example shows an address written using Kanji and Hepburn Romanization.

Kanji + Heburn Romanization: 東京都Minato-ku Kounan 2-16-3

Kanji + Katakana: 東京都ミナト区

Note that when you reverse-geocode, the address is returned in the native Japanese character sets (Kanji, Katakana and Hiragana) only. Hepburn Romanization (Latin characters) are not supported for reverse-geocoding.

Hepburn (Hebon) Romanization (Latin character set) and its variants

Many of the Hepburn Romanization variants can be used when you geocode a Japanese address. Examples of supported variants include:

- Using suffixes that come from phonetic sounds. The following example contains the suffixes “-ku” and “-to” which are suffixes that come from the phonetic sounds that mean ward and prefecture respectively.

  2-16-3 Kounan Minato-ku, Tokyo-to 108-0075 Japan

- Using multiple forms of Japanese Hepburn spelling. For example, you can specify “Konan” instead of the Japanese phonetic spelling “Kounan”.

  2-16-3 Konan, Minato Ward, Tokyo-to, Japan

- Using special Unicode characters that are part of Hepburn such as a Latin character “o” with macron ō:

  2-16-3 Kōnan, Minato-ku, Tōkyō, Japan

Custom Address Support

You can also geocode custom or unofficial addresses. For example, if an address contains additional information that is not part of the official address, you may still be able to geocode the address.

Address alone: 長野県長野市南長野

Address with extra value: 長野県長野市大字南長野

Note that while the Bing Maps geocoding APIs are designed to try to handle custom addresses, there may be some custom addresses that are not successfully geocoded.

Geospatial Endpoint Service

Article • 05/22/2024

> **Note**
>
> **Bing Maps for Enterprise service retirement**
>
> Bing Maps for Enterprise is deprecated and will be retired. Free (Basic) account customers can continue to use Bing Maps for Enterprise services until June 30th, 2025. Enterprise account customers can continue to use Bing Maps for Enterprise services until June 30th, 2028. To avoid service disruptions, all implementations using Bing Maps for Enterprise REST APIs and SDKs will need to be updated to use Azure Maps by the retirement date that applies to your Bing Maps for Enterprise account type.
>
> Azure Maps is Microsoft's next-generation maps and geospatial services for developers. Azure Maps has many of the same features as Bing Maps for Enterprise, and more. To get started with Azure Maps, create a free Azure subscription and an Azure Maps account. For more information about azure Maps, see Azure Maps Documentation. For migration guidance, see Bing Maps Migration Overview.

The Geospatial Endpoint Service is a REST service that provides information about Geospatial Platform services for the language and geographical region you specify. The service information includes available service endpoints and language support for these endpoints. Disputed geographical areas and embargoed countries or regions that do not have any service support are also identified.

This documentation does not explain how to use the service endpoints returned in the response.

Request URLs

Use the following URLs to make a Geospatial Endpoint Service request. Options are provided for requests with and without specific geographical coordinates.

Get the service information for the language and region specified.

```
url
```

```
http://dev.virtualearth.net/REST/V1/GeospatialEndpoint/language/userRegion?
```

```
key={BingMapsKey}
```

Get the service information for the language, region and location coordinates specified.

The latitude and longitude coordinates are reverse-geocoded to determine the location. If this location corresponds to a non-disputed country or region, then this location overrides the userRegion value in the request. However, if the coordinates are in a disputed country or region, then the userRegion in the request is used. For example, if the coordinates represent a disputed area along the border of India and China, and userRegion is set to IN (India) in the request and the language is set to hi-in (Hindi), then IN services for Hindi are returned. In the same example, if the userRegion is set to CN (China) and the language is set to zh-hans (Simplified Chinese), then CN services for Simplified Chinese are returned in the response.

url

```
http://dev.virtualearth.net/REST/V1/GeospatialEndpoint/language/userRegion/latitude,longitude?key={BingMapsKey}
```

Parameters

Expand table

Parameter	Alias	Description	Values
language		The preferred language.	An IETF language code, that includes the language and region code subtags, such as en-us or zh-hans. **Example**: en-us This code represents English as it is spoken in the United States. **Example**: zh-hans This code represents Simplified Chinese.
userRegion		A country or region.	An ISO 3166-1 alpha-2 region code, such as US, IN, and CN. **Example**: US

Parameter	Alias	Description	Values
			This code represents the United States.
latitude, longitude		A location on the Earth. These coordinates are reverse-geocoded to determine the country or region. If this location is in a disputed area, then the userRegion parameter value is used to determine the corresponding region.	The latitude and longitude of the user's location in degrees. **Example**: 33.977531,75.726013
output	o	**Optional**. The output format for the response.	One of the following values: - json **[default]** - xml **Example**: o=xml
key		Your Bing Maps Key.	The GUID value that represents a Bing Maps Key.

Response

The response returns the following information:

- Whether this is a politically disputed area, such as an area claimed by more than one country/region.
- Whether services are available in the user's region.
- A list of available geospatial services including endpoints and language support for each service.

Using the Endpoints with Parameters

It is important to note that the endpoints returned in the response are not typically URLs that you can execute. They give the base URL to which you can add parameters, such as a quad key for an image, route waypoints or an address to geocode. This documentation does not cover the parameters specific to each service.

For example, a common endpoint for geocoding is `dev.virtualearth.net\REST\v1\Locations`. However, this URL does not return any results

without adding parameters. The following example shows how to geocode an address by providing address information and a Bing Maps Key.

url

```
http://dev.virtualearth.net/REST/v1/Locations?
q=1%20Microsoft%20Way%20Redmond%20WA%2098052&o=xml&key=YourBingMapsKey
```

Response Fields

The following example shows the general structure for the JSON response. You can also request an XML response. Example requests and JSON and XML responses are provided in the Examples section.

JSON

```
{
   "authenticationResultCode":"ValidCredentials",

"brandLogoUri":"http:\/\/dev.virtualearth.net\/Branding\/logo_powered_by.png
",
   "copyright":"Copyright © 2013 Microsoft and its suppliers. All rights
reserved. This API cannot be accessed and the content and any results may
not be used, reproduced or transmitted in any manner without express written
permission from Microsoft Corporation.",
   "resourceSets":[
      {
         "estimatedTotal":1,
         "resources":[
            {

"__type":"GeospatialEndpoint:http:\/\/schemas.microsoft.com\/search\/local\/
ws\/rest\/v1",
               "isDisputedArea":false,
               "isSupported":true,
               "ur":"FR",
               "services":[
                  {
                     "endpoint":"ecn.dynamic.t{0-
3}.tiles.virtualearth.net\/comp\/ch\/{quadkey}?mkt=
{market}&it=G,VE,BX,L,LA&shading=hill&n=z&ur={userregion}",
                     "fallbackLanguage":null,
                     "languageSupported":true,
                     "serviceName":"MapTiles"
                  },
                  {
                     "endpoint":"dev.virtualearth.net\/REST\/v1\/Locations",
                     "fallbackLanguage":null,
                     "languageSupported":true,
                     "serviceName":"Geocode"
```

```
                    },
                    {
                        "endpoint":"dev.virtualearth.net\/REST\/v1\/Routes",
                        "fallbackLanguage":null,
                        "languageSupported":true,
                        "serviceName":"Route"
                    }
                ]
            }
        ]
    }
    ],
    "statusCode":200,
    "statusDescription":null,
    "traceId":"961e9c1e63e64bd4aa7d140ee4e05697"
}
```


General Fields

Expand table

JSON	XML	Type	Description
isDisputedArea	IsDisputedArea	Boolean	Specifies if this area in the request is claimed by more than one country/region. For example, many areas along the border of India and China are disputed areas. Even though an area is disputed area, it may still be supported by Geospatial Platform services.
isSupported	IsSupported	Boolean	Specifies if Geospatial Platform services are available in the country or region. Microsoft does not support services in embargoed areas. For example, if you request Geospatial Platform Service information for Cuba (CU), isSupported is set to true, and no service information is returned.
ur (user region)	UR (user region)	string	The country or region that was used to determine service support. If you specified a latitude and longitude in the request that is in a non-disputed country or region, this country or region is returned in the response. Please see the section Region Localities below for more details on the `User Region Codes` and `Culture Codes`.
services	Services	array	Information for each geospatial service that is available in the country or region and language

JSON	XML	Type	Description
			specified in the request. See the **Service Fields** table for the information provided for each service. For a list of available services, see Supported Services.

