Place Autocomplete

> Looking to use this service in a JavaScript application? Check out the Places Library of the Google Maps API v3.

> **Note: The `id` and `reference` fields are deprecated** as of June 24, 2014. They are replaced by the new place ID, a textual identifier that uniquely identifies a place and can be used to retrieve information about the place. The Places API currently returns a `place_id` in all responses, and accepts a `placeid` in the Place Details request or `place_id` in the Place Delete request. Soon after June 24, 2015, the API will stop returning the `id` and `reference` fields in responses. Some time later, the API will no longer accept the `reference` in requests. We recommend that you update your code to use the new **place ID** instead of `id` and `reference` as soon as possible.

The Place Autocomplete service is a web service that returns place predictions in response to an HTTP request. The request specifies a textual search string and optional geographic bounds. The service can be used to provide autocomplete functionality for text-based geographic searches, by returning places such as businesses, addresses and points of interest as a user types.

Place Autocomplete Requests

> The Place Autocomplete service is part of the Google Places API and shares an API key and quotas with the Google Places API.

> **Note:** You can use Place Autocomplete even without a map. If you do show a map, it must be a Google map. When you display predictions from the Place Autocomplete service without a map, you must include the 'Powered by Google' logo.

The Place Autocomplete service can match on full words as well as substrings. Applications can therefore send queries as the user types, to provide on-the-fly place predictions.

The returned predictions are designed to be presented to the user to aid them in selecting the desired place. You can send a Place Details request for more information about any of the places which are returned.

A Place Autocomplete request is an HTTP URL of the following form:

```
https://maps.googleapis.com/maps/api/place/autocomplete/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

Certain parameters are required to initiate a Place Autocomplete request. As is standard in URLs, all parameters are separated using the ampersand (`&`) character. The list of parameters and their possible values are enumerated below.

Required parameters

- `input` — The text string on which to search. The Place Autocomplete service will return candidate matches based on this string and order results based on their perceived relevance.
- `key` — Your application's API key. This key identifies your application for purposes of quota management. Visit the Google Developers Console to select an API Project and obtain your key. Google Maps API for Work customers must use the API project created for them as part of their Google Places API for Work purchase.

Optional parameters

- `offset` — The position, in the input term, of the last character that the service uses to match predictions. For example, if the input is 'Google' and the `offset` is 3, the service will match on 'Goo'. The string determined by the `offset` is matched against the first word in the input term only. For example, if the input term is 'Google abc' and the `offset` is 3, the service will attempt to match against 'Goo abc'. If no `offset` is supplied, the service will use the whole term. The `offset` should generally be set to the position of the text caret.
- `location` — The point around which you wish to retrieve place information. Must be specified as *latitude*,*longitude*.
- `radius` — The distance (in meters) within which to return place results. Note that setting a `radius` biases results to the indicated area, but may not fully restrict results to the specified area. See Location Biasing below.
- `language` — The language in which to return results. See the supported list of domain languages. Note that we often update supported languages so this list may not be exhaustive. If language is not supplied, the Place Autocomplete service will attempt to use the native language of the domain from which the request is sent.
- `types` — The types of place results to return. See Place Types below. If no type is specified, all types will be returned.
- `components` — A grouping of places to which you would like to restrict your results. Currently, you can use `components` to filter by country. The country must be passed as a two character, ISO 3166-1 Alpha-2 compatible country code. For example: `components=country:fr` would restrict your results to places within France.

Location Biasing

You may bias results to a specified circle by passing a `location` and a `radius` parameter. This instructs the Place Autocomplete service to *prefer* showing results within that circle. Results outside of the defined area may still be displayed. You can use the `components` parameter to filter results to show only those places within a specified country.

> **Note:** If you do not supply the location and radius, the API will attempt to detect the user's location from their IP address, and will bias the results to that location. If you would prefer to have no location bias, set the `location` to '0,0' and `radius` to '20000000' (20 thousand kilometers), to encompass the entire world.

Tip: Establishment results generally do not rank highly enough to show in results when the search area is large. If you want establishments to appear in mixed establishment/geocode results, you can specify a smaller radius. Alternativley, use `types=establishment` to restrict results to establishments only.

Place Types

You may restrict results from a Place Autocomplete request to be of a certain type by passing a `types` parameter. The parameter specifies a type or a type collection, as listed in the supported types below. If nothing is specified, all types are returned. In general only a single type is allowed. The exception is that you can safely mix the `geocode` and `establishment` types, but note that this will have the same effect as specifying no types. The supported types are:

- `geocode` instructs the Place Autocomplete service to return only geocoding results, rather than business results. Generally, you use this request to disambiguate results where the location specified may be indeterminate.
- `address` instructs the Place Autocomplete service to return only geocoding results with a precise address. Generally, you use this request when you know the user will be looking for a fully specified address.
- `establishment` instructs the Place Autocomplete service to return only business results.
- the `(regions)` type collection instructs the Places service to return any result matching the following types:
  - `locality`
  - `sublocality`
  - `postal_code`
  - `country`
  - `administrative_area_level_1`
  - `administrative_area_level_2`
- the `(cities)` type collection instructs the Places service to return results that match `locality` or `administrative_area_level_3`.

Example Autocomplete Requests

A request for establishments containing the string "Amoeba" within an area centered in San Francisco, CA:

