**The Google Geocoding API**

1. [**What is Geocoding?**](http://code.google.com/apis/maps/documentation/geocoding/#Geocoding)
2. [**Audience**](http://code.google.com/apis/maps/documentation/geocoding/#Audience)
3. [**Usage Limits**](http://code.google.com/apis/maps/documentation/geocoding/#Limits)
4. [**Geocoding Requests**](http://code.google.com/apis/maps/documentation/geocoding/#GeocodingRequests)
5. [**Geocoding Responses**](http://code.google.com/apis/maps/documentation/geocoding/#GeocodingResponses)
   1. [JSON Output Formats](http://code.google.com/apis/maps/documentation/geocoding/#JSON)
   2. [XML Output Formats](http://code.google.com/apis/maps/documentation/geocoding/#XML)
   3. [Status Codes](http://code.google.com/apis/maps/documentation/geocoding/#StatusCodes)
   4. [Results](http://code.google.com/apis/maps/documentation/geocoding/#Results)
   5. [Address Component Types](http://code.google.com/apis/maps/documentation/geocoding/#Types)
6. [**Reverse Geocoding**](http://code.google.com/apis/maps/documentation/geocoding/#ReverseGeocoding)
7. [**Viewport Biasing**](http://code.google.com/apis/maps/documentation/geocoding/#Viewports)
8. [**Region Biasing**](http://code.google.com/apis/maps/documentation/geocoding/#RegionCodes)

This document discusses the newest version of the Geocoding API (V3). Note that the legacy [Geocoding API V2](http://code.google.com/apis/maps/documentation/geocoding/v2/index.html) has been deprecated. Users of that service should upgrade to this version.

Looking to use this service in a JavaScript application? Check out the [Geocoder](http://code.google.com/apis/maps/documentation/javascript/services.html#Geocoding) class of the Google Maps API v3.

**What is Geocoding?**

Geocoding is the process of converting addresses (like "1600 Amphitheatre Parkway, Mountain View, CA") into geographic coordinates (like latitude 37.423021 and longitude -122.083739), which you can use to place markers or position the map. The Google Geocoding API provides a direct way to access a geocoder via an HTTP request. Additionally, the service allows you to perform the converse operation (turning coordinates into addresses); this process is known as "reverse geocoding."

**Audience**

This document is intended for website and mobile developers who want to use geocoding data within maps provided by one of the Google Maps APIs. It provides an introduction to using this API and reference material on the available parameters.

This service is generally designed for geocoding static (known in advance) addresses for placement of application content on a map; this service is **not** designed to respond in real time to user input, for example. For dynamic geocoding (for example, within a user interface element), consult the documentation for the [JavaScript API V2 Client Geocoder](http://code.google.com/apis/maps/documentation/services.html#Geocoding), the [JavaScript API V3 Client Geocoder](http://code.google.com/apis/maps/documentation/javascript/services.html#Geocoding), or the [Maps API for Flash Client Geocoder](http://code.google.com/apis/maps/documentation/flash/services.html#Geocoding).

Geocoding is a time and resource intensive task. Whenever possible, pre-geocode known addresses (using the Geocoding API described here or another geocoding service), and store your results in a temporary cache of your own design.

**Usage Limits**

Use of the Google Geocoding API is subject to a query limit of 2,500 geolocation requests per day. (User of Google Maps API Premier may perform up to 100,000 requests per day.) This limit is enforced to prevent abuse and/or repurposing of the Geocoding API, and this limit may be changed in the future without notice. Additionally, we enforce a request rate limit to prevent abuse of the service. If you exceed the 24-hour limit or otherwise abuse the service, the Geocoding API may stop working for you temporarily. If you continue to exceed this limit, your access to the Geocoding API may be blocked.

Note: the Geocoding API may only be used in conjunction with a Google map; geocoding results without displaying them on a map is prohibited. For complete details on allowed usage, consult the [Maps API Terms of Service License Restrictions](http://code.google.com/apis/maps/terms.html#section_10_12).

**Geocoding Requests**

A Geocoding API request must be of the following form:

http://maps.googleapis.com/maps/api/geocode/*output*?*parameters*

where output may be either of the following values:

* json (recommended) indicates output in JavaScript Object Notation (JSON)
* xml indicates output as XML

To access the Geocoding API over HTTPS, use:

**https**://maps.googleapis.com/maps/api/geocode/*output*?*parameters*

HTTPS is recommended for applications that include sensitive user data, such as a user's location, in requests.

