Distance Matrix Service

Note: Server-side libraries

This page describes the client-side service available with the Maps JavaScript API. If you want to work with Google Maps web services on your server, take a look at the Node.js Client for Google Maps Services (/maps/web-services/client-library). The page at that link also introduces the Java Client, Python Client and Go Client for Google Maps Services.

Overview

Also see the Maps JavaScript API Reference: Distance Matrix (/maps/documentation/javascript/reference/distance-matrix)

Google's Distance Matrix service computes travel distance and journey duration between multiple origins and destinations using a given mode of travel.

This service does not return detailed route information. Route information, including polylines and textual directions, can be obtained by passing the desired single origin and destination to the Directions Service (/maps/documentation/javascript/directions).

Getting started

Before using the Distance Matrix service in the Maps JavaScript API, first ensure that the Distance Matrix API is enabled in the Google Cloud Console, in the same project you set up for the Maps JavaScript API.

To view your list of enabled APIs:

1. Go to the Google Cloud Console (https://console.cloud.google.com/project/_/apiui/apis/enabled?utm_source=Docs_EnabledAPIsView).
2. Click the **Select a project** button, then select the same project you set up for the Maps JavaScript API and click **Open**.

3. From the list of APIs on the **Dashboard**, look for **Distance Matrix API**.
4. If you see the API in the list, you're all set. If the API is *not* listed, enable it:
    a. At the top of the page, select **ENABLE API** to display the **Library** tab. Alternatively, from the left side menu, select **Library**.
    b. Search for **Distance Matrix API**, then select it from the results list.
    c. Select **ENABLE**. When the process finishes, **Distance Matrix API** appears in the list of APIs on the **Dashboard**.

Pricing and policies

Pricing

Effective July 16, 2018, a new pay-as-you-go pricing plan went into effect for Maps, Routes, and Places. To learn more about the new pricing and usage limits for your use of the JavaScript Distance Matrix service, see Usage and Billing (/maps/documentation/distance-matrix/usage-and-billing) for the Distance Matrix API.

Note: Each query sent to the Distance Matrix service is limited by the number of allowed elements, where the number of *origins* times the number of *destinations* defines the number of elements.

Policies

Use of the Distance Matrix service must be in accordance with the policies described for the Distance Matrix API (/maps/documentation/distance-matrix/policies).

Distance Matrix Requests

Accessing the Distance Matrix service is asynchronous, since the Google Maps API needs to make a call to an external server. For that reason, you need to pass a *callback* method to execute upon completion of the request, to process the results.

You access the Distance Matrix service within your code via the `google.maps.DistanceMatrixService` constructor object. The `DistanceMatrixService.getDistanceMatrix()` method initiates a request to the Distance Matrix service, passing it a `DistanceMatrixRequest` object literal containing the origins, destinations, and travel mode, as well as a callback method to execute upon receipt of the response.

```
var origin1 = new google.maps.LatLng(55.930385, -3.118425);
var origin2 = 'Greenwich, England';
var destinationA = 'Stockholm, Sweden';
var destinationB = new google.maps.LatLng(50.087692, 14.421150);

var service = new google.maps.DistanceMatrixService();
service.getDistanceMatrix(
  {
    origins: [origin1, origin2],
    destinations: [destinationA, destinationB],
    travelMode: 'DRIVING',
    transitOptions: TransitOptions,
    drivingOptions: DrivingOptions,
    unitSystem: UnitSystem,
    avoidHighways: Boolean,
    avoidTolls: Boolean,
  }, callback);

function callback(response, status) {
  // See Parsing the Results (#distance_matrix_parsing_the_results) for
  // the basics of a callback function.
}
```

View example (/maps/documentation/javascript/examples/distance-matrix)

The `DistanceMatrixRequest` contains the following fields:

- **`origins`** (required) — An array containing one or more address strings, **`google.maps.LatLng`** objects, or Place (/maps/documentation/javascript/reference/directions#Place) objects from which to calculate distance and time.
- **`destinations`** (required) — An array containing one or more address strings, **`google.maps.LatLng`** objects, or Place (/maps/documentation/javascript/reference/directions#Place) objects to which to calculate distance and time.
- **`travelMode`** (*optional*) — The mode of transport to use when calculating directions. See the section on travel modes (#travel_modes).
- **`transitOptions`** (*optional*) — Options that apply only to requests where **`travelMode`** is **`TRANSIT`**.

