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

Documentation

Job basics

A Job (https://cloud.google.com/talent-solution/job-search/docs/reference/rest/v4beta1/projects.jobs) resource represents a single job posting (also referred to as a "job listing" or "job requisition"). A job belongs to a Company (https://cloud.google.com/talent-solution/job-search/docs/reference/rest/v4beta1/projects.companies) resource that represents the hiring entity responsible for the job.

You can access a job using the LIST and GET methods and manipulate it using CREATE, UPDATE, and DELETE methods. It can take several minutes for the Cloud Talent Solution index to reflect changes.

Jobs are contained within the scope of a service account. Only search requests authenticated using the credentials of a particular service account can be used to access the content of these jobs.

For easy troubleshooting and triaging, synchronize the Cloud Talent Solution job index with your own job index, and maintain a relationship between the `name` generated by Cloud Talent Solution and the unique job identifier in your system. As jobs change or are introduced into your system, the appropriate CRUD call should be sent to CTS in real time to ensure that these changes are reflected immediately. The CTS index must be added to the existing job ingestion pipeline.

Create a job

You can create a Job using the code sample below. See Quickstart: Create companies and jobs (https://cloud.google.com/talent-solution/job-search/docs/quickstart-jobs-and-companies#create_a_job_in_your_company) for more details.

GO JAVA PHP PYTHON RUBY

LOB/5FF9596979C3431FF123D7CB91712BC357E380AB/JOBS/V4/HOWTO/CREATE_JOB_SAMPLE.GO)

FEEDBACK (#)

```
import (
        "context"
        "fmt"
        "io"

        talent "cloud.google.com/go/talent/apiv4beta1"
        talentpb "google.golang.org/genproto/googleapis/cloud/talent/v4beta1"
)

// createJob create a job as given.
func createJob(w io.Writer, projectID, companyID, requisitionID, title, URI, descr
        ctx := context.Background()

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

        jobToCreate := &talentpb.Job{
                CompanyName:   fmt.Sprintf("projects/%s/companies/%s", projectID,
                RequisitionId: requisitionID,
                Title:         title,
                ApplicationInfo: &talentpb.Job_ApplicationInfo{
                        Uris: []string{URI},
                },
                Description:  description,
                Addresses:    []string{address1, address2},
                LanguageCode: languageCode,
        }

        // Construct a createJob request.
        req := &talentpb.CreateJobRequest{
                Parent: fmt.Sprintf("projects/%s", projectID),
                Job:    jobToCreate,
        }

        resp, err := c.CreateJob(ctx, req)
        if err != nil {
                fmt.Printf("Failed to create job: %v\n", err)
                return nil, err
        }

        fmt.Printf("Created job: %q\n", resp.GetName())

        return resp, nil
}
```

Required fields

The following fields are required during Job creation and update:

- `companyName`: The resource name of the company that owns the job, such as `projects/[project_id]/tenants/[tenant_id]/companies/[company_id]`.
- `requisitionId`: The requisition ID, also referred to as the posting ID, is a value that you assign to identify a job. You can use this field for client identification and requisition tracking. The maximum number of allowed characters is 225.

  The uniqueness of a job posting is determined using a combination of the `requisitionID`, the `companyName`, and location. If a job is created with a specific key of these attributes, this key is stored in the Cloud Talent Solution index and no other jobs with these same fields can be created until the job is deleted.
- `title`: The job title, for example "Software Engineer." The maximum number of allowed characters is 500.

  To fix the problem of missed search results due to non-standard job titles, Cloud Talent Solution leverages all job fields to understand the context of the job and internally store a "clean" job title. When a search request is sent to the service, the query of the search is also cleaned, and the ontologies are used to map the cleaned query to relevant clean jobs.
- `description`: The description of the job, which typically includes a multi-paragraph description of the company and related information. Separate fields are provided on the Job object for responsibilities, qualifications, and other job characteristics. Use of these separate fields is recommended.

This field accepts and sanitizes HTML input, and accepts bold, italic, ordered list, and unordered list markup tags. The maximum number of allowed characters is 100,000.

One of the following:

- `applicationInfo.uris` (https://cloud.google.com/talent-solution/job-search/docs/reference/rest/v4beta1/projects.jobs#ApplicationInfo.FIELDS.uris) : The URL(s) of the application page(s).
- `applicationInfo.emails` (https://cloud.google.com/talent-solution/job-search/docs/reference/rest/v4beta1/projects.jobs#ApplicationInfo.FIELDS.emails) : Email address(es) to which resumes or applications should be sent.
- `applicationInfo.instruction` (https://cloud.google.com/talent-solution/job-search/docs/reference/rest/v4beta1/projects.jobs#ApplicationInfo.FIELDS.instruction) : Application instructions, such as "Mail your application to ...". This field accepts and sanitizes HTML input, and accepts bold, italic, ordered list, and unordered list markup tags. The maximum number of allowed characters is 3,000.