Region Localities

In order to get this label to appear, the user region of the map has to be set to one of following `User Region Codes`.

Expand table

User Region Code	Official Short Form	Culture Code
IL	Israel	He-il
KR	Korea	Ko-kr
PK	Pakistan	pa-pk
IN	India	En-in
CN	China	zh-cn
AR	Argentina	es-AR
BH	Bahrain	ar-BH
EG	Egypt	ar-EG
JO	Jordon	ar-JO
SA	Saudi Arabia	ar-SA
KW	Kuwait	ar-KW
OM	Oman	ar-OM
QA	Qatar	ar-QA
MA	Morocco	ar-MA
AE	UAE	ar-AE

Service Fields

Expand table

JSON	XML	Type	Description
endpoint	Endpoint	URL	The URL service endpoint to use in this region. Note that to use the service, you must typically add parameters specific to the service. These parameters are not described in this documentation.
languageSupported	LanguageSupported	Boolean	Set to true if the service supports the language in the request for the region. If the language is supported, then the service endpoint will return responses using this language. If it is not supported, then the service will use the fallback language.
fallbackLanguage	FallbackLanguage	string	Specifies the secondary or fallback language in this region or country. If the requested language is not supported and a fallback language is not available, United States English (en-us) is used.
serviceName	ServiceName	string	An abbreviated name for the service. See Supported Services for a list of available services.

Response Container Fields

Expand table

JSON	XML	Type	Description
statusCode	StatusCode	integer	The HTTP Status code for the request.
statusDescription	StatusDescription	string	A description of the HTTP status code.
authenticationResultCode	AuthenticationResultCode	One of the following values: ValidCredentials	A status code that offers additional information about authentication success or failure.

JSON	XML	Type	Description
		InvalidCredentials CredentialsExpired NotAuthorized NoCredentials None	
traceId	TraceId	string	A unique identifier for the request.
copyright	Copyright	string	A copyright notice.
brandLogoUri	BrandLogoUri	string	A URL that references a brand image to support contractual branding requirements.
resourceSets	ResourceSets	collection	A collection of ResourceSet objects. A ResourceSet is a container of Resources returned by the request. For more information, see the ResourceSet section below.
estimatedTotal	EstimatedTotal	long	An estimate of the total number of resources in the ResourceSet.
resources	Resources	collection	A collection of one or more resources. The resources that are returned depend on the request. Information about resources is provided in the API reference for each

JSON	XML	Type	Description
			Bing Maps REST Services API.
errorDetails	ErrorDetails	string[]	A collection of error descriptions. For example, ErrorDetails can identify parameter values that are not valid or missing.

Supported Services

The following table describes the services that are supported by the Geospatial Endpoint Service. Information about one or more of these services may be returned in the response.

Expand table

Service Type	Response Service Name	Description
Map Imagery	MapTiles	Returns tiles from one of the following map tile services depending on the user region and language specified in the request. - On Demand Tile Service: Returns map tiles that are created on the server. This service endpoint will be provided when the user is in a geopolitically sensitive area, such as a disputed area or a country associated with such an area. - Pre-rendered Tile Service: Returns map tiles that have been pre-rendered during offline processing. This service endpoint will be provided when the user is not in a geopolitically sensitive area.
Map Imagery	TrafficTiles	Returns map tiles that show traffic flow overlays.
Map Imagery	StaticMapsB2B	Static Map API (Bing Maps REST Services): Returns static maps for the parameters that you specify. For parameter descriptions, see the documentation. **Note**: You must replace the *dev.virtualearth.net* endpoint in the documentation with the endpoint provided in the response.

Service Type	Response Service Name	Description
Map Metadata	MetadataB2B	Map Imagery Metadata API (Bing Maps REST Services): Returns map metadata for the parameters that you specify. For parameter descriptions, see the documentation. **Note**: You must use the endpoint provided in the response to replace the public dev.virtualearth.net endpoint in the documentation.
Geocode	Geocode	Locations API (Bing Maps REST Services): Returns geocoded or reverse-geocoded location information for the parameters you specify. For parameter descriptions, see the documentation. **Note**: You must replace the *dev.virtualearth.net* endpoint in the documentation with the endpoint provided in the response.
Route	Route	Routes API (Bing Maps REST Services): Returns route information for the parameters you specify. For parameter descriptions, see the documentation. **Note**: You must use the endpoint provided in the response to replace the public dev.virtualearth.net endpoint in the documentation.
Image URL	BingLogo	Returns the attribution logo to display with Bing Maps tiles that do not include Nokia data.
Image URL	CombinedLogo	Returns the attribution logo to display with Bing Maps tiles that include Nokia data.

Examples

Typical Service support example

Language: **fr-fr (French)**, userRegion: **FR (France)**

url

```
http://dev.virtualearth.net/REST/V1/GeospatialEndpoint/fr-fr/FR?key=
{BingMapsKey}
```

