
google-cloud Documentation

Release 0.27.1

Google Cloud Platform

Sep 15, 2017

1	Configuration	1
1.1	Overview	1
1.2	Authentication	2
2	Authentication	3
2.1	Overview	3
2.2	Client-Provided Authentication	3
2.3	Explicit Credentials	4
2.4	Troubleshooting	5
3	Long-Running Operations	7
4	Shared Core Modules	9
4.1	Base Client	9
4.2	Exceptions	10
4.3	Environment Variables	11
4.4	IAM Support	11
5	BigQuery	15
5.1	Client	15
5.2	Datasets	18
5.3	Jobs	22
5.4	Query	38
5.5	Schemas	41
5.6	Tables	42
5.7	Authentication / Configuration	48
5.8	Projects	48
5.9	Datasets	49
5.10	Tables	50
5.11	Jobs	51
6	Bigtable	59
6.1	Base for Everything	59
6.2	Client	60
6.3	Cluster	62
6.4	Instance	64
6.5	Instance Admin API	67

6.6	Table	69
6.7	Table Admin API	72
6.8	Column Families	74
6.9	Bigtable Row	76
6.10	Row Data	84
6.11	Bigtable Row Filters	86
6.12	Data API	94
7	Datastore	99
7.1	Datastore Client	99
7.2	Entities	102
7.3	Keys	104
7.4	Queries	107
7.5	Transactions	110
7.6	Batches	112
7.7	Helpers	114
7.8	Modules	115
8	DNS	131
8.1	DNS Client	131
8.2	Managed Zones	132
8.3	Resource Record Sets	135
8.4	Change Sets	136
8.5	Client	137
8.6	Projects	138
8.7	Project Quotas	138
8.8	Managed Zones	138
8.9	Resource Record Sets	139
8.10	Change requests	139
9	Natural Language	141
9.1	Authentication and Configuration	141
9.2	Documents	141
9.3	Analyze Entities	142
9.4	Analyze Sentiment	143
9.5	Annotate Text	144
9.6	API Reference	144
10	Pub/Sub	163
10.1	Authentication and Configuration	163
10.2	Publishing	164
10.3	Subscribing	164
10.4	Learn More	164
11	Resource Manager	197
11.1	Client	197
11.2	Projects	199
11.3	Authentication	202
12	Runtimeconfig	203
12.1	Runtime Configuration Client	203
12.2	Configuration	204
12.3	Variables	206
12.4	Modules	208

13	Spanner	209
13.1	Client	209
13.2	Instance Admin API	210
13.3	Database Admin API	212
13.4	Non-Admin Database Usage	213
13.5	Batching Modifications	215
13.6	Read-only Transactions via Snapshots	218
13.7	Read-write Transactions	220
13.8	Advanced Session Pool Topics	223
13.9	Spanner Client	224
13.10	Instance API	227
13.11	Database API	230
13.12	Session API	233
13.13	Session Pools API	236
13.14	Keyset API	240
13.15	Snapshot API	241
13.16	Batch API	241
13.17	Transaction API	242
13.18	StreamedResultSet API	242
14	Speech	245
14.1	Authentication and Configuration	245
14.2	Asynchronous Recognition	245
14.3	Synchronous Recognition	246
14.4	Streaming Recognition	247
14.5	API Reference	250
15	Stackdriver Error Reporting	259
15.1	Error Reporting Client	259
15.2	Error Reporting Utilities	261
15.3	Authentication and Configuration	261
15.4	Reporting an exception	262
15.5	Reporting an error without an exception	263
16	Stackdriver Monitoring	265
16.1	Stackdriver Monitoring Client	265
16.2	Metric Descriptors	271
16.3	Monitored Resource Descriptors	273
16.4	Groups	274
16.5	Time Series Query	277
16.6	Time Series	282
16.7	Label Descriptors	283
16.8	Introduction	284
16.9	The Stackdriver Monitoring Client Object	284
16.10	Monitored Resource Descriptors	285
16.11	Metric Descriptors	285
16.12	Groups	286
16.13	Time Series Queries	287
16.14	Writing Custom Metrics	287
17	Stackdriver Logging	289
17.1	Stackdriver Logging Client	289
17.2	Logger	292
17.3	Entries	295
17.4	Metrics	296

17.5	Sinks	298
17.6	Integration with Python logging module	299
17.7	Python Logging Module Handler	300
17.8	Google App Engine flexible Log Handler	302
17.9	Google Container Engine Log Handler	303
17.10	Python Logging Handler Sync Transport	303
17.11	Python Logging Handler Threaded Transport	304
17.12	Python Logging Handler Sync Transport	304
17.13	Authentication and Configuration	305
17.14	Writing log entries	305
17.15	Retrieving log entries	305
17.16	Delete all entries for a logger	306
17.17	Manage log metrics	306
17.18	Export log entries using sinks	307
17.19	Integration with Python logging module	309
18	Storage	311
18.1	Blobs / Objects	311
18.2	Buckets	321
18.3	ACL	331
18.4	Batches	335
19	Translation	339
19.1	Translation Client	339
19.2	Authentication / Configuration	341
19.3	Methods	341
20	Vision	343
20.1	Authentication and Configuration	343
20.2	Annotate an Image	344
20.3	Single-feature Shortcuts	344
20.4	No results found	344
20.5	API Reference	345
21	Google Cloud Client Library for Python	359
21.1	Getting started	359
	Python Module Index	361

1.1 Overview

Use service client objects to configure your applications.