Commonly used fields

- `postingExpireTime`: The time, based on the timestamp, when the job posting expires. After this time occurs, the job is marked as expired and won't appear in search results. This date should be before 2100/12/31 in UTC timezone. Invalid dates (such as past dates) are ignored. The default date when the job expires is 30 days after the job creation time in UTC time zone.

  The content of expired jobs can still be retrieved up to 60 days after the job has expired by using the GET operator. After this 60 day deadline, the job won't be returned through a GET operation.

★ **Note:** The 60 day limit is configurable by customer support. File a ticket with the support team if you need to extend access.

- `addresses`: Job location(s). Providing the full street address(es) of the hiring location is recommended to enable better Job Search results, including searches by commute time.

The maximum number of allowed characters is 500. More information about `addresses` is available in the Best practices (#best_practices) section below.

- `promotionValue`: A value greater than 0 defines this job as a "featured job", which is returned only in searches of type `FEATURED_JOBS`. Higher values are returned higher in the featured search results. See Featured Jobs (https://cloud.google.com/talent-solution/job-search/docs/featured-jobs) for more information.

Using custom job fields

Cloud Talent Solution includes several job fields that are built in to its API schemas. However, you may need additional fields that aren't present in the out-of-the-box options. While we recommended that you use the out-of-the box fields wherever possible, Cloud Talent Solution also provides some `customAttributes` fields for a job. These can be filterable or non-filterable. Refer to the `customAttributes` (https://cloud.google.com/talent-solution/job-search/docs/reference/rest/v4beta1/projects.jobs#Job.FIELDS.custom_attributes) documentation for more information.

- `customAttributes` (https://cloud.google.com/talent-solution/job-search/docs/custom-attributes): This field stores up to 100 custom attributes used to store custom data about the job. These fields can be filtered against using a search request specifying the `jobQuery` field. Additionally, any of these fields can be set in the `keywordSearchableJobCustomAttributes` attribute of the `company`, so a search term that has an exact match in any of the fields in `keywordSearchableJobCustomAttributes` returns any Job that includes the match.

The following code example shows how to create a job with a `customAttribute`:

GO JAVA PHP PYTHON RUBY

For more on installing and creating a Cloud Talent Solution client, see Cloud Talent Solution Client Libraries (https://cloud.google.com/talent-solution/job-search/docs/libraries).

DPLATFORM/GOLANG-SAMPLES/BLOB/MASTER/JOBS/V4/HOWTO/CUSTOM_ATTRIBUTE_SAMPLE.GO)

FEEDBACK (#)

```
import (
        "context"
        "fmt"
        "io"

        talent "cloud.google.com/go/talent/apiv4beta1"
        "github.com/gofrs/uuid"
        talentpb "google.golang.org/genproto/googleapis/cloud/talent/v4beta1"
        money "google.golang.org/genproto/googleapis/type/money"
)

// createJobWithCustomAttributes creates a job with custom attributes.
func createJobWithCustomAttributes(w io.Writer, projectID, companyID, jobTitle str
        ctx := context.Background()

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

        // requisitionID shoud be the unique ID in your system
        requisitionID := fmt.Sprintf("job-with-custom-attribute-%s", uuid.Must(uui
        jobToCreate := &talentpb.Job{
                Company:       fmt.Sprintf("projects/%s/companies/%s", projectID,
                RequisitionId: requisitionID,
                Title:         jobTitle,
                ApplicationInfo: &talentpb.Job_ApplicationInfo{
                        Uris: []string{"https://googlesample.com/career"},
                },
                Description:     "Design, devolop, test, deploy, maintain and impr
                LanguageCode:    "en-US",
                PromotionValue:  2,
                EmploymentTypes: []talentpb.EmploymentType{talentpb.EmploymentType
                Addresses:       []string{"Mountain View, CA"},
                CustomAttributes: map[string]*talentpb.CustomAttribute{
                        "someFieldString": {
                                Filterable:   true,
                                StringValues: []string{"someStrVal"},
                        },
                        "someFieldLong": {
                                Filterable: true,
                                LongValues: []int64{900},
                        },
                },
                CompensationInfo: &talentpb.CompensationInfo{
                        Entries: []*talentpb.CompensationInfo_CompensationEntry{
                                {
                                        Type: talentpb.CompensationInfo_BASE,
                                        Unit: talentpb.CompensationInfo_HOURLY,
                                        CompensationAmount: &talentpb.Compensation
                                                Amount: &money.Money{
                                                        CurrencyCode: "USD",
                                                        Units:        1,
                                                },
                                        },
                                },
                        },
                },
        }

        // Construct a createJob request.
        req := &talentpb.CreateJobRequest{
                Parent: fmt.Sprintf("projects/%s", projectID),
                Job:    jobToCreate,
        }

        resp, err := c.CreateJob(ctx, req)
        if err != nil {
                return nil, fmt.Errorf("CreateJob: %v", err)
        }

        fmt.Fprintf(w, "Created job with custom attributres: %q\n", resp.GetName()
        fmt.Fprintf(w, "Custom long field has value: %v\n", resp.GetCustomAttribut

        return resp, nil
}
```