In either case, certain parameters are required while some are optional. As is standard in URLs, all parameters are separated using the ampersand (&) character. The list of parameters and their possible values are enumerated below.

The Geocoding API defines a geocoding request using the following URL parameters:

* address (*required*) — The address that you want to geocode.\*

     OR

* latlng (*required*) — The textual latitude/longitude value for which you wish to obtain the closest, human-readable address.\*
* bounds (*optional*) — The bounding box of the viewport within which to bias geocode results more prominently. (For more information see [Viewport Biasing](http://code.google.com/apis/maps/documentation/geocoding/#Viewports) below.)
* region (*optional*) — The region code, specified as a ccTLD ("top-level domain") two-character value. (For more information see[Region Biasing](http://code.google.com/apis/maps/documentation/geocoding/#RegionCodes) below.)
* language (*optional*) — The language in which to return results. See the [supported list of domain languages](http://code.google.com/apis/maps/faq.html#languagesupport). Note that we often update supported languages so this list may not be exhaustive. If language is not supplied, the geocoder will attempt to use the native language of the domain from which the request is sent wherever possible.
* sensor (*required*) — Indicates whether or not the geocoding request comes from a device with a location sensor. This value must be either true or false.

\* Note: You may pass either an address or a latlng to lookup. (If you pass a latlng, the geocoder performs what is known as a*reverse geocode*. See [Reverse Geocoding](http://code.google.com/apis/maps/documentation/geocoding/#ReverseGeocoding) for more information.)

The bounds and region parameters will only influence, not fully restrict, results from the geocoder.

**Geocoding Responses**

Geocoding responses are returned in the format indicates by the output flag within the URL request's path.

**JSON Output Formats**

In this example, the Geocoding API requests a json response for a query on "1600 Amphitheatre Parkway, Mountain View, CA":

http://maps.googleapis.com/maps/api/geocode/json?address=1600+Amphitheatre+Parkway,+Mountain+View,+CA&sensor=*true\_or\_false*

We've left the sensor parameter in this example as a variable *true\_or\_false* to emphasize that you **must** set this value to either trueor false explicitly.

The JSON returned by this request is shown below. Note that actual JSON may contain less whitespace. You should not make assumptions about the amount or format of whitespace between requests.

{  
  "status": "OK",  
  "results": [ {  
    "types": [ "street\_address" ],  
    "formatted\_address": "1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA",  
    "address\_components": [ {  
      "long\_name": "1600",  
      "short\_name": "1600",  
      "types": [ "street\_number" ]  
    }, {  
      "long\_name": "Amphitheatre Pkwy",  
      "short\_name": "Amphitheatre Pkwy",  
      "types": [ "route" ]  
    }, {  
      "long\_name": "Mountain View",  
      "short\_name": "Mountain View",  
      "types": [ "locality", "political" ]  
    }, {  
      "long\_name": "California",  
      "short\_name": "CA",  
      "types": [ "administrative\_area\_level\_1", "political" ]  
    }, {  
      "long\_name": "United States",  
      "short\_name": "US",  
      "types": [ "country", "political" ]  
    }, {  
      "long\_name": "94043",  
      "short\_name": "94043",  
      "types": [ "postal\_code" ]  
    } ],  
    "geometry": {  
      "location": {  
        "lat": 37.4219720,  
        "lng": -122.0841430  
      },  
      "location\_type": "ROOFTOP",  
      "viewport": {  
        "southwest": {  
          "lat": 37.4188244,  
          "lng": -122.0872906  
        },  
        "northeast": {  
          "lat": 37.4251196,  
          "lng": -122.0809954  
        }  
      }  
    }  
  } ]  
}

Note that the JSON response contains two root elements:

* "status" contains metadata on the request. See [Status Codes](http://code.google.com/apis/maps/documentation/geocoding/#StatusCodes) below.
* "results" contains an array of geocoded address information and geometry information.

Generally, only one entry in the "results" array is returned for address lookups, though the geocoder may return several results when address queries are ambiguous.