For example:

```
>>> from google.cloud import bigquery
>>> client = bigquery.Client()
```

When creating a client in this way, the project ID will be determined by searching these locations in the following order.

- GOOGLE_CLOUD_PROJECT environment variable
- GOOGLE_APPLICATION_CREDENTIALS JSON file
- Default service configuration path from `$ gcloud beta auth application-default login`.
- Google App Engine application ID
- Google Compute Engine project ID (from metadata server)

You can override the detection of your default project by setting the `project` parameter when creating client objects.

```
>>> from google.cloud import bigquery
>>> client = bigquery.Client(project='my-project')
```

You can see what project ID a client is referencing by accessing the `project` property on the client object.

```
>>> client.project
u'my-project'
```

1.2 Authentication

The authentication credentials can be implicitly determined from the environment or directly. See [Authentication](#).

Logging in via `gcloud beta auth application-default login` will automatically configure a JSON key file with your default project ID and credentials.

Setting the `GOOGLE_APPLICATION_CREDENTIALS` and `GOOGLE_CLOUD_PROJECT` environment variables will override the automatically configured credentials.

You can change your default project ID to `my-new-default-project` by using the `gcloud` CLI tool to change the configuration.

```
$ gcloud config set project my-new-default-project
```


2.1 Overview

- **If you're running in Compute Engine or App Engine**, authentication should “just work”.
- **If you're developing locally**, the easiest way to authenticate is using the [Google Cloud SDK](#):

```
$ gcloud auth application-default login
```

Note that this command generates credentials for client libraries. To authenticate the CLI itself, use:

```
$ gcloud auth login
```

Previously, `gcloud auth login` was used for both use cases. If your `gcloud` installation does not support the new command, please update it:

```
$ gcloud components update
```

- **If you're running your application elsewhere**, you should download a [service account](#) JSON keyfile and point to it using an environment variable:

```
$ export GOOGLE_APPLICATION_CREDENTIALS="/path/to/keyfile.json"
```

2.2 Client-Provided Authentication

Every package uses a *Client* as a base for interacting with an API. For example:

```
from google.cloud import datastore
client = datastore.Client()
```

Passing no arguments at all will “just work” if you’ve followed the instructions in the [Overview](#). The credentials are inferred from your local environment by using Google [Application Default Credentials](#).

2.2.1 Credential Discovery Precedence

When loading the `Application Default Credentials`, the library will check for credentials in your environment by following the precedence outlined by `google.auth.default()`.

2.3 Explicit Credentials

The Application Default Credentials discussed above can be useful if your code needs to run in many different environments or if you just don't want authentication to be a focus in your code.

However, you may want to be explicit because

- your code will only run in one place
- you may have code which needs to be run as a specific service account every time (rather than with the locally inferred credentials)
- you may want to use two separate accounts to simultaneously access data from different projects

In these situations, you can create an explicit `Credentials` object suited to your environment. After creation, you can pass it directly to a `Client`:

```
client = Client(credentials=credentials)
```

Tip: To create a credentials object, follow the [google-auth-guide](#).

2.3.1 Google App Engine Environment

To create `credentials` just for Google App Engine:

```
from google.auth import app_engine
credentials = app_engine.Credentials()
```

2.3.2 Google Compute Engine Environment

To create `credentials` just for Google Compute Engine:

```
from google.auth import compute_engine
credentials = compute_engine.Credentials()
```

2.3.3 Service Accounts

A `service account` is stored in a JSON keyfile.

The `from_service_account_json()` factory can be used to create a `Client` with service account credentials.

For example, with a JSON keyfile:

```
client = Client.from_service_account_json('/path/to/keyfile.json')
```

Tip: Previously the Google Cloud Console would issue a PKCS12/P12 key for your service account. This library does not support that key format. You can generate a new JSON key for the same service account from the console.

2.3.4 User Accounts (3-legged OAuth 2.0) with a refresh token

The majority of cases are intended to authenticate machines or workers rather than actual user accounts. However, it's also possible to call Google Cloud APIs with a user account via [OAuth 2.0](#).

Tip: A production application should **use a service account**, but you may wish to use your own personal user account when first getting started with the `google-cloud-python` library.

