Job Search documentation (https://cloud.google.com/talent-solution/job-search/)

Documentation

Commute search

Search requests can include a commute time filter that restricts returned jobs to those within a specified travel time from a start point. Results include the estimated commute time in seconds for matched jobs.

Specific commute time search

To return jobs within a specific commute time, send a `jobs.search` (https://cloud.google.com/talent-solution/job-search/docs/reference/rest/v4beta1/projects.jobs/search) request and include a `CommuteFilter` object in the `JobQuery.commuteFilter` field. Cloud Talent Solution uses the job's `address` (https://cloud.google.com/talent-solution/job-search/docs/reference/rest/v4beta1/projects.jobs#Job.FIELDS.addresses) to calculate the commute time to that job. When a detailed address is not provided Cloud Talent Solution attempts to infer the actual street address of the job.

GO JAVA PHP PYTHON RUBY

JDPLATFORM/GOLANG-SAMPLES/BLOB/MASTER/JOBS/V4/HOWTO/COMMUTE_SEARCH_SAMPLE.GO)

FEEDBACK (#)

```
import (
        "context"
        "fmt"
        "io"
        talent "cloud.google.com/go/talent/apiv4beta1"
        "github.com/golang/protobuf/ptypes/duration"
        "google.golang.org/api/iterator"
        talentpb "google.golang.org/genproto/googleapis/cloud/talent/v4beta1"
        "google.golang.org/genproto/googleapis/type/latlng"
)

// commuteSearch searches for jobs within commute filter.
func commuteSearch(w io.Writer, projectID, companyID string) error {
        ctx := context.Background()

        // Initialize a jobService client.
        c, err := talent.NewJobClient(ctx)
        if err != nil {
                return fmt.Errorf("talent.NewJobClient: %v", err)
        }

        // Construct a jobQuery request with a commute filter.
        jobQuery := &talentpb.JobQuery{
                CommuteFilter: &talentpb.CommuteFilter{
                        CommuteMethod:  talentpb.CommuteMethod_TRANSIT,
                        TravelDuration: &duration.Duration{Seconds: 1800},
                        StartCoordinates: &latlng.LatLng{
                                Latitude:  37.422408,
                                Longitude: -122.085609,
                        },
                },
        }
        if companyID != "" {
                jobQuery.Companies = []string{fmt.Sprintf("projects/%s/companies/%
        }

        // Construct a searchJobs request with a jobQuery.
        req := &talentpb.SearchJobsRequest{
                Parent: fmt.Sprintf("projects/%s", projectID),
                // Make sure to set the RequestMetadata the same as the associated
                // search request.
                RequestMetadata: &talentpb.RequestMetadata{
                        // Make sure to hash your userID.
                        UserId: "HashedUsrID",
                        // Make sure to hash the sessionID.
                        SessionId: "HashedSessionID",
                        // Domain of the website where the search is conducted.
                        Domain: "www.googlesample.com",
                },
                // Set the actual search term as defined in the jobQuery.
                JobQuery: jobQuery,
        }

        it := c.SearchJobs(ctx, req)

        for {
                resp, err := it.Next()
                if err == iterator.Done {
                        return nil
                }
                if err != nil {
                        return fmt.Errorf("it.Next: %v", err)
                }
                fmt.Fprintf(w, "Matcing job: %q\n", resp.GetJob().GetName())
                fmt.Fprintf(w, "Job address: %v\n", resp.GetCommuteInfo().GetJobLo
        }
}
```

Required fields

- `commuteMethod`: The method of transportation used to calculate the commute time. Options are `DRIVING`, `TRANSIT`, `WALKING` and `CYCLING` transit modes. **Walking and cycling routes may not reflect real-world conditions such as construction or include clear walking or cycling paths. These responses include `warnings` in the returned result which you must display to your users.**
- `travelDuration`: The maximum travel time in seconds. The maximum allowed value is `3600s` (one hour). The format is `123s`.
- `startCoordinates`: The latitude and longitude of the location from which to calculate the commute time. This field accepts a `LatLng` (https://cloud.google.com/talent-solution/job-search/docs/reference/rest/v4beta1/Location#LatLng) object.
- Either `roadTraffic` or `departureTime`: Commute search results are based on historical and aggregated data rather than live traffic conditions. Users receive the same commute search results no matter what time of day they send a query.

- `roadTraffic`: Specifies the traffic density to use when calculating commute time. The `BUSY_HOUR`/`TRAFFIC_FREE` options are the calculated average traffic conditions at morning rush hour and midnight, respectively. This field must not be present if `departureTime` is specified.
- `departureTime`: The departure hour to use to calculate traffic impact. Accepts an integer between 0 and 23, representing the hour in the timezone of the `startLocation`. Traffic conditions are calculated from average historical traffic conditions at the specified time of day. This field must not be present if `roadTraffic` is specified.

Optional fields

- `allowImpreciseAddresses`: "Precise" addresses are defined as either street level addresses or GPS coordinates. If `allowImpreciseAddresses` is set to `true`, jobs with "imprecise" addresses (city, state, or country only) might also be returned. For city level and coarser level addresses, the API uses text matching to return results. If this field is set to `false` or is not specified, only jobs that include precise addresses are returned.

Note: If `allowImpreciseAddresses` is set to `true`, Job Search is not able to calculate accurate commute times to jobs with city level and coarser address information. Jobs with imprecise addresses return a `travelDuration` time of 0 regardless of their distance from the job seeker.

Last updated December 13, 2019.