Note that these results generally need to be *parsed* if you wish to extract values from the results. Parsing JSON is relatively easy. See[Parsing JSON](http://code.google.com/apis/maps/documentation/webservices/index.html#ParsingJSON) for some recommended design patterns.

**XML Output Formats**

In this example, the Geocoding API requests an xml response for the identical query shown above for "1600 Amphitheatre Parkway, Mountain View, CA":

http://maps.googleapis.com/maps/api/geocode/xml?address=1600+Amphitheatre+Parkway,+Mountain+View,+CA&sensor=*true\_or\_false*

The XML returned by this request is shown below.

<GeocodeResponse>   
 <status>OK</status>   
 <result>   
  <type>street\_address</type>   
  <formatted\_address>1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA</formatted\_address>   
  <address\_component>   
   <long\_name>1600</long\_name>   
   <short\_name>1600</short\_name>   
   <type>street\_number</type>   
  </address\_component>   
  <address\_component>   
   <long\_name>Amphitheatre Pkwy</long\_name>   
   <short\_name>Amphitheatre Pkwy</short\_name>   
   <type>route</type>   
  </address\_component>   
  <address\_component>   
   <long\_name>Mountain View</long\_name>   
   <short\_name>Mountain View</short\_name>   
   <type>locality</type>   
   <type>political</type>   
  </address\_component>   
  <address\_component>   
   <long\_name>San Jose</long\_name>   
   <short\_name>San Jose</short\_name>   
   <type>administrative\_area\_level\_3</type>   
   <type>political</type>   
  </address\_component>   
  <address\_component>   
   <long\_name>Santa Clara</long\_name>   
   <short\_name>Santa Clara</short\_name>   
   <type>administrative\_area\_level\_2</type>   
   <type>political</type>   
  </address\_component>   
  <address\_component>   
   <long\_name>California</long\_name>   
   <short\_name>CA</short\_name>   
   <type>administrative\_area\_level\_1</type>   
   <type>political</type>   
  </address\_component>   
  <address\_component>   
   <long\_name>United States</long\_name>   
   <short\_name>US</short\_name>   
   <type>country</type>   
   <type>political</type>   
  </address\_component>   
  <address\_component>   
   <long\_name>94043</long\_name>   
   <short\_name>94043</short\_name>   
   <type>postal\_code</type>   
  </address\_component>   
  <geometry>   
   <location>   
    <lat>37.4217550</lat>   
    <lng>-122.0846330</lng>   
   </location>   
   <location\_type>ROOFTOP</location\_type>   
   <viewport>   
    <southwest>   
     <lat>37.4188514</lat>   
     <lng>-122.0874526</lng>   
    </southwest>   
    <northeast>   
     <lat>37.4251466</lat>   
     <lng>-122.0811574</lng>   
    </northeast>   
   </viewport>   
  </geometry>   
 </result>   
</GeocodeResponse>

Note that the XML response consists of a single <GeocodeResponse> and two top-level elements:

* <status> contains metadata on the request. See [Status Codes](http://code.google.com/apis/maps/documentation/geocoding/#StatusCodes) below.
* Zero or more <result> elements, each containing a single set of geocoded address information and geometry information.

Note that this response is considerably longer than the JSON response. For that reason, we recommend that you use json as the preferred output flag unless your service requires xml for some reason. Additionally, processing XML trees requires some care, so that you reference proper nodes and elements. See [Parsing XML with XPath](http://code.google.com/apis/maps/documentation/webservices/index.html#XMLParsing) for some recommended design patterns for output processing.

The remainder of this documentation will use JSON syntax. In most cases, the output format does not matter for purposes of illustrating concepts or field names in the documentation. However, note the following subtle differences:

* XML results are wrapped in a root <GeocodeResponse> element.
* JSON denotes entries with multiple elements by plural arrays (types), while XML denotes these using multiple singular elements (<type>).
* Blank elements are indicated through empty arrays in JSON, but by the absense of any such element in XML. A response that generates no results will return an empty results array in JSON, but no <result> elements in XML, for example.

