Job Search documentation

Package google.type

Index

- `Date` (https://cloud.google.com/talent-solution/job-search/docs/reference/rpc/google.type#google.type.Date) (message)
- `LatLng` (https://cloud.google.com/talent-solution/job-search/docs/reference/rpc/google.type#google.type.LatLng) (message)
- `Money` (https://cloud.google.com/talent-solution/job-search/docs/reference/rpc/google.type#google.type.Money) (message)
- `PostalAddress` (https://cloud.google.com/talent-solution/job-search/docs/reference/rpc/google.type#google.type.PostalAddress) (message)
- `TimeOfDay` (https://cloud.google.com/talent-solution/job-search/docs/reference/rpc/google.type#google.type.TimeOfDay) (message)

Date

Represents a whole or partial calendar date, e.g. a birthday. The time of day and time zone are either specified elsewhere or are not significant. The date is relative to the Proleptic Gregorian Calendar. This can represent:

- A full date, with non-zero year, month and day values
- A month and day value, with a zero year, e.g. an anniversary
- A year on its own, with zero month and day values
- A year and month value, with a zero day, e.g. a credit card expiration date

Related types are `google.type.TimeOfDay` (https://cloud.google.com/talent-solution/job-search/docs/reference/rpc/google.type#google.type.TimeOfDay) and `google.protobuf.Timestamp`.

Fields	
`year`	`int32` Year of date. Must be from 1 to 9999, or 0 if specifying a date without a year.
`month`	`int32` Month of year. Must be from 1 to 12, or 0 if specifying a year without a month and day.
`day`	`int32` Day of month. Must be from 1 to 31 and valid for the year and month, or 0 if specifying a year by itself or a year and month where the day is not significant.

LatLng

An object representing a latitude/longitude pair. This is expressed as a pair of doubles representing degrees latitude and degrees longitude. Unless specified otherwise, this must conform to the WGS84 standard (http://www.unoosa.org/pdf/icg/2012/template/WGS_84.pdf). Values must be within normalized ranges.

Fields	
`latitude`	`double` The latitude in degrees. It must be in the range [-90.0, +90.0].
`longitude`	`double` The longitude in degrees. It must be in the range [-180.0, +180.0].

Money

Represents an amount of money with its currency type.

Fields	
`currency_code`	`string` The 3-letter currency code defined in ISO 4217.
`units`	`int64` The whole units of the amount. For example if **`currencyCode`** is `"USD"`, then 1 unit is one US dollar.
`nanos`	`int32` Number of nano (10^-9) units of the amount. The value must be between -999,999,999 and +999,999,999 inclusive. If **`units`** is positive, **`nanos`** must be positive or zero. If **`units`** is zero, **`nanos`** can be positive, zero, or negative. If **`units`** is negative, **`nanos`** must be negative or zero. For example $-1.75 is represented as **`units`**=-1 and **`nanos`**=-750,000,000.

PostalAddress

Represents a postal address, e.g. for postal delivery or payments addresses. Given a postal address, a postal service can deliver items to a premise, P.O. Box or similar. It is not intended to model geographical locations (roads, towns, mountains).

In typical usage an address would be created via user input or from importing existing data, depending on the type of process.

Advice on address input / editing: - Use an i18n-ready address widget such as https://github.com/google/libaddressinput (https://github.com/google/libaddressinput)) - Users should not be presented with UI elements for input or editing of fields outside countries where that field is used.

For more guidance on how to use this schema, please see: https://support.google.com/business/answer/6397478 (https://support.google.com/business/answer/6397478)

Fields	
`revision`	`int32` The schema revision of the **`PostalAddress`**. This must be set to 0, which is latest revision. All new revisions **must** be backward compatible with old revisions.
`region_code`	`string` Required. CLDR region code of the country/region of the address. This is n inferred and it is up to the user to ensure the value is correct. See http://cldr.unicode.org/ (http://cldr.unicode.org/) and http://www.unicode.org/cldr/charts/30/supplemental/territory_informatic (http://www.unicode.org/cldr/charts/30/supplemental/territory_informat for details. Example: "CH" for Switzerland.
`language_code`	`string` Optional. BCP-47 language code of the contents of this address (if known) often the UI language of the input form or is expected to match one of the languages used in the address' country/region, or their transliterated equiv This can affect formatting in certain countries, but is not critical to the corr of the data and will never affect any validation or other non-formatting rela operations. If this value is not known, it should be omitted (rather than specifying a pos incorrect default). Examples: "zh-Hant", "ja", "ja-Latn", "en".

Fields	
`postal_code`	`string` Optional. Postal code of the address. Not all countries use or require posta to be present, but where they are used, they may trigger additional validati other parts of the address (e.g. state/zip validation in the U.S.A.).
`sorting_code`	`string` Optional. Additional, country-specific, sorting code. This is not used in mos Where it is used, the value is either a string like "CEDEX", optionally followed number (e.g. "CEDEX 7"), or just a number alone, representing the "sector c (Jamaica), "delivery area indicator" (Malawi) or "post office indicator" (e.g. d'Ivoire).
`administrative_area`	`string` Optional. Highest administrative subdivision which is used for postal addre country or region. For example, this can be a state, a province, an oblast, o prefecture. Specifically, for Spain this is the province and not the autonom community (e.g. "Barcelona" and not "Catalonia"). Many countries don't us administrative area in postal addresses. E.g. in Switzerland this should be unpopulated.
`locality`	`string` Optional. Generally refers to the city/town portion of the address. Example IT comune, UK post town. In regions of the world where localities are not w defined or do not fit into this structure well, leave locality empty and use address_lines.
`sublocality`	`string` Optional. Sublocality of the address. For example, this can be neighborhoo boroughs, districts.

Fields	
`address_lines[]`	`string` Unstructured address lines describing the lower levels of an address. Because values in address_lines do not have type information and may sor contain multiple values in a single field (e.g. "Austin, TX"), it is important tha order is clear. The order of address lines should be "envelope order" for the country/region of the address. In places where this can vary (e.g. Japan), address_language is used to make it explicit (e.g. "ja" for large-to-small ord "ja-Latn" or "en" for small-to-large). This way, the most specific line of an ad be selected based on the language. The minimum permitted structural representation of an address consists o region_code with all remaining information placed in the address_lines. It w possible to format such an address very approximately without geocoding semantic reasoning could be made about any of the address components was at least partially resolved. Creating an address only containing a region_code and address_lines, and geocoding is the recommended way to handle completely unstructured ad (as opposed to guessing which parts of the address should be localities or administrative areas).
`recipients[]`	`string` Optional. The recipient at the address. This field may, under certain circums contain multiline information. For example, it might contain "care of" inform
`organization`	`string` Optional. The name of the organization at the address.

TimeOfDay

Represents a time of day. The date and time zone are either not significant or are specified elsewhere. An API may choose to allow leap seconds. Related types are `google.type.Date` (https://cloud.google.com/talent-solution/job-search/docs/reference/rpc/google.type#google.type.Date) and `google.protobuf.Timestamp`.

Fields	
`hours`	`int32` Hours of day in 24 hour format. Should be from 0 to 23. An API may choose to allow the value "24:00:00" for scenarios like business closing time.
`minutes`	`int32` Minutes of hour of day. Must be from 0 to 59.
`seconds`	`int32` Seconds of minutes of the time. Must normally be from 0 to 59. An API may allow the value 60 if it allows leap-seconds.
`nanos`	`int32` Fractions of seconds in nanoseconds. Must be from 0 to 999,999,999.

Last updated July 3, 2019.