JSON

```
{
   "authenticationResultCode":"ValidCredentials",
```

```
   "brandLogoUri":"http:\/\/dev.virtualearth.net\/Branding\/logo_powered_by.
png",
   "copyright":"Copyright © 2013 Microsoft and its suppliers. All rights res
erved. This API cannot be accessed and the content and any results may not b
e used, reproduced or transmitted in any manner without express written perm
ission from Microsoft Corporation.",
   "resourceSets":[
      {
         "estimatedTotal":1,
         "resources":[
            {
               "__type":"GeospatialEndpointResponse:http:\/\/schemas.microso
ft.com\/search\/local\/ws\/rest\/v1",
               "isDisputedArea":false,
               "isSupported":true,
               "services":[
                  {
                     "endpoint":"ecn.dynamic.t{0-
3}.tiles.virtualearth.net\/comp\/ch\/{quadkey}?mkt=
{market}&it=G,VE,BX,L,LA&shading=hill&n=z&ur={userregion}",
                     "fallbackLanguage":"fr",
                     "languageSupported":false,
                     "serviceName":"MapTiles"
                  },
                  {
                     "endpoint":"ecn.t{0-
7}.tiles.virtualearth.net\/tiles\/dp\/content?mkt={market}&p=tf&a=
{quadkey}&n=z",
                     "fallbackLanguage":"fr",
                     "languageSupported":false,
                     "serviceName":"TrafficTiles"
                  },
                  {
                     "endpoint":"dev.virtualearth.net\/REST\/V1\/Imagery\/Ma
p",
                     "fallbackLanguage":"fr",
                     "languageSupported":false,
                     "serviceName":"StaticMapsB2B"
                  },
                  {
                     "endpoint":"dev.virtualearth.net\/REST\/V1\/Imagery\/Me
tadata",
                     "fallbackLanguage":"fr",
                     "languageSupported":false,
                     "serviceName":"MetadataB2B"
                  },
                  {
                     "endpoint":"dev.virtualearth.net\/REST\/v1\/Locations",
                     "fallbackLanguage":"fr",
                     "languageSupported":false,
                     "serviceName":"Geocode"
                  },
                  {
                     "endpoint":"dev.virtualearth.net\/REST\/v1\/Routes",
                     "fallbackLanguage":"fr",
```

```
                    "languageSupported":false,
                    "serviceName":"Route"
                },
                {
                    "endpoint":"ecn.dev.virtualearth.net\/Branding\/logo_po
wered_by.png",
                    "fallbackLanguage":"fr",
                    "languageSupported":false,
                    "serviceName":"BingLogo"
                },
                {
                    "endpoint":"ecn.dev.virtualearth.net\/Branding\/logo_po
wered_by.png",
                    "fallbackLanguage":"fr",
                    "languageSupported":false,
                    "serviceName":"CombinedLogo"
                }
            ],
            "ur":"FR"
        }
    ]
  }
  ],
  "statusCode":200,
  "statusDescription":"OK",
  "traceId":"4bcd1b7077e84c049cfbbcda3632c22a"
}
```


You would receive the following JSON response if the output=xml parameter was set in this example.

XML


```
<?xml version="1.0" encoding="utf-8"?>
<Response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.microsoft.com/search/local/ws/rest/v1">
  <Copyright>Copyright © 2013 Microsoft and its suppliers. All rights
reserved. This API cannot be accessed and the content and any results may
not be used, reproduced or transmitted in any manner without express written
permission from Microsoft Corporation.</Copyright>

<BrandLogoUri>http://dev.virtualearth.net/Branding/logo_powered_by.png</Bran
dLogoUri>
  <StatusCode>200</StatusCode>
  <StatusDescription>OK</StatusDescription>
  <AuthenticationResultCode>ValidCredentials</AuthenticationResultCode>
  <TraceId>00e37177aa52425083eae9fc4874fceb</TraceId>
  <ResourceSets>
    <ResourceSet>
      <EstimatedTotal>1</EstimatedTotal>
      <Resources>
        <Resource xsi:type="GeospatialEndpointResponse">
```

```
          <UR>FR</UR>
          <IsDisputedArea>false</IsDisputedArea>
          <IsSupported>true</IsSupported>
          <Services>
            <ServiceInfo>
              <ServiceName>MapTiles</ServiceName>
              <Endpoint>ecn.dynamic.t{0-
3}.tiles.virtualearth.net/comp/ch/{quadkey}?mkt=
{market}&amp;it=G,VE,BX,L,LA&amp;shading=hill&amp;n=z&amp;ur={userregion}
</Endpoint>
              <LanguageSupported>false</LanguageSupported>
              <FallbackLanguageId>fr</FallbackLanguageId>
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>TrafficTiles</ServiceName>
              <Endpoint>ecn.t{0-7}.tiles.virtualearth.net/tiles/dp/content?
mkt={market}&amp;p=tf&amp;a={quadkey}&amp;n=z</Endpoint>
              <LanguageSupported>false</LanguageSupported>
              <FallbackLanguageId>fr</FallbackLanguageId>
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>StaticMapsB2B</ServiceName>
              <Endpoint>dev.virtualearth.net/REST/V1/Imagery/Map</Endpoint>
              <LanguageSupported>false</LanguageSupported>
              <FallbackLanguageId>fr</FallbackLanguageId>
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>MetadataB2B</ServiceName>

<Endpoint>dev.virtualearth.net/REST/V1/Imagery/Metadata</Endpoint>
              <LanguageSupported>false</LanguageSupported>
              <FallbackLanguageId>fr</FallbackLanguageId>
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>Geocode</ServiceName>
              <Endpoint>dev.virtualearth.net/REST/v1/Locations</Endpoint>
              <LanguageSupported>false</LanguageSupported>
              <FallbackLanguageId>fr</FallbackLanguageId>
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>Route</ServiceName>
              <Endpoint>dev.virtualearth.net/REST/v1/Routes</Endpoint>
              <LanguageSupported>false</LanguageSupported>
              <FallbackLanguageId>fr</FallbackLanguageId>
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>BingLogo</ServiceName>

<Endpoint>ecn.dev.virtualearth.net/Branding/logo_powered_by.png</Endpoint>
              <LanguageSupported>false</LanguageSupported>
              <FallbackLanguageId>fr</FallbackLanguageId>
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>CombinedLogo</ServiceName>
```

```
<Endpoint>ecn.dev.virtualearth.net/Branding/logo_powered_by.png</Endpoint>
              <LanguageSupported>false</LanguageSupported>
              <FallbackLanguageId>fr</FallbackLanguageId>
            </ServiceInfo>
          </Services>
        </Resource>
      </Resources>
    </ResourceSet>
  </ResourceSets>
</Response>
```

No Service support example

Language: **en-us (English)**, userRegion: **CU (Cuba)**

url

```
http://dev.virtualearth.net/REST/V1/GeospatialEndpoint/en-us/cu?key=
{BingMapsKey}
```

JSON


```
{
   "authenticationResultCode":"ValidCredentials",
   "brandLogoUri":"http:\/\/dev.virtualearth.net\/Branding\/logo_powered_by.
png",
   "copyright":"Copyright © 2013 Microsoft and its suppliers. All rights res
erved. This API cannot be accessed and the content and any results may not b
e used, reproduced or transmitted in any manner without express written perm
ission from Microsoft Corporation.",
   "resourceSets":[
      {
         "estimatedTotal":1,
         "resources":[
            {
               "__type":"GeospatialEndpointResponse:http:\/\/schemas.microso
ft.com\/search\/local\/ws\/rest\/v1",
               "isDisputedArea":false,
               "isSupported":false,
               "services":null,
               "ur":"CU"
            }
         ]
      }
   ],
   "statusCode":200,
   "statusDescription":"OK",
   "traceId":"eaaa2cd43ebb48a597baa601eb2437df
}
```

You would receive the following JSON response if the output=xml parameter was set in this example. Note that the **Services** array does not appear.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<Response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.microsoft.com/search/local/ws/rest/v1">
  <Copyright>Copyright © 2013 Microsoft and its suppliers. All rights
reserved. This API cannot be accessed and the content and any results may
not be used, reproduced or transmitted in any manner without express written
permission from Microsoft Corporation.</Copyright>

<BrandLogoUri>http://dev.virtualearth.net/Branding/logo_powered_by.png</Bran
dLogoUri>
  <StatusCode>200</StatusCode>
  <StatusDescription>OK</StatusDescription>
  <AuthenticationResultCode>ValidCredentials</AuthenticationResultCode>
  <TraceId>2559b704c7e34392b95bfb99437060adTraceId>
  <ResourceSets>
    <ResourceSet>
      <EstimatedTotal>1</EstimatedTotal>
      <Resources>
        <Resource xsi:type="GeospatialEndpointResponse">
          <UR>CU</UR>
          <IsDisputedArea>false</IsDisputedArea>
          <IsSupported>false</IsSupported>
        </Resource>
      </Resources>
    </ResourceSet>
  </ResourceSets>
</Response>
```

Disputed area example

Language: **zh-hans (Simplified Chinese)**, userRegion: **CN (China)**, Coordinates: **Disputed Area between India and China**

> ⓘ **Note**
>
> The response shows `CN` as the region because the latitude and longitude are in a disputed area.

url