Valid values are described in the section on transit options (#transit_options).

- `drivingOptions` (*optional*) specifies values that apply only to requests where `travelMode` is `DRIVING`. Valid values are described in the section on Driving Options (#DrivingOptions).
- `unitSystem` (*optional*) — The unit system to use when displaying distance. Accepted values are:
    - `google.maps.UnitSystem.METRIC` (default)
    - `google.maps.UnitSystem.IMPERIAL`
- `avoidHighways` (*optional*) — If `true`, the routes between origins and destinations will be calculated to avoid highways where possible.
- `avoidTolls` (*optional*) — If `true`, the directions between points will be calculated using non-toll routes, wherever possible.

Note: The **`durationInTraffic`** field is now **deprecated**. It was previously the recommended way for Google Maps Platform Premium Plan customers to specify whether the result should include a duration that takes into account current traffic conditions. You should now use the **`drivingOptions`** field instead.

Travel Modes

When calculating times and distances, you can specify which transportation mode to use. The following travel modes are currently supported:

- `BICYCLING` requests bicycling directions via bicycle paths & preferred streets (currently only available in the US and some Canadian cities).
- `DRIVING` (default) indicates standard driving directions using the road network.
- `TRANSIT` requests directions via public transit routes. This option may only be specified if the request includes an API key. See the section on transit options (#transit_options) for the available options in this type of request.
- `WALKING` requests walking directions via pedestrian paths & sidewalks (where available).

Transit Options

The Transit Service is currently 'experimental'. During this phase, we will be implementing rate limits to prevent API abuse. We will eventually enforce a cap on total queries per map load based on fair usage of the API.

The available options for a distance matrix request vary across travel modes. In transit requests, the **avoidHighways** and **avoidTolls** options are ignored. You can specify transit-specific routing options through the **TransitOptions** (/maps/documentation/javascript/3.exp/reference#TransitOptions) object literal.

Transit requests are time sensitive. Calculations will only be returned for times in the future.

The **TransitOptions** object literal contains the following fields:

```
{
  arrivalTime: Date,
  departureTime: Date,
  modes: [transitMode1, transitMode2]
  routingPreference: TransitRoutePreference
}
```

These fields are explained below:

- **arrivalTime** (*optional*) specifies the desired time of arrival as a **Date** object. If arrival time is specified, departure time is ignored.
- **departureTime** (*optional*) specifies the desired time of departure as a **Date** object. The **departureTime** will be ignored if **arrivalTime** is specified. Defaults to now (that is, the current time) if no value is specified for either **departureTime** or **arrivalTime**.
- **modes** (*optional*) is an array containing one or more **TransitMode** object literals. This field may only be included if the request includes an API key. Each **TransitMode** specifies a preferred mode of transit. The following values are permitted:
  - **BUS** indicates that the calculated route should prefer travel by bus.
  - **RAIL** indicates that the calculated route should prefer travel by train, tram, light rail, and subway.
  - **SUBWAY** indicates that the calculated route should prefer travel by subway.
  - **TRAIN** indicates that the calculated route should prefer travel by train.
  - **TRAM** indicates that the calculated route should prefer travel by tram and light rail.
- **routingPreference** (*optional*) specifies preferences for transit routes. Using this option, you

can bias the options returned, rather than accepting the default best route chosen by the API. This field may only be specified if the request includes an API key. The following values are permitted:

- `FEWER_TRANSFERS` indicates that the calculated route should prefer a limited number of transfers.
- `LESS_WALKING` indicates that the calculated route should prefer limited amounts of walking.

Driving Options

Use the `drivingOptions` object to specify a departure time for calculating the best route to your destination given the expected traffic conditions. You can also specify whether you want the estimated time in traffic to be pessimistic, optimistic, or the best estimate based on historical traffic conditions and live traffic.