Retrieve a job

GO JAVA PHP PYTHON RUBY

For more on installing and creating a Cloud Talent Solution client, see Cloud Talent Solution Client Libraries (https://cloud.google.com/talent-solution/job-search/docs/libraries).

S/BLOB/5FF9596979C3431FF123D7CB91712BC357E380AB/JOBS/V4/HOWTO/GET_JOB_SAMPLE.GO)

FEEDBACK (#)

```go
import (
	"context"
	"fmt"
	"io"

	talent "cloud.google.com/go/talent/apiv4beta1"
	talentpb "google.golang.org/genproto/googleapis/cloud/talent/v4beta1"
)

// getJob gets an existing job by its resource name.
func getJob(w io.Writer, projectID, jobID string) (*talentpb.Job, error) {
	ctx := context.Background()

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

	// Construct a getJob request.
	jobName := fmt.Sprintf("projects/%s/jobs/%s", projectID, jobID)
	req := &talentpb.GetJobRequest{
		// The resource name of the job to retrieve.
		// The format is "projects/{project_id}/jobs/{job_id}".
		Name: jobName,
	}

	resp, err := c.GetJob(ctx, req)
	if err != nil {
		fmt.Printf("Failed to get job %s: %v\n", jobName, err)
		return nil, err
	}

	fmt.Fprintf(w, "Job: %q\n", resp.GetName())
	fmt.Fprintf(w, "Job title: %v\n", resp.GetTitle())

	return resp, err
}
```

List jobs

GO JAVA PHP PYTHON RUBY

For more on installing and creating a Cloud Talent Solution client, see Cloud Talent Solution Client Libraries (https://cloud.google.com/talent-solution/job-search/docs/libraries).

S/BLOB/5FF9596979C3431FF123D7CB91712BC357E380AB/JOBS/V4/HOWTO/LIST_JOB_SAMPLE.GO)

FEEDBACK (#)

```
import (
        "context"
        "fmt"
        "io"

        talent "cloud.google.com/go/talent/apiv4beta1"
        "google.golang.org/api/iterator"
        talentpb "google.golang.org/genproto/googleapis/cloud/talent/v4beta1"
)

// listJobs lists jobs with a filter, for example
// `companyName="projects/my-project/companies/123"`.
func listJobs(w io.Writer, projectID, companyID string) error {
        ctx := context.Background()

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

        // Construct a listJobs request.
        companyName := fmt.Sprintf("projects/%s/companies/%s", projectID, companyI
        req := &talentpb.ListJobsRequest{
                Parent: "projects/" + projectID,
                Filter: fmt.Sprintf("companyName=%q", companyName),
        }

        it := c.ListJobs(ctx, req)

        for {
                resp, err := it.Next()
                if err == iterator.Done {
                        return nil
                }
                if err != nil {
                        fmt.Printf("it.Next: %v\n", err)
                        return err
                }
                fmt.Fprintf(w, "Listing job: %v\n", resp.GetName())
                fmt.Fprintf(w, "Job title: %v\n", resp.GetTitle())
        }
}
```

Delete a job

GO JAVA PHP PYTHON RUBY

For more on installing and creating a Cloud Talent Solution client, see Cloud Talent Solution Client Libraries (https://cloud.google.com/talent-solution/job-search/docs/libraries).