```
http://dev.virtualearth.net/REST/V1/GeospatialEndpoint/zh-
```

```
hans/cn/32.750323,79.376221?key={BingMapsKey}
```

JSON


```
{
   "authenticationResultCode":"ValidCredentials",
   "brandLogoUri":"http:\/\/dev.virtualearth.net\/Branding\/logo_powered_by.
png",
   "copyright":"Copyright © 2013 Microsoft and its suppliers. All rights res
erved. This API cannot be accessed and the content and any results may not b
e used, reproduced or transmitted in any manner without express written perm
ission from Microsoft Corporation.",
   "resourceSets":[
      {
         "estimatedTotal":1,
         "resources":[
            {
               "__type":"GeospatialEndpointResponse:http:\/\/schemas.microso
ft.com\/search\/local\/ws\/rest\/v1",
               "isDisputedArea":true,
               "isSupported":true,
               "services":[
                  {
                     "endpoint":"r{0-
3}.tiles.ditu.live.com\/tiles\/r{quadkey}.png?g={generation}",
                     "fallbackLanguage":null,
                     "languageSupported":true,
                     "serviceName":"RoadWithLabels"
                  },
                  {
                     "endpoint":"r{0-
3}.tiles.ditu.live.com\/tiles\/r{quadkey}.png?g={generation}",
                     "fallbackLanguage":null,
                     "languageSupported":true,
                     "serviceName":"MapTiles"
                  },
                  {
                     "endpoint":"traftile.mapabc.com\/trafficengine\/traffic
tile?Key={quadkey}",
                     "fallbackLanguage":null,
                     "languageSupported":true,
                     "serviceName":"TrafficTiles"
                  },
                  {
                     "endpoint":"dev.ditu.live.com\/REST\/V1\/Imagery\/Map",
                     "fallbackLanguage":null,
                     "languageSupported":true,
                     "serviceName":"StaticMapsB2B"
                  },
                  {
                     "endpoint":"dev.ditu.live.com\/REST\/V1\/Locations",
                     "fallbackLanguage":null,
                     "languageSupported":true,
```

```
                    "serviceName":"Geocode"
                },
                {
                    "endpoint":"dev.ditu.live.com\/REST\/V1\/Routes",
                    "fallbackLanguage":null,
                    "languageSupported":true,
                    "serviceName":"Route"
                }
            ],
            "ur":"CN"
        }
      ]
    }
  ],
  "statusCode":200,
  "statusDescription":"OK",
  "traceId":"711c2b03cb1744f2849eddfd7a7650a3"
}
```


You would receive the following JSON response if the output=xml parameter was set in this example.

XML


```
<?xml version="1.0" encoding="utf-8"?>
<Response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.microsoft.com/search/local/ws/rest/v1">
  <Copyright>Copyright © 2013 Microsoft and its suppliers. All rights
reserved. This API cannot be accessed and the content and any results may
not be used, reproduced or transmitted in any manner without express written
permission from Microsoft Corporation.</Copyright>

<BrandLogoUri>http://dev.virtualearth.net/Branding/logo_powered_by.png</Bran
dLogoUri>
  <StatusCode>200</StatusCode>
  <StatusDescription>OK</StatusDescription>
  <AuthenticationResultCode>ValidCredentials</AuthenticationResultCode>
  <TraceId>94994c4c84da4c40a32e55c0c2a76b06</TraceId>
  <ResourceSets>
    <ResourceSet>
      <EstimatedTotal>1</EstimatedTotal>
      <Resources>
        <Resource xsi:type="GeospatialEndpointResponse">
          <UR>CN</UR>
          <IsDisputedArea>true</IsDisputedArea>
          <IsSupported>true</IsSupported>
          <Services>
            <ServiceInfo>
              <ServiceName>RoadWithLabels</ServiceName>
              <Endpoint>r{0-3}.tiles.ditu.live.com/tiles/r{quadkey}.png?g=
{generation}</Endpoint>
              <LanguageSupported>true</LanguageSupported>
```

```
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>MapTiles</ServiceName>
              <Endpoint>r{0-3}.tiles.ditu.live.com/tiles/r{quadkey}.png?g=
{generation}</Endpoint>
              <LanguageSupported>true</LanguageSupported>
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>TrafficTiles</ServiceName>
              <Endpoint>traftile.mapabc.com/trafficengine/traffictile?Key=
{quadkey}</Endpoint>
              <LanguageSupported>true</LanguageSupported>
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>StaticMapsB2B</ServiceName>
              <Endpoint>dev.ditu.live.com/REST/V1/Imagery/Map</Endpoint>
              <LanguageSupported>true</LanguageSupported>
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>Geocode</ServiceName>
              <Endpoint>dev.ditu.live.com/REST/V1/Locations</Endpoint>
              <LanguageSupported>true</LanguageSupported>
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>Route</ServiceName>
              <Endpoint>dev.ditu.live.com/REST/V1/Routes</Endpoint>
              <LanguageSupported>true</LanguageSupported>
            </ServiceInfo>
          </Services>
        </Resource>
      </Resources>
    </ResourceSet>
  </ResourceSets>
</Response>
```

Language: **hi-in (Hindi)**, userRegion: **IN (India)**, Coordinates: **Disputed Area between India and China**

> ⓘ Note
>
> The response shows `IN` as the region because the latitude and longitude coordinates are in a disputed area.

url

```
http://dev.virtualearth.net/REST/V1/GeospatialEndpoint/hi-
in/in/32.750323,79.376221?key={BingMapsKey}
```

JSON

```
{
   "authenticationResultCode":"ValidCredentials",
   "brandLogoUri":"http:\/\/dev.virtualearth.net\/Branding\/logo_powered_by.
png",
   "copyright":"Copyright © 2013 Microsoft and its suppliers. All rights res
erved. This API cannot be accessed and the content and any results may not b
e used, reproduced or transmitted in any manner without express written perm
ission from Microsoft Corporation.",
   "resourceSets":[
      {
         "estimatedTotal":1,
         "resources":[
            {
               "__type":"GeospatialEndpointResponse:http:\/\/schemas.microso
ft.com\/search\/local\/ws\/rest\/v1",
               "isDisputedArea":true,
               "isSupported":true,
               "services":[
                  {
                     "endpoint":"ecn.dynamic.t{0-
3}.tiles.virtualearth.net\/comp\/ch\/{quadkey}?mkt=en-
us&it=G,L&shading=hil&n=z",
                     "fallbackLanguage":"en-US",
                     "languageSupported":false,
                     "serviceName":"RoadWithLabels"
                  },
                  {
                     "endpoint":"ecn.dynamic.t{0-
3}.tiles.virtualearth.net\/comp\/ch\/{quadkey}?mkt=en-us&it=A,G,L&n=z",
                     "fallbackLanguage":"en-US",
                     "languageSupported":false,
                     "serviceName":"AerialWithLabels"
                  },
                  {
                     "endpoint":"ecn.dynamic.t{0-
3}.tiles.virtualearth.net\/comp\/ch\/{quadkey}?mkt=en-
us&it=G&shading=hill&n=z",
                     "fallbackLanguage":"en-US",
                     "languageSupported":false,
                     "serviceName":"RoadWithoutLabels"
                  },
                  {
                     "endpoint":"ecn.dynamic.t{0-
3}.tiles.virtualearth.net\/comp\/ch\/{quadkey}?mkt=en-us&it=A,G&n=z",
                     "fallbackLanguage":"en-US",
                     "languageSupported":false,
                     "serviceName":"AerialWithoutLabels"
                  },
                  {
                     "endpoint":"ecn.dynamic.t{0-
3}.tiles.virtualearth.net\/comp\/ch\/{quadkey}?mkt=
{market}&it=G,VE,BX,L,LA&shading=hill&n=z&ur={userregion}",
                     "fallbackLanguage":"en-US",
                     "languageSupported":false,
```