The `drivingOptions` object contains the following fields:

```
{
  departureTime: Date,
  trafficModel: TrafficModel
}
```

These fields are explained below:

- `departureTime` (*required for the `drivingOptions` object literal to be valid*) specifies the desired time of departure as a `Date` object. The value must be set to the current time or some time in the future. It cannot be in the past. (The API converts all dates to UTC to ensure consistent handling across time zones.) If you include the `departureTime` in the request, the API returns the best route given the expected traffic conditions at the time, and includes the predicted time in traffic (`duration_in_traffic`) in the response. If you don't specify a departure time (that is, if the request does not include `drivingOptions`), the returned route is a generally good route without taking traffic conditions into account.

  Note: If departure time is not specified, choice of route and duration are based on road network and average time-independent traffic conditions. Results for a given request may vary over time due to changes in the road network, updated average traffic conditions, and the distributed nature of the service. Results may also vary between nearly-equivalent routes at any time or frequency.

- `trafficModel` (*optional*) specifies the assumptions to use when calculating time in traffic. This setting affects the value returned in the `duration_in_traffic` field in the response, which contains the predicted time in traffic based on historical averages. Defaults to `best_guess`. The following values are permitted:
    - `bestguess` (default) indicates that the returned `duration_in_traffic` should be the best estimate of travel time given what is known about both historical traffic conditions and live traffic. Live traffic becomes more important the closer the `departureTime` is to now..
    - `pessimistic` indicates that the returned `duration_in_traffic` should be longer than the actual travel time on most days, though occasional days with particularly bad traffic conditions may exceed this value.
    - `optimistic` indicates that the returned `duration_in_traffic` should be shorter than the actual travel time on most days, though occasional days with particularly good traffic conditions may be faster than this value.

Below is a sample `DistanceMatrixRequest` for driving routes, including a departure time and traffic model:

```
{
  origins: [{lat: 55.93, lng: -3.118}, 'Greenwich, England'],
  destinations: ['Stockholm, Sweden', {lat: 50.087, lng: 14.421}],
  travelMode: 'DRIVING',
  drivingOptions: {
    departureTime: new Date(Date.now() + N),  // for the time N milliseconds from now.
    trafficModel: 'optimistic'
  }
}
```

Distance Matrix Responses

A successful call to the Distance Matrix service returns a `DistanceMatrixResponse` object and a `DistanceMatrixStatus` object. These are passed to the callback function you specified in the request.

The `DistanceMatrixResponse` object contains distance and duration information for each origin/destination pair for which a route could be calculated.

```
{
  "originAddresses": [ "Greenwich, Greater London, UK", "13 Great Carleton Square, Edi
  "destinationAddresses": [ "Stockholm County, Sweden", "Dlouhá 609/2, 110 00 Praha-St
  "rows": [ {
    "elements": [ {
      "status": "OK",
      "duration": {
        "value": 70778,
        "text": "19 hours 40 mins"
      },
      "distance": {
        "value": 1887508,
        "text": "1173 mi"
      }
    }, {
      "status": "OK",
      "duration": {
        "value": 44476,
        "text": "12 hours 21 mins"
      },
      "distance": {
        "value": 1262780,
        "text": "785 mi"
      }
    } ]
  }, {
    "elements": [ {
      "status": "OK",
      "duration": {
        "value": 96000,
        "text": "1 day 3 hours"
      },
      "distance": {
        "value": 2566737,
        "text": "1595 mi"
      }
    }, {
      "status": "OK",
      "duration": {
        "value": 69698,
        "text": "19 hours 22 mins"
      },
      "distance": {
        "value": 1942009,
        "text": "1207 mi"
      }
    } ]
  } ]
}
```

Distance Matrix Results

The supported fields in a response are explained below.