**Status Codes**

The "status" field within the Geocoding response object contains the status of the request, and may contain debugging information to help you track down why Geocoding is not working. The "status" field may contain the following values:

* "OK" indicates that no errors occurred; the address was successfully parsed and at least one geocode was returned.
* "ZERO\_RESULTS" indicates that the geocode was successful but returned no results. This may occur if the geocode was passed a non-existent address or a latlng in a remote location.
* "OVER\_QUERY\_LIMIT" indicates that you are over your quota.
* "REQUEST\_DENIED" indicates that your request was denied, generally because of lack of a sensor parameter.
* "INVALID\_REQUEST" generally indicates that the query (address or latlng) is missing.

**Results**

When the geocoder returns results, it places them within a (JSON) results array. Even if the geocoder returns no results (such as if the address doesn't exist) it still returns an empty results array. (XML responses consist of zero or more <result> elements.)

A typical result is made up of the following fields:

* The types[] array indicates the *type* of the returned result. This array contains a set of one or more tags identifying the type of feature returned in the result. For example, a geocode of "Chicago" returns "locality" which indicates that "Chicago" is a city, and also returns "political" which indicates it is a political entity.
* formatted\_address is a string containing the human-readable address of this location. Often this address is equivalent to the "postal address," which sometimes differs from country to country. (Note that some countries, such as the United Kingdom, do not allow distribution of true postal addresses due to licensing restrictions.) This address is generally composed of one or more*address components*. For example, the address "111 8th Avenue, New York, NY" contains separate address components for "111" (the street number, "8th Avenue" (the route), "New York" (the city) and "NY" (the US state). These address components contain additional information as noted below.
* address\_components[] is an array containing the separate address components, as explained above. Eachaddress\_component typically contains:
  + types[] is an array indicating the *type* of the address component.
  + long\_name is the full text description or name of the address component as returned by the Geocoder.
  + short\_name is an abbreviated textual name for the address component, if available. For example, an address component for the state of Alaska may have a long\_name of "Alaska" and a short\_name of "AK" using the 2-letter postal abbreviation.

Note that address\_components[] may contain more address components than noted within the formatted\_address.

* geometry contains the following information:
  + location contains the geocoded latitude,longitude value. For normal address lookups, this field is typically the most important.
  + location\_type stores additional data about the specified location. The following values are currently supported:
    - "ROOFTOP" indicates that the returned result is a precise geocode for which we have location information accurate down to street address precision.
    - "RANGE\_INTERPOLATED" indicates that the returned result reflects an approximation (usually on a road) interpolated between two precise points (such as intersections). Interpolated results are generally returned when rooftop geocodes are unavailable for a street address.
    - "GEOMETRIC\_CENTER" indicates that the returned result is the geometric center of a result such as a polyline (for example, a street) or polygon (region).
    - "APPROXIMATE" indicates that the returned result is approximate.
  + viewport contains the recommended viewport for displaying the returned result, specified as two latitude,longitude values defining the southwest and northeast corner of the viewport bounding box. Generally the viewport is used to frame a result when displaying it to a user.
  + bounds (optionally returned) stores the bounding box which can fully contain the returned result. Note that these bounds may not match the recommended viewport. (For example, San Francisco includes the [Farallon islands](http://en.wikipedia.org/wiki/Farallon_Islands), which are technically part of the city, but probably should not be returned in the viewport.)
* partial\_match indicates that the geocoder did not return an exact match for the original request, though it did match part of the requested address. You may wish to examine the original request for misspellings and/or an incomplete address. Partial matches most often occur for street addresses that do not exist within the locality you pass in the request.

As the exact format of an individual response to a Geocoding API request is not guaranteed, you should never assume that elements are in absolute positions. (In particular, the number of address\_components within a Geocoding API response vary based on the address requested and can change over time.) Instead, you should **parse** the response and select appropriate values via **expressions**. See [Parsing Web Service Responses](http://code.google.com/apis/maps/documentation/webservices/index.html#Parsing) for more information.

**Address Component Types**

The types[] array within the returned result indicates the *address type*. These types may also be returned withinaddress\_components[] arrays to indicate the type of the particular address component. Addresses within the geocoder may have multiple types; the types may be considered "tags". For example, many cities are tagged with the political and locality type.