```
                    "serviceName":"MapTiles"
                },
                {
                    "endpoint":"ecn.t{0-
7}.tiles.virtualearth.net\/tiles\/dp\/content?mkt={market}&p=tf&a=
{quadkey}&n=z",
                    "fallbackLanguage":"en-US",
                    "languageSupported":false,
                    "serviceName":"TrafficTiles"
                },
                {
                    "endpoint":"dev.virtualearth.net\/REST\/V1\/Imagery\/Ma
p",
                    "fallbackLanguage":"en-US",
                    "languageSupported":false,
                    "serviceName":"StaticMapsB2B"
                },
                {
                    "endpoint":"dev.virtualearth.net\/REST\/V1\/Imagery\/Me
tadata",
                    "fallbackLanguage":"en-US",
                    "languageSupported":false,
                    "serviceName":"MetadataB2B"
                },
                {
                    "endpoint":"dev.virtualearth.net\/REST\/v1\/Locations",
                    "fallbackLanguage":"en-US",
                    "languageSupported":false,
                    "serviceName":"Geocode"
                },
                {
                    "endpoint":"dev.virtualearth.net\/REST\/v1\/Routes",
                    "fallbackLanguage":"en-US",
                    "languageSupported":false,
                    "serviceName":"Route"
                }
            ],
            "ur":"IN"
        }
    ]
  }
 ],
 "statusCode":200,
 "statusDescription":"OK",
 "traceId":"c3d472f7ba6b4c8bbcb4a41c1ce82e8b
}
```


You would receive the following JSON response if the output=xml parameter was set in this example.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<Response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.microsoft.com/search/local/ws/rest/v1">
  <Copyright>Copyright © 2013 Microsoft and its suppliers. All rights
reserved. This API cannot be accessed and the content and any results may
not be used, reproduced or transmitted in any manner without express written
permission from Microsoft Corporation.</Copyright>

<BrandLogoUri>http://dev.virtualearth.net/Branding/logo_powered_by.png</Bran
dLogoUri>
  <StatusCode>200</StatusCode>
  <StatusDescription>OK</StatusDescription>
  <AuthenticationResultCode>ValidCredentials</AuthenticationResultCode>
  <TraceId>105c3b6e5e6a4577a4bb405b72a2b413|CPKM001262|02.00.117.100|
</TraceId>
  <ResourceSets>
    <ResourceSet>
      <EstimatedTotal>1</EstimatedTotal>
      <Resources>
        <Resource xsi:type="GeospatialEndpointResponse">
          <UR>IN</UR>
          <IsDisputedArea>true</IsDisputedArea>
          <IsSupported>true</IsSupported>
          <Services>
            <ServiceInfo>
              <ServiceName>RoadWithLabels</ServiceName>
              <Endpoint>ecn.dynamic.t{0-
3}.tiles.virtualearth.net/comp/ch/{quadkey}?mkt=en-
us&amp;it=G,L&amp;shading=hil&amp;n=z</Endpoint>
              <LanguageSupported>false</LanguageSupported>
              <FallbackLanguageId>en-US</FallbackLanguageId>
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>AerialWithLabels</ServiceName>
              <Endpoint>ecn.dynamic.t{0-
3}.tiles.virtualearth.net/comp/ch/{quadkey}?mkt=en-
us&amp;it=A,G,L&amp;n=z</Endpoint>
              <LanguageSupported>false</LanguageSupported>
              <FallbackLanguageId>en-US</FallbackLanguageId>
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>RoadWithoutLabels</ServiceName>
              <Endpoint>ecn.dynamic.t{0-
3}.tiles.virtualearth.net/comp/ch/{quadkey}?mkt=en-
us&amp;it=G&amp;shading=hill&amp;n=z</Endpoint>
              <LanguageSupported>false</LanguageSupported>
              <FallbackLanguageId>en-US</FallbackLanguageId>
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>AerialWithoutLabels</ServiceName>
              <Endpoint>ecn.dynamic.t{0-
3}.tiles.virtualearth.net/comp/ch/{quadkey}?mkt=en-
us&amp;it=A,G&amp;n=z</Endpoint>
```

```
              <LanguageSupported>false</LanguageSupported>
              <FallbackLanguageId>en-US</FallbackLanguageId>
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>MapTiles</ServiceName>
              <Endpoint>ecn.dynamic.t{0-
3}.tiles.virtualearth.net/comp/ch/{quadkey}?mkt=
{market}&amp;it=G,VE,BX,L,LA&amp;shading=hill&amp;n=z&amp;ur={userregion}
</Endpoint>
              <LanguageSupported>false</LanguageSupported>
              <FallbackLanguageId>en-US</FallbackLanguageId>
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>TrafficTiles</ServiceName>
              <Endpoint>ecn.t{0-7}.tiles.virtualearth.net/tiles/dp/content?
mkt={market}&amp;p=tf&amp;a={quadkey}&amp;n=z</Endpoint>
              <LanguageSupported>false</LanguageSupported>
              <FallbackLanguageId>en-US</FallbackLanguageId>
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>StaticMapsB2B</ServiceName>
              <Endpoint>dev.virtualearth.net/REST/V1/Imagery/Map</Endpoint>
              <LanguageSupported>false</LanguageSupported>
              <FallbackLanguageId>en-US</FallbackLanguageId>
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>MetadataB2B</ServiceName>

<Endpoint>dev.virtualearth.net/REST/V1/Imagery/Metadata</Endpoint>
              <LanguageSupported>false</LanguageSupported>
              <FallbackLanguageId>en-US</FallbackLanguageId>
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>Geocode</ServiceName>
              <Endpoint>dev.virtualearth.net/REST/v1/Locations</Endpoint>
              <LanguageSupported>false</LanguageSupported>
              <FallbackLanguageId>en-US</FallbackLanguageId>
            </ServiceInfo>
            <ServiceInfo>
              <ServiceName>Route</ServiceName>
              <Endpoint>dev.virtualearth.net/REST/v1/Routes</Endpoint>
              <LanguageSupported>false</LanguageSupported>
              <FallbackLanguageId>en-US</FallbackLanguageId>
            </ServiceInfo>
          </Services>
        </Resource>
      </Resources>
    </ResourceSet>
  </ResourceSets>
</Response>
```

SSL Certificate Validation for Java Applications

Article • 05/22/2024

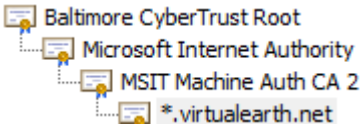
> **Note**
>
> **Bing Maps for Enterprise service retirement**
>
> Bing Maps for Enterprise is deprecated and will be retired. Free (Basic) account customers can continue to use Bing Maps for Enterprise services until June 30th, 2025. Enterprise account customers can continue to use Bing Maps for Enterprise services until June 30th, 2028. To avoid service disruptions, all implementations using Bing Maps for Enterprise REST APIs and SDKs will need to be updated to use Azure Maps by the retirement date that applies to your Bing Maps for Enterprise account type.
>
> Azure Maps is Microsoft's next-generation maps and geospatial services for developers. Azure Maps has many of the same features as Bing Maps for Enterprise, and more. To get started with Azure Maps, create a free Azure subscription and an Azure Maps account. For more information about azure Maps, see Azure Maps Documentation. For migration guidance, see Bing Maps Migration Overview.