LOB/5FF9596979C3431FF123D7CB91712BC357E380AB/JOBS/V4/HOWTO/DELETE_JOB_SAMPLE.GO)

FEEDBACK (#)

```
import (
        "context"
        "fmt"
        "io"

        talent "cloud.google.com/go/talent/apiv4beta1"
        talentpb "google.golang.org/genproto/googleapis/cloud/talent/v4beta1"
)

// deleteJob deletes an existing job by its resource name.
func deleteJob(w io.Writer, projectID, jobID string) error {
        ctx := context.Background()

        // Initialize a jobService client.
        c, err := talent.NewJobClient(ctx)
        if err != nil {
                fmt.Printf("talent.NewJobClient: %v\n", err)
                return err
        }
        // Construct a deleteJob request.
        jobName := fmt.Sprintf("projects/%s/jobs/%s", projectID, jobID)
        req := &talentpb.DeleteJobRequest{
                // The resource name of the job to be deleted.
                // The format is "projects/{project_id}/jobs/{job_id}".
                Name: jobName,
        }

        if err := c.DeleteJob(ctx, req); err != nil {
                fmt.Printf("Delete(%s): %v\n", jobName, err)
                return err
        }

        fmt.Printf("Deleted job: %q\n", jobName)

        return err
}
```

Best practices

Location fields

Whenever possible, we recommend providing the street address of a job in the `addresses` field. This helps with location detection and relevance. When a street-level address isn't available, enter as much information as possible. Addresses are supported up to the country level. Region designations (such as "Pacific Northwest") are not supported.

Cloud Talent Solution uses the data in the `addresses` (https://cloud.google.com/talent-solution/job-search/docs/reference/rest/v4beta1/projects.jobs#Job.FIELDS.addresses) field to populate the (output only) `derivedInfo.locations` (https://cloud.google.com/talent-solution/job-search/docs/reference/rest/v4beta1/projects.jobs#DerivedInfo.FIELDS.locations) field. When a full address isn't provided, the service uses other signals, such as the company name, to determine if a more complete address can be inferred for the job posting.

For example, if the location of a software position is specified as `Mountain View`, and the company to which the job is associated is `Google`, the service looks up the `company` object to see if a better street address is provided in the `headquartersAddress` field and whether that street address is in the same city as the job posting. If so, the service understands that the job is "likely" located at that street address and fills in the `derivedInfo.locations` (https://cloud.google.com/talent-solution/job-search/docs/reference/rest/v4beta1/projects.jobs#DerivedInfo.FIELDS.locations) field appropriately.

If company address data isn't available, the service uses a combination of proprietary knowledge and job/company information to inform the `derivedInfo.locations` field.

Because the `derivedInfo.locations` value is a best-guess effort, you may wish to use the `derivedInfo.locations` data, or the `addresses` field, when displaying the job address.

A job posting may have no more than 50 locations associated with it. If a job has more locations you can split the job into multiple jobs, each with a unique requisitionId (for example, 'ReqA' , 'ReqA-1', 'ReqA-2', and so on). Having multiple jobs with the same `requisitionId`, , `companyName` and `languageCode` is not allowed. If the original `requisitionId` must be preserved, a `CustomAttribute` (https://cloud.google.com/talent-solution/job-search/docs/reference/rest/v3/projects.jobs#CustomAttribute) should be used for storage. It's recommended you group the locations closest to each other in the same job for a better search experience.

Supported addresses

Any address that is recognized by the Google Maps Geocoding API (https://developers.google.com/maps/documentation/geocoding/intro) (in the `formattedAddress` field) is accepted by Cloud Talent Solution. The service returns a 400 error if you attempt to create a Job or execute a search using an unrecognized address.

If a business address is listed incorrectly in the Google Maps Geocoding API, file a bug (https://developers.google.com/maps/documentation/geocoding/support#issue-tracker) to have it corrected. Corrections may take up to 5 days to take effect.

Address autocompletion

Cloud Talent Solution does not provide autocomplete suggestions for locations. Use the Google Maps Places API (https://developers.google.com/places/web-service/autocomplete), or other similar location services, to populate autocomplete suggestions.

Statewide, nationwide, and telecommute jobs

Jobs can be specified as statewide, nationwide, or telecommute by using the `postingRegion` (https://cloud.google.com/talent-solution/job-search/docs/reference/rest/v4beta1/projects.jobs#PostingRegion) field.

- `ADMINISTRATIVE_AREA` and `NATION` Jobs are returned for any search with a specified location inside the state/country of the job posting. For example, if an `ADMINISTRATIVE_AREA` job has a location of "WA, USA" it is returned for searches with `LocationFilter` specifing "Seattle".
- `TELECOMMUTE` Jobs are returned in any location-related search, but are treated as less relevant. They can be targeted in a search by setting the `telecommutePreference` flag to `TELECOMMUTE_ALLOWED` in the search's `LocationFilter` (https://cloud.google.com/talent-solution/job-search/docs/reference/rest/v4beta1/JobQuery#LocationFilter) .