```
https://maps.googleapis.com/maps/api/place/autocomplete/xml?input=Amoeba&types=establishment&lo
cation=37.76999,-122.44696&radius=500&key=API_KEY
```

A request for addresses containing "Vict" with results in French:

```
https://maps.googleapis.com/maps/api/place/autocomplete/json?input=Vict&types=geocode&language=
fr&key=API_KEY
```

A request for cities containing "Vict" with results in Brazilian Portuguese:

```
https://maps.googleapis.com/maps/api/place/autocomplete/json?input=Vict&types=(cities)&language
=pt_BR&key=API_KEY
```

Note that you'll need to replace the API key in these examples with your own key.

Place Autocomplete Responses

Place Autocomplete responses are returned in the format indicated by the `output` flag within the request's URL path. The results below are indicative of what may be returned for a query with the following parameters:

```
input=Paris&types=geocode
```

JSON XML

```
{
  "status": "OK",
  "predictions" : [
      {
         "description" : "Paris, France",
         "id" : "691b237b0322f28988f3ce03e321ff72a12167fd",
         "matched_substrings" : [
            {
               "length" : 5,
               "offset" : 0
            }
         ],
         "place_id" : "ChIJD7fiBh9u5kcRYJSMaMOCCwQ",
         "reference" : "CjQlAAAA_KB6EEceSTfkteSSF6U0pvumHCoLUboRcDlAH05N1pZJLmOQbYmboEi0SwXBS
oI2EhAhj249tFDCVh4R-PXZkPK8GhTBmp_6_lWljaf1joVs1SH2ttB_tw",
         "terms" : [
            {
               "offset" : 0,
               "value" : "Paris"
            },
            {
               "offset" : 7,
               "value" : "France"
            }
         ],
         "types" : [ "locality", "political", "geocode" ]
      },
      {
         "description" : "Paris Avenue, Earlwood, New South Wales, Australia",
         "id" : "359a75f8beff14b1c94f3d42c2aabfac2afbabad",
         "matched_substrings" : [
            {
               "length" : 5,
               "offset" : 0
            }
         ],
         "place_id" : "ChIJrU3KAHG6EmsR5Uwfrk7azrI",
         "reference" : "CkQ2AAAARbzLE-tsSQPgwv8JKBaVtbjY48kInQo9tny0k07FOYb3Z_z_yDTFhQB_Ehpu-
IKhvj8Msdb1rJlX7xMr9kfOVRIQVuL4tOtx9L7U8pC0Zx5bLBoUTFbw9R2lTn_EuBayhDvugt8T0Oo",
         "terms" : [
            {
               "offset" : 0,
               "value" : "Paris Avenue"
            },
            {
               "offset" : 14,
               "value" : "Earlwood"
            },
            {
               "offset" : 24,
               "value" : "New South Wales"
            },
            {
               "offset" : 41,
               "value" : "Australia"
            }
         ],
         "types" : [ "route", "geocode" ]
      },
      {
         "description" : "Paris Street, Carlton, New South Wales, Australia",
         "id" : "bee539812eeda477dad282bcc8310758fb31d64d",
         "matched_substrings" : [
            {
               "length" : 5,
               "offset" : 0
            }
         ],
         "place_id" : "ChIJCfeffMi5EmsRp7ykjcnb3VY",
         "reference" : "CkQ1AAAAERlxMXkaNPLDxUJFLm4xkzX_h8I49HvGPvmtZjlYSVWp9yUhQSwfsdveHV0y
hzYki3nguTBTVX2NzmJDukq9RIQNcoFTuz642b4LIzmLgcr5RoUrZhuNqnFHegHsAjtoUUjmhy4_rA",
         "terms" : [
            {
               "offset" : 0,
               "value" : "Paris Street"
            },
            {
               "offset" : 14,
               "value" : "Carlton"
            },
            {
               "offset" : 23,
               "value" : "New South Wales"
            },
            {
               "offset" : 40,
               "value" : "Australia"
            }
         ],
         "types" : [ "route", "geocode" ]
      },
  ...additional results ...
```

A **JSON response** contains two root elements:

- `status` contains metadata on the request. See Status Codes below.
- `predictions` contains an array of places, with information about the place. See Place Autocomplete Results for information about these results. The Places API returns up to 5 results.

Of particular interest within the results are the `place_id` elements, which can be used to request more specific details about the place via a separate query. See Place Details Requests.

See Processing JSON with Javascript for help parsing JSON responses.

An **XML response** consists of a single `<AutocompletionResponse>` element with two types of child elements:

- A single `<status>` element contains metadata on the request. See Status Codes below.