Bing Maps APIs support both HTTP and HTTPS requests that use Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols. The SSL and TLS endpoints are secured by certificates. For example, when you query a data source directly from your browser by using the Bing Spatial Data Services (SDS) as in the following example, you can view the certificate chain by clicking the “lock” icon.

url

```
https://spatial.virtualearth.net/REST/v1/data/
  f22876ec257b474b82fe2ffcb8393150/
  NavteqNA/
  NavteqPOIs
  ?spatialFilter=nearby(40.83274904439099,-74.3163299560546935,5)
  &$filter=EntityTypeID%20eq%20'6000'
  &$select=EntityID,DisplayName,Latitude,Longitude,__Distance&
  $top=3
  &key=INSERT_YOUR_BING_MAPS_KEY
```

The following image shows an example of a certificate chain.



Java applications that access the Bing Maps APIs using SSL must implement their own certificate validation checks. The level of validation can vary per application, and often includes validation of the root certificate against a trusted root list. The validation performed by an application is equivalent to the validation that your browser uses to verify an SSL certificate before showing the "lock" icon.

For Java applications, private keys and the associated X.509 certificate chains that authenticate the corresponding public keys are found in the "keystore". The "keystore" also manages certificates from trusted entities. After you install the Java Runtime Environment (JRE) or the Java Development Kit (JDK), you will find the "keystore" in the `/lib/security` directory of your JRE. By default, this subdirectory contains a file named **cacerts** that contains certificates for many Root Certificate Authorities (Root CAs).

Java provides a "keytool" in order to manage your "keystore". To view a list of currently installed certificates, open a command prompt and run the following command from the `bin` directory of the JRE.

Console

```
keytool -list -keystore ..\lib\security\cacerts
```

If you access a Bing Maps API from a Java application via SSL and you do not have the certificate for the Root CA in your "keystore" you will see an error message similar to this:

`Exception in thread "main" com.sun.jersey.api.client.ClientHandlerException: javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target…`

Configuring Root Certificates

The following steps show how to install the SSL certificates required to access Bing Maps APIs.

Baltimore CyberTrust Root

To configure the Baltimore CyberTrust Root:

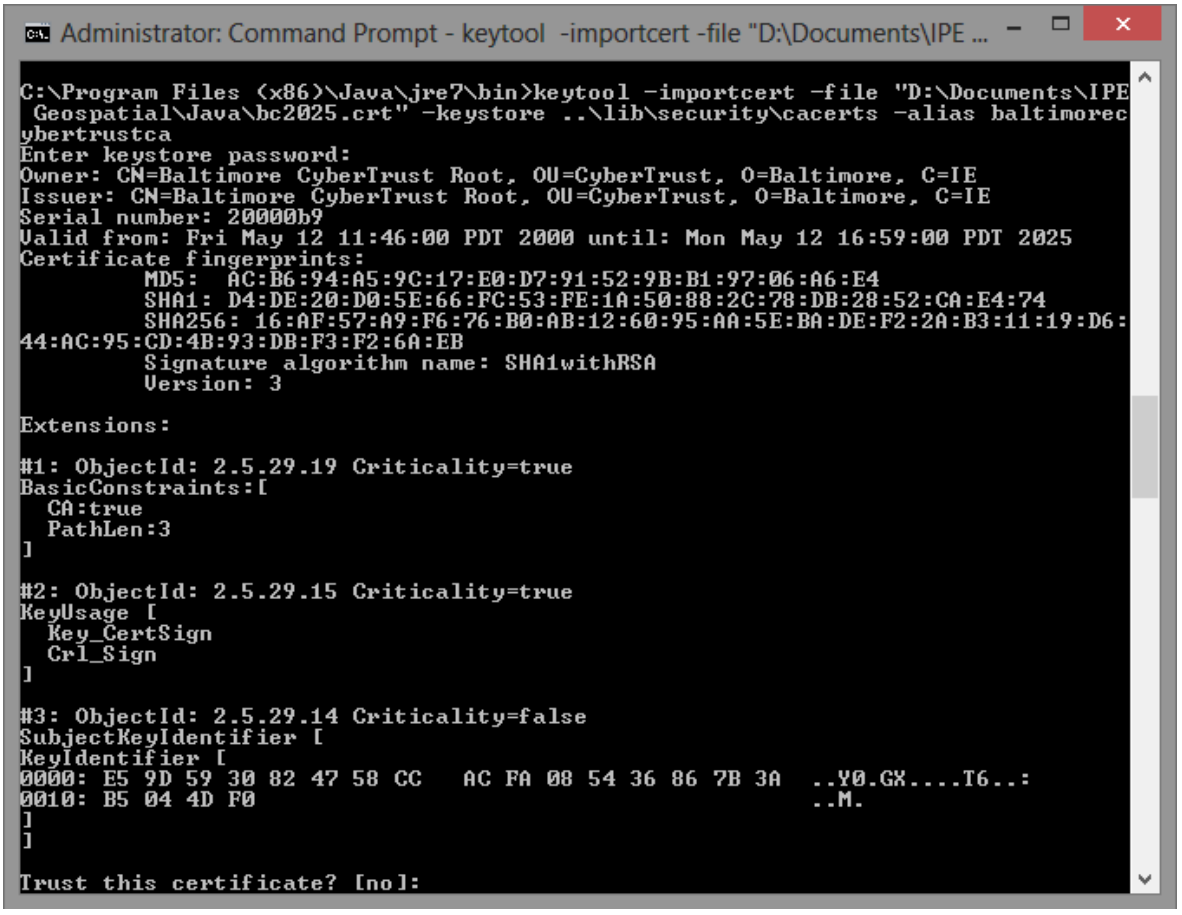
1. Download the Baltimore CyberTrust Root cert and save it to a location on your computer. Note the location as you will need it for the install command below.

2. Open a command prompt and run the following "keytool" command from the `bin` directory of the JRE. This starts the certificate installation. You must insert the directory path to the downloaded certificate from step 1 before running the command. If you are running the Windows operating system, you will need to run the command prompt as an administrator.

   Console

   ```
   keytool -importcert -file
   "InsertCertificateDownloadLocation\bc2025.crt" -keystore
   ..\lib\security\cacerts -alias baltimorecybertrustca
   ```

   When you run the command, the information in the following screenshot appears. To verify the install, type **yes**.



Understanding Scale and Resolution

Article • 07/26/2024

One of the more difficult questions to answer about a Bing map involves determining the exact scale or resolution of a particular image. The answer is complicated as resolution depends on several factors including the current latitude and longitude. Scale is further dependent on screen resolution. In this article we will look at the factors that affect scale and resolution and provide you with formulas you can use to calculate approximate values.