The following types are supported and returned by the HTTP Geocoder:

* street\_address indicates a precise street address.
* route indicates a named route (such as "US 101").
* intersection indicates a major intersection, usually of two major roads.
* political indicates a political entity. Usually, this type indicates a polygon of some civil administration.
* country indicates the national political entity, and is typically the highest order type returned by the Geocoder.
* administrative\_area\_level\_1 indicates a first-order civil entity below the country level. Within the United States, these administrative levels are states. Not all nations exhibit these administrative levels.
* administrative\_area\_level\_2 indicates a second-order civil entity below the country level. Within the United States, these administrative levels are counties. Not all nations exhibit these administrative levels.
* administrative\_area\_level\_3 indicates a third-order civil entity below the country level. This type indicates a minor civil division. Not all nations exhibit these administrative levels.
* colloquial\_area indicates a commonly-used alternative name for the entity.
* locality indicates an incorporated city or town political entity.
* sublocality indicates an first-order civil entity below a locality
* neighborhood indicates a named neighborhood
* premise indicates a named location, usually a building or collection of buildings with a common name
* subpremise indicates a first-order entity below a named location, usually a singular building within a collection of buildings with a common name
* postal\_code indicates a postal code as used to address postal mail within the country.
* natural\_feature indicates a prominent natural feature.
* airport indicates an airport.
* park indicates a named park.
* point\_of\_interest indicates a named point of interest. Typically, these "POI"s are prominent local entities that don't easily fit in another category such as "Empire State Building" or "Statue of Liberty."

In addition to the above, address components may exhibit the following types:

* post\_box indicates a specific postal box.
* street\_number indicates the precise street number.
* floor indicates the floor of a building address.
* room indicates the room of a building address.

**Reverse Geocoding (Address Lookup)**

The term *geocoding* generally refers to translating a human-readable address into a location on a map. The process of doing the converse, translating a location on the map into a human-readable address, is known as *reverse geocoding*.

The Geocoding API supports reverse geocoding directly using the latlng parameter. For example, the following query contains the latitude/longitude value for a location in Brooklyn:

http://maps.googleapis.com/maps/api/geocode/json?latlng=40.714224,-73.961452&sensor=*true\_or\_false*

Note: Ensure that no space exists between the latitude and longitude values when passed in the latlng parameter.

This query returns the following result:

{  
  "status": "OK",  
  "results": [ {  
    "types": [ "street\_address" ],  
    "formatted\_address": "275-291 Bedford Ave, Brooklyn, NY 11211, USA",  
    "address\_components": [ {  
      "long\_name": "275-291",  
      "short\_name": "275-291",  
      "types": [ "street\_number" ]  
    }, {  
      "long\_name": "Bedford Ave",  
      "short\_name": "Bedford Ave",  
      "types": [ "route" ]  
    }, {  
      "long\_name": "New York",  
      "short\_name": "New York",  
      "types": [ "locality", "political" ]  
    }, {  
      "long\_name": "Brooklyn",  
      "short\_name": "Brooklyn",  
      "types": [ "administrative\_area\_level\_3", "political" ]  
    }, {  
      "long\_name": "Kings",  
      "short\_name": "Kings",  
      "types": [ "administrative\_area\_level\_2", "political" ]  
    }, {  
      "long\_name": "New York",  
      "short\_name": "NY",  
      "types": [ "administrative\_area\_level\_1", "political" ]  
    }, {  
      "long\_name": "United States",  
      "short\_name": "US",  
      "types": [ "country", "political" ]  
    }, {  
      "long\_name": "11211",  
      "short\_name": "11211",  
      "types": [ "postal\_code" ]  
    } ],  
    "geometry": {  
      "location": {  
        "lat": 40.7142298,  
        "lng": -73.9614669  
      },  
      "location\_type": "RANGE\_INTERPOLATED",  
      "viewport": {  
        "southwest": {  
          "lat": 40.7110822,  
          "lng": -73.9646145  
        },  
        "northeast": {  
          "lat": 40.7173774,  
          "lng": -73.9583193  
        }  
      }  
    }  
  },  
    
  ... Additional results[] ...

Note that the reverse geocoder returned more than one result. The result's "formatted\_addresses" are not just postal addresses, but any way to geographically name a location. For example, when geocoding a point in the city of Chicago, the geocoded point may be denoted as a street address, as the city (Chicago), as its state (Illinois) or as a country (The United States). All are "addresses" to the geocoder. The reverse geocoder returns any of these types as valid results.