- **`originAddresses`** is an array containing the locations passed in the **`origins`** field of the Distance Matrix request. The addresses are returned as they are formatted by the geocoder.
- **`destinationAddresses`** is an array containing the locations passed in the **`destinations`** field, in the format returned by the geocoder.
- **`rows`** is an array of **`DistanceMatrixResponseRow`** objects, with each row corresponding to an origin.
- **`elements`** are children of **`rows`**, and correspond to a pairing of the row's origin with each destination. They contain status, duration, distance, and fare information (if available) for each origin/destination pair.
- Each **`element`** contains the following fields:
  - **`status`**: See Status Codes (#distance_matrix_status_codes) for a list of possible status codes.
  - **`duration`**: The length of time it takes to travel this route, expressed in seconds (the **`value`** field) and as **`text`**. The textual value is formatted according to the **`unitSystem`** specified in the request (or in metric, if no preference was supplied).
  - **`duration_in_traffic`**: The length of time it takes to travel this route taking into account current traffic conditions, expressed in seconds (the **`value`** field) and as **`text`**. The textual value is formatted according to the **`unitSystem`** specified in the request (or in metric, if no preference was supplied). The **`duration_in_traffic`** is only returned to Google Maps Platform Premium Plan customers where traffic data is available, the **`mode`** is set to **`driving`**, and **`departureTime`** is included as part of the **`distanceMatrixOptions`** field in the request.
  - **`distance`**: The total distance of this route, expressed in meters (**`value`**) and as **`text`**. The

textual value is formatted according to the `unitSystem` specified in the request (or in metric, if no preference was supplied).

- `fare`: Contains the total fare (that is, the total ticket costs) on this route. This property is only returned for transit requests and only for transit providers where fare information is available. The information includes:
  - `currency`: An ISO 4217 currency code (https://en.wikipedia.org/wiki/ISO_4217) indicating the currency that the amount is expressed in.
  - `value`: The total fare amount, in the currency specified above.

Status Codes

The Distance Matrix response includes a status code for the response as a whole, as well as a status for each element.

Response Status Codes

Status codes that apply to the `DistanceMatrixResponse` are passed in the `DistanceMatrixStatus` object and include:

- `OK` — The request is valid. This status can be returned even if no routes were found between any of the origins and destinations. See Element Status Codes (#distance_matrix_element_status_codes) for the element-level status information.
- `INVALID_REQUEST` — The provided request was invalid. This is often due to missing required fields. See the list of supported fields (#distance_matrix_results) above.
- `MAX_ELEMENTS_EXCEEDED` — The product of origins and destinations exceeds the per-query limit (/maps/documentation/distance-matrix/usage-and-billing#other-usage-limits).
- `MAX_DIMENSIONS_EXCEEDED` — Your request contained more than 25 origins, or more than 25 destinations.
- `OVER_QUERY_LIMIT` — Your application has requested too many elements within the allowed time period. The request should succeed if you try again after a reasonable amount of time.
- `REQUEST_DENIED` — The service denied use of the Distance Matrix service by your web page.
- `UNKNOWN_ERROR` — A Distance Matrix request could not be processed due to a server error. The request may succeed if you try again.

Element Status Codes

The following status codes apply to specific `DistanceMatrixElement` objects:

- `NOT_FOUND` — The origin and/or destination of this pairing could not be geocoded.
- `OK` — The response contains a valid result.
- `ZERO_RESULTS` — No route could be found between the origin and destination.

Parsing the Results

The `DistanceMatrixResponse` object contains one `row` for each origin that was passed in the request. Each row contains an `element` field for each pairing of that origin with the provided destination(s).

```
function callback(response, status) {
  if (status == 'OK') {
    var origins = response.originAddresses;
    var destinations = response.destinationAddresses;

    for (var i = 0; i < origins.length; i++) {
      var results = response.rows[i].elements;
      for (var j = 0; j < results.length; j++) {
        var element = results[j];
        var distance = element.distance.text;
        var duration = element.duration.text;
        var from = origins[i];
        var to = destinations[j];
      }
    }
  }
}
```

Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/), and code samples are licensed under the Apache 2.0 License (https://www.apache.org/licenses/LICENSE-2.0). For details, see the Google Developers Site Policies (https://developers.google.com/site-policies). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2023-08-31 UTC.