The Mercator Projection

The first problem with both scale and resolution is that the world is neither flat, nor spherical. Any flat representation of the planet is therefore a compromise between accuracy and convenience. The Mercator projection is one of the traditional map projections and offers some very compelling advantages:

- The map is conformal. In other words, that means that the scale is constant around any position. In other words, once we have our "x" scale, we also know our "y" scale for a map segment.
- Rhumb lines are straight segments - The easiest way to understand a rhumb line is to imagine you are sailing a ship. If you point your ship on a compass heading and go forward, you are traveling a rhumb line. The nice feature of Mercator maps is that rhumb lines are straight lines. You can therefore track your course with a ruler, watch and compass (assuming you correct for magnetic declination of course).

Because of these two features, Mercator projection maps are immensely valuable to sailors. They are also very useful for mapping applications because the math works out nicely.

However, there are problems with the Mercator projection. The Mercator projection enforces parallel lines of latitude and longitude. That means that the distance between lines defining one degree of longitude will remain constant, even though the actual distance varies greatly as you travel away from the equator. Latitude is affected the same way. The stretching effect makes objects away from the equator appear larger, although the shape remains relatively correct.

The practical effect of using the Mercator projection is that our local maps are easy to understand and draw, but that the scale and resolution are going to change as a function of latitude and longitude.

Calculating Resolution

Map resolution is a function of the latitude, the zoom level, and a constant value. The constant is based on the diameter of the Earth and the equations Microsoft used to set the zoom levels.

At any latitude and zoom level, you can determine the scale by using the following equation:

```
Map resolution = 156543.04 meters/pixel * cos(latitude) / (2 ^ zoomlevel)
```

Remember that the zoom level goes from 1 to 19, and that latitude goes from -90 to 90 (assuming your cosine function works with degrees; if you need radians, multiply the latitude by Pi/180). The equation naturally fails if you get too close to either pole, as the Mercator projection also fails when you get too close to the poles.

From this equation, we can calculate an estimated scale for each zoom level by assuming that we are at the equator:

Expand table

Zoom Level	Scale (m/pixel)	Zoom Level	Scale (m/pixel)
1	78271.52	**11**	76.44
2	39135.76	**12**	38.22
3	19567.88	**13**	19.11
4	9783.94	**14**	9.55
5	4891.97	**15**	4.78
6	2445.98	**16**	2.39
7	1222.99	**17**	1.19
8	611.50	**18**	0.60
9	305.75	**19**	0.30
10	152.87		

Table 1 Resolution at the equator by zoom level

Keep in mind that these scales will fluctuate with latitude. For example, in Toronto (latitude 43.65), at zoom level 13, the resolution will be `19.11 * cos(43.65)` or about

13.8 meters/pixel. If you are American, you can convert that to feet by 3.28 feet/meter to get 54.26 feet/pixel.

Calculating Scale

Most map people prefer to think in terms of map scale rather than resolution. That is, they'd rather know that 1 map inch translates to 1000 feet. In order to calculate scale, you have to know screen resolution, zoom level, and latitude. You also have to assume that the screen resolution is fixed and equal in both x and y directions. Since screen resolution is usually defined in pixels per inch, you also have to convert to metric.

The equation becomes:

```
Map scale = 1 : (ScreenRes pixels/inch * 39.37 inches/meter * 156543.04 meters/pixel * cos(latitude) / (2 ^ zoomlevel))
```

For example, assuming a typical screen resolution of 85 pixels/inch and a zoom level of 13, you would have a resolution of 1 : 85 * 39.37 * 19.11 or 1 : 63950. That means that every inch on the screen translates to 63,950 inches, or about 1 mile. For those metrically inclined, that would be 1 cm on the screen mapping to 63950 centimeters, or about 0.64 kilometers.

Conclusion

In practical terms, you shouldn't ever really need to worry about resolution or scale for most mapping tasks. Just use the built in scale and let Microsoft do the math. However, if you need to precisely map your points or calculate your own distances, you should be familiar with both equations described above.

Getting Streetside Tiles from Imagery Metadata

Article • 05/22/2024

> **Note**
>
> **Bing Maps for Enterprise service retirement**
>
> Bing Maps for Enterprise is deprecated and will be retired. Free (Basic) account customers can continue to use Bing Maps for Enterprise services until June 30th, 2025. Enterprise account customers can continue to use Bing Maps for Enterprise services until June 30th, 2028. To avoid service disruptions, all implementations using Bing Maps for Enterprise REST APIs and SDKs will need to be updated to use Azure Maps by the retirement date that applies to your Bing Maps for Enterprise account type.
>
> Azure Maps is Microsoft's next-generation maps and geospatial services for developers. Azure Maps has many of the same features as Bing Maps for Enterprise, and more. To get started with Azure Maps, create a free Azure subscription and an Azure Maps account. For more information about azure Maps, see Azure Maps Documentation. For migration guidance, see Bing Maps Migration Overview.

The Bing Maps REST Imagery API now supports the retrieval of Streetside images. In this article, we demonstrate how to to make an Imagery API Metadata URL request at a location on Earth and then use this request to piece together the raw tiles to create a panorama of that location.

Using for our location a point in Ballard, Seattle (`47.668687,-122.384795`) we create a URL request for the Streetside metadata at that location using the Imagery Metadata API.

URL Request

url

```
http://dev.virtualearth.net/REST/v1/Imagery/MetaData/Streetside/47.668687,-122.384795?key={BingMapsKey}
```

JSON Response

JSON


```
{
  "authenticationResultCode": "ValidCredentials",
  "brandLogoUri":
"http://dev.virtualearth.net/Branding/logo_powered_by.png",
  "copyright": "Copyright © 2018 Microsoft and its suppliers. All rights
reserved. This API cannot be accessed and the content and any results may
not be used, reproduced or transmitted in any manner without express written
permission from Microsoft Corporation.",
  "resourceSets": [
    {
      "estimatedTotal": 1,
      "resources": [
        {
          "__type":
"StreetsideMetadata:http://schemas.microsoft.com/search/local/ws/rest/v1",
          "imageHeight": 256,
          "imageUrl": "http://ecn.
{subdomain}.tiles.virtualearth.net/tiles/hs0203232101212100{faceId}{tileId}?
g=6617&key={BingMapsKey}",
          "imageUrlSubdomains": [
            "t0",
            "t1",
            "t2",
            "t3"
          ],
          "imageWidth": 256,
          "imageryProviders": null,
          "vintageEnd": "17 Jul 2014 GMT",
          "vintageStart": "17 Jul 2014 GMT",
          "zoomMax": 4,
          "zoomMin": 1,
          "he": 52.286,
          "lat": 47.668696,
          "lon": -122.384813,
          "pi": 0.638,
          "ro": -0.326
        }
      ]
    }
  ],
  "statusCode": 200,
  "statusDescription": "OK",
  "traceId": "3f6f6409f04d4e4b93dcdd50fee6b508|CO3124D6DA|7.7.0.0"
}
```


For more information about this resource, see the Imagery Metadata.

Piecing the Tiles Together

The above JSON response returned a URL in the `imageURL` field.

url

```
http://ecn.
{subdomain}.tiles.virtualearth.net/tiles/hs0203232101212100{faceId}{tileId}?
g=6617&key={BingMapsKey}
```

This field contains three placeholders: `subdomain`, `faceId`, and `tileId`. Here we have four options for `subdomain`, found in the `imageUrlSubdomains` of the JSON response.