The reverse geocoder matches political entities (countries, provinces, cities and neighborhoods), street addresses, and postal codes.

The full list of formatted\_address values returned by the previous query are shown below.

"formatted\_address": "275-291 Bedford Ave, Brooklyn, NY 11211, USA",  
"formatted\_address": "Williamsburg, NY, USA",  
"formatted\_address": "New York 11211, USA",  
"formatted\_address": "Kings, New York, USA",  
"formatted\_address": "Brooklyn, NY, USA",  
"formatted\_address": "New York, NY, USA",  
"formatted\_address": "New York, USA",  
"formatted\_address": "United States"

Generally, addresses are returned from most specific to least specific; the more exact address is the most prominent result, as it is in this case. Note that we return different types of addresses, from the most specific street address to less specific political entities such as neighborhoods, cities, counties, states, etc. If you wish to match a more general address, you may wish to inspect the "types"field of the returned Placemarks. (See [Address Component Types](http://code.google.com/apis/maps/documentation/geocoding/#Types) above.)

Note: Reverse geocoding is an estimate. The geocoder will attempt to find the closest addressable location within a certain tolerance; if no match is found, the geocoder will return zero results.

**Viewport Biasing**

You can also instruct the Geocoding service to prefer results within a given viewport (expressed as a bounding box). You do so within the request URL by setting the bounds parameter. Note that biasing only *prefers* results within the bounds; if more relevant results exist outside of these bounds, they may be included.

The bounds parameter defines the latitude/longitude coordinates of the southwest and northeast corners of this bounding box using a pipe (|) character to separate the coordinates.

For example, a geocode for "Winnetka" generally returns this suburb of Chicago:

Request:

http://maps.googleapis.com/maps/api/geocode/json?address=Winnetka&sensor=false

Response:

{  
  "status": "OK",  
  "results": [ {  
    "types": [ "locality", "political" ],  
    "formatted\_address": "Winnetka, IL, USA",  
    "address\_components": [ {  
      "long\_name": "Winnetka",  
      "short\_name": "Winnetka",  
      "types": [ "locality", "political" ]  
    }, {  
      "long\_name": "Illinois",  
      "short\_name": "IL",  
      "types": [ "administrative\_area\_level\_1", "political" ]  
    }, {  
      "long\_name": "United States",  
      "short\_name": "US",  
      "types": [ "country", "political" ]  
    } ],  
    "geometry": {  
      "location": {  
        "lat": 42.1083080,  
        "lng": -87.7417070  
      },  
      "location\_type": "APPROXIMATE",  
      "viewport": {  
        "southwest": {  
          "lat": 42.0917501,  
          "lng": -87.7737218  
        },  
        "northeast": {  
          "lat": 42.1248616,  
          "lng": -87.7096922  
        }  
      },  
      "bounds": {  
        "southwest": {  
          "lat": 42.0885320,  
          "lng": -87.7715480  
        },  
        "northeast": {  
          "lat": 42.1284090,  
          "lng": -87.7110160  
        }  
      }  
    }  
  } ]  
}

However, adding a bounds argument defining a bounding box for the San Fernando Valley of Los Angeles results in this geocode returning the neighborhood named "Winnetka" in that location:

Request:

http://maps.googleapis.com/maps/api/geocode/json?address=Winnetka&bounds=34.172684,-118.604794|34.236144,-118.500938&sensor=false

Response:

{  
  "status": "OK",  
  "results": [ {  
    "types": [ "sublocality", "political" ],  
    "formatted\_address": "Winnetka, California, USA",  
    "address\_components": [ {  
      "long\_name": "Winnetka",  
      "short\_name": "Winnetka",  
      "types": [ "sublocality", "political" ]  
    }, {  
      "long\_name": "Los Angeles",  
      "short\_name": "Los Angeles",  
      "types": [ "administrative\_area\_level\_3", "political" ]  
    }, {  
      "long\_name": "Los Angeles",  
      "short\_name": "Los Angeles",  
      "types": [ "administrative\_area\_level\_2", "political" ]  
    }, {  
      "long\_name": "California",  
      "short\_name": "CA",  
      "types": [ "administrative\_area\_level\_1", "political" ]  
    }, {  
      "long\_name": "United States",  
      "short\_name": "US",  
      "types": [ "country", "political" ]  
    } ],  
    "geometry": {  
      "location": {  
        "lat": 34.2131710,  
        "lng": -118.5710220  
      },  
      "location\_type": "APPROXIMATE",  
      "viewport": {  
        "southwest": {  
          "lat": 34.1947148,  
          "lng": -118.6030368  
        },  
        "northeast": {  
          "lat": 34.2316232,  
          "lng": -118.5390072  
        }  
      },  
      "bounds": {  
        "southwest": {  
          "lat": 34.1791050,  
          "lng": -118.5883200  
        },  
        "northeast": {  
          "lat": 34.2353090,  
          "lng": -118.5534191  
        }  
      }  
    }  
  } ]  
}