- Zero or more `<prediction>` elements, each containing information about a single place. See Place Autocomplete Results for information about these results. The Places API returns up to 5 results.

We recommend that you use `json` as the preferred output flag unless your application requires `xml` for some reason. Processing XML trees requires some care, so that you reference proper nodes and elements. See Parsing XML with XPath for help processing XML.

Status Codes

The `status` field within the Place Autocomplete response object contains the status of the request, and may contain debugging information to help you track down why the Place Autocomplete request failed. The `status` field may contain the following values:

- `OK` indicates that no errors occurred and at least one result was returned.
- `ZERO_RESULTS` indicates that the search was successful but returned no results. This may occur if the search was passed a `bounds` 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 an invalid `key` parameter.
- `INVALID_REQUEST` generally indicates that the `input` parameter is missing.

Error Messages

When the Places service returns a status code other than `OK`, there may be an additional `error_message` field within the response object. This field contains more detailed information about the reasons behind the given status code.

> **Note:** This field is not guaranteed to be always present, and its content is subject to change.

Place Autocomplete Results

When the Places service returns JSON results from a search, it places them within a `predictions` array. Even if the service returns no results (such as if the `location` is remote) it still returns an empty `predictions` array. XML responses consist of zero or more `<prediction>` elements.

Each prediction result contains the following fields:

- `description` contains the human-readable name for the returned result. For `establishment` results, this is usually the business name.
- `place_id` is a textual identifier that uniquely identifies a place. To retrieve information about the place, pass this identifier in the `placeId` field of a Places API request. For more information about place IDs, see the place ID overview.
- `reference` contains a unique token that you can use to retrieve additional information about this place in a Place Details request. Although this token uniquely identifies the place, the converse is not true. A place may have many valid reference tokens. It's not guaranteed that the same token will be returned for any given place across different searches. **Note:** The `reference` is deprecated in favor of `place_id`. See the deprecation notice on this page.

- `id` contains a unique stable identifier denoting this place. This identifier may not be used to retrieve information about this place, but can be used to consolidate data about this place, and to verify the identity of a place across separate searches. **Note:** The `id` is deprecated in favor of `place_id`. See the deprecation notice on this page.
- `terms` contains an array of terms identifying each section of the returned description (a section of the description is generally terminated with a comma). Each entry in the array has a `value` field, containing the text of the term, and an `offset` field, defining the start position of this term in the description, measured in Unicode characters.
- `types` contains an array of types that apply to this place. For example: `[ "political", "locality" ]` or `[ "establishment", "geocode" ]`.
- `matched_substring` contains an `offset` value and a `length`. These describe the location of the entered term in the prediction result text, so that the term can be highlighted if desired.

> **Note:** The Place Autocomplete response does not include the `scope` or `alt_ids` fields that you may see in search results or place details. This is because Autocomplete returns only Google-scoped place IDs. It does not return app-scoped place IDs that have not yet been accepted into the Google Places database. For more details about Google-scoped and app-scoped place IDs, see the documentation on adding places.

The `sensor` Parameter

The Google Places API previously required that you include the `sensor` parameter to indicate whether your application used a sensor to determine the user's location. This parameter is no longer required.

Last updated .