The `faceId` and `tileId` are integers which are used to label the available raw tiles for that location.

In the below picture, each tile is labeled with its `faceId` and `tileId` in the format: `"{faceId}-{tileID}"`.

For example, to retieve the tile with `faceId = 0` and `tileId = 1` (letting `subdomain` be `t0`) we can use the following image URL:

url

```
http://ecn.t0.tiles.virtualearth.net/tiles/hs020323210121210001?g=6617&key=
```

```
{BingMapsKey}
```

This URL request returns the following JPEG image:

Code Sample

Below is a .NET Core 2.0 C# source code example for creating the above panorama by making multiple REST calls to the ImageUrl for each valid `faceId` and `tileId`.

This source is also available on GitHub.

Csharp

```
using System;
using System.Drawing;
using System.Drawing.Imaging;
using System.Net;
using System.IO;
using System.Diagnostics;

namespace GridBuilder
{
    public class MakeImageGrid
    {
        private const int bit_map_size = 256;
        private const string BingMapsKey = "API_KEY_HERE";
        private const string subdomain = "t0";
        private const string URL = "http://ecn.
{0}.tiles.virtualearth.net/tiles/hs0203232101212100{1}{2}?g=6616&key={3}";

        private const int master_size_x = bit_map_size * 4;
        private const int master_size_y = bit_map_size * 3;

        public void Print_Grid(string save_file_name)
        {
            // see
```

```
https://learn.microsoft.com/dotnet/api/system.drawing.bitmap
            Bitmap cube_image = new Bitmap(master_size_x, master_size_y);
            Bitmap tmp_image;

            for (int face_id = 0; face_id < 2; face_id++)
            {
                for (int tile_id = 0; tile_id < ((face_id == 0) ? 4 : 3);
tile_id++)
                {
                    tmp_image = GetCubeSide(face_id, tile_id);
                    UpdateCube(ref cube_image, ref tmp_image, face_id,
tile_id);
                }
            }

            cube_image.Save(save_file_name, ImageFormat.Jpeg);
        }

        private void UpdateCube(ref Bitmap cube_image, ref Bitmap tmp_image,
int face_id, int tile_id)
        {
            int x = 0;
            int y = 0;
            int x_var;
            int y_var;
            Color color;

            int move_len = bit_map_size - 2;

            if (face_id == 0)
            {
                y = move_len;
                x = move_len * (tile_id);
            }
            else
            {
                switch(tile_id)
                {
                    case 1:
                        y = 0;
                        x = move_len;
                        break;
                    case 0:
                        y = move_len;
                        break;
                    case 2:
                    default:
                        x = move_len;
                        y = 2 * move_len;
                        break;
                }
            }

            for (int i = 2; i < bit_map_size; i++)
            {
```

```
                for (int j = 2; j < bit_map_size; j++)
                {
                    color = tmp_image.GetPixel(i, j);
                    x_var = x + i;
                    y_var =y + j;
                    cube_image.SetPixel(x_var, y_var, color);
                }
            }

            Graphics g = Graphics.FromImage(cube_image);
            using (Font font = new Font("Times New Roman", 24,
FontStyle.Bold, GraphicsUnit.Pixel))
            {
                PointF p = new PointF(x, y);
                g.DrawString($"{face_id}-{tile_id}", font,
Brushes.IndianRed, p);
                g.Flush();
            }
        }

        private Bitmap GetCubeSide(int face_id, int tile_id)
        {
            Bitmap bp;
            using (WebClient webClient = new WebClient())
            {
                Uri fin_url = new Uri(string.Format(URL, subdomain, face_id,
tile_id, BingMapsKey));
                Console.WriteLine("http: {0}", fin_url);
                byte[] data = webClient.DownloadData(fin_url);
                using (var stream = new MemoryStream(data))
                {
                    bp = new Bitmap(stream);
                }
            }

            return bp;
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            var save_file = "test.jpeg";
            MakeImageGrid image = new MakeImageGrid();
            image.Print_Grid(save_file);
            Console.ReadKey();
        }
    }
}
```

Open Maps: understanding ODbL

Article • 05/22/2024

> ⓘ Note
>
> **Bing Maps for Enterprise service retirement**
>
> Bing Maps for Enterprise is deprecated and will be retired. Free (Basic) account customers can continue to use Bing Maps for Enterprise services until June 30th, 2025. Enterprise account customers can continue to use Bing Maps for Enterprise services until June 30th, 2028. To avoid service disruptions, all implementations using Bing Maps for Enterprise REST APIs and SDKs will need to be updated to use Azure Maps by the retirement date that applies to your Bing Maps for Enterprise account type.
>
> Azure Maps is Microsoft's next-generation maps and geospatial services for developers. Azure Maps has many of the same features as Bing Maps for Enterprise, and more. To get started with Azure Maps, create a free Azure subscription and an Azure Maps account. For more information about azure Maps, see Azure Maps Documentation. For migration guidance, see Bing Maps Migration Overview.

ODBL in Bing Maps

Certain data available in the Bing Maps v8 web control API will be subject to the Open Database License (ODbL). This data will be identified as the **building layer.**

What is ODbL?

The Open Database License (ODbL) is an open source data license that enables users to utilize the data in their products, services and applications, subject to certain license requirements. The ODbL was developed by Open Data Commons and is used by several open source mapping initiatives, such as OpenStreetMap and Paris Open Data. Microsoft is including map data from databases under ODbL licenses in its APIs in order to provide a robust mapping solution to users.

> ⚠ Warning
>
> **This page is for informational purposes only**. It does not constitute legal advice or interpretation. For information about using data subject to ODbL, seek the

guidance of an attorney. Microsoft is not liable or responsible for your use of any data subject to ODbL.

What are the requirements for use of ODbL data?

Attribute: If you use data under an ODbL license, you must provide attribution to the source of the data. If you are using the Bing Maps v8 web control API, such credits are already listed in the Bing Suppliers Page (https://bingexplore.azurewebsites.net/bing-data-suppliers/en/) that is linked from the Bing Maps TOU that you must link to under the terms of the Bing Maps API agreement. However, Microsoft cannot give you specific guidelines about attribution, as the final method of attribution will vary based on your use case.

- **Share-Alike:** If you adapt any database that is subject to the ODbL, either by adding your own or third-party content, or by removing or modifying inaccurate content, the resulting adapted version of the database must also be released under the ODbL.
- **Other Requirements:** You should review the ODbL text at https://opendatacommons.org/licenses/odbl/ to ensure that your usage of any ODbL content complies with the license terms.

Do I have to share my data if I use ODbL data?

You will not have to automatically share your data if you use data under the ODbL license in your application. If you simply use the data as-is, without modifying the database or creating a hybrid database that combines multiple datasets, it is unlikely that you will have to share your contributed data. If you segregate your data from ODbL-licensed data or just use ODbL-licensed data to create a map, it is unlikely that you would have to share your segregated data or the output map.

However, if you combine a number of sources into a single, unified database or modify an existing ODbL database using other information, it is likely that you would need to release the resulting database under the ODbL license, which means that other people could see your data and use it. You should consult your legal representative to determine whether your specific use of ODbL data may subject you to share-alike terms.

Where can I get more information about ODbL licensing rules?

Contact your legal representative if you have questions about using ODbL content in your application. Microsoft cannot provide you with any specific advice about usage of ODbL content.