**Region Biasing**

The Google Geocoding API returns address results influenced by the region (typically the country) from which the request is sent. For example, searches for "San Francisco" may return different results if sent from a domain within the United States than one sent from Spain.

You can set the Geocoding API to return results biased to a particular region using the region parameter. This parameter takes a[ccTLD](http://en.wikipedia.org/wiki/CcTLD) (country code top-level domain) argument specifying the region bias. Most ccTLD codes are identical to ISO 3166-1 codes, with some notable exceptions. For example, the United Kingdom's ccTLD is "uk" (.co.uk) while its ISO 3166-1 code is "gb" (technically for the entity of "The United Kingdom of Great Britain and Northern Ireland").

Geocoding results can be biased for every domain in which the main Google Maps application is officially launched. Note that biasing only *prefers* results for a specific domain; if more relevant results exist outside of this domain, they may be included.

For example, a geocode for "Toledo" returns this result, as the default domain for the Geocoding API is set to the United States:

http://maps.googleapis.com/maps/api/geocode/json?address=Toledo&sensor=false

# Returns:

#

{

"status": "OK",

"results": [ {

"types": [ "locality", "political" ],

"formatted\_address": "Toledo, OH, USA",

"address\_components": [ {

"long\_name": "Toledo",

"short\_name": "Toledo",

"types": [ "locality", "political" ]

}, {

"long\_name": "Ohio",

"short\_name": "OH",

"types": [ "administrative\_area\_level\_1", "political" ]

}, {

"long\_name": "United States",

"short\_name": "US",

"types": [ "country", "political" ]

} ],

"geometry": {

"location": {

"lat": 41.6529200,

"lng": -83.5777820

},

"location\_type": "APPROXIMATE",

"viewport": {

"southwest": {

"lat": 41.5861889,

"lng": -83.7058414

},

"northeast": {

"lat": 41.7195821,

"lng": -83.4497226

}

},

"bounds": {

"southwest": {

"lat": 41.5803170,

"lng": -83.6947540

},

"northeast": {

"lat": 41.7326310,

"lng": -83.4545660

}

}

}

} ]

}

A geocode for "Toledo" with region=es (Spain) will return the Spanish city:

http://maps.googleapis.com/maps/api/geocode/json?address=Toledo&sensor=false&region=es

#

# Returns

#

{

"status": "OK",

"results": [ {

"types": [ "locality", "political" ],

"formatted\_address": "Toledo, España",

"address\_components": [ {

"long\_name": "Toledo",

"short\_name": "Toledo",

"types": [ "locality", "political" ]

}, {

"long\_name": "Toledo",

"short\_name": "TO",

"types": [ "administrative\_area\_level\_2", "political" ]

}, {

"long\_name": "Castilla-La Mancha",

"short\_name": "CM",

"types": [ "administrative\_area\_level\_1", "political" ]

}, {

"long\_name": "España",

"short\_name": "ES",

"types": [ "country", "political" ]

} ],

"geometry": {

"location": {

"lat": 39.8567775,

"lng": -4.0244759

},

"location\_type": "APPROXIMATE",

"viewport": {

"southwest": {

"lat": 39.7882200,

"lng": -4.1525353

},

"northeast": {

"lat": 39.9252666,

"lng": -3.8964165

}

},

"bounds": {

"southwest": {

"lat": 39.8105550,

"lng": -4.1796354

},

"northeast": {

"lat": 39.9250920,

"lng": -3.8147915

}

}

}

} ]

}