The simplest way to use credentials from a user account is via Application Default Credentials using `gcloud auth login` (as mentioned above) and `google.auth.default()`:

```
import google.auth

credentials, project = google.auth.default()
```

This will still follow the [precedence](#) described above, so be sure none of the other possible environments conflict with your user provided credentials.

Advanced users of `oauth2client` can also use custom flows to create credentials using [client secrets](#) or using a [webserver flow](#). After creation, `Credentials` can be serialized with `to_json()` and stored in a file and then and deserialized with `from_json()`. In order to use `oauth2client`'s credentials with this library, you'll need to [convert them](#).

2.4 Troubleshooting

2.4.1 Setting up a Service Account

If your application is not running on Google Compute Engine, you need a [Google Developers Service Account](#).

1. Visit the [Google Developers Console](#).
2. Create a new project or click on an existing project.
3. Navigate to **APIs & auth > APIs** and enable the APIs that your application requires.

Note: You may need to enable billing in order to use these services.

- **BigQuery**
 - BigQuery API
- **Datastore**
 - Google Cloud Datastore API
- **Pub/Sub**
 - Google Cloud Pub/Sub
- **Storage**
 - Google Cloud Storage

– Google Cloud Storage JSON API

1. Navigate to **APIs & auth > Credentials**.

You should see a screen like one of the following:

Find the “Add credentials” drop down and select “Service account” to be guided through downloading a new JSON keyfile.

If you want to re-use an existing service account, you can easily generate a new keyfile. Just select the account you wish to re-use, and click **Generate new JSON key**:

2.4.2 Using Google Compute Engine

If your code is running on Google Compute Engine, using the inferred Google [Application Default Credentials](#) will be sufficient for retrieving credentials.

However, by default your credentials may not grant you access to the services you intend to use. Be sure when you [set up the GCE instance](#), you add the correct scopes for the APIs you want to access:

- **All APIs**
 - `https://www.googleapis.com/auth/cloud-platform`
 - `https://www.googleapis.com/auth/cloud-platform.read-only`
- **BigQuery**
 - `https://www.googleapis.com/auth/bigquery`
 - `https://www.googleapis.com/auth/bigquery.insertdata`
- **Datastore**
 - `https://www.googleapis.com/auth/datastore`
 - `https://www.googleapis.com/auth/userinfo.email`
- **Pub/Sub**
 - `https://www.googleapis.com/auth/pubsub`
- **Storage**
 - `https://www.googleapis.com/auth/devstorage.full_control`
 - `https://www.googleapis.com/auth/devstorage.read_only`
 - `https://www.googleapis.com/auth/devstorage.read_write`

Long-Running Operations

Wrap long-running operations returned from Google Cloud APIs.

class `google.cloud.operation.Operation` (*name*, *client*, *******caller_metadata*)

Bases: `object`

Representation of a Google API Long-Running Operation.

This wraps an operation `protobuf` object and attempts to interact with the long-running operations `service` (specific to a given API). (Some services also offer a `JSON` API that maps the same underlying data type.)

Parameters

- **name** (*str*) – The fully-qualified path naming the operation.
- **client** (*Client*) – The client used to poll for the status of the operation. If the operation was created via JSON/HTTP, the client must own a `Connection` to send polling requests. If created via `protobuf`, the client must have a `gRPC` stub in the `_operations_stub` attribute.
- **caller_metadata** (*dict*) – caller-assigned metadata about the operation

complete

Has the operation already completed?

Return type `bool`

Returns True if already completed, else false.

error = None

Error that resulted from a failed (complete) operation.

Only one of this and `response` can be populated.

classmethod `from_dict` (*operation*, *client*, *******caller_metadata*)

Factory: construct an instance from a dictionary.

Parameters

- **operation** (*dict*) – Operation as a JSON object.

- **client** (*Client*) – The client used to poll for the status of the operation.
- **caller_metadata** (*dict*) – caller-assigned metadata about the operation

Return type *Operation*

Returns new instance, with attributes based on the protobuf.

classmethod from_pb (*operation_pb*, *client*, ***caller_metadata*)

Factory: construct an instance from a protobuf.

Parameters

- **operation_pb** (*Operation*) – Protobuf to be parsed.
- **client** (object: must provide `_operations_stub` accessor.) – The client used to poll for the status of the operation.
- **caller_metadata** (*dict*) – caller-assigned metadata about the operation

Return type *Operation*

Returns new instance, with attributes based on the protobuf.

metadata = None

Metadata about the current operation (as a protobuf).

Code that uses operations must register the metadata types (via `register_type()`) to ensure that the metadata fields can be converted into the correct types.

poll()

Check if the operation has finished.

Return type *bool*

Returns A boolean indicating if the current operation has completed.

Raises **ValueError** – if the operation has already completed.

response = None

Response returned from completed operation.

Only one of this and `error` can be populated.

target = None

Instance associated with the operations – callers may set.

`google.cloud.operation.register_type(klass, type_url=None)`

Register a class as the factory for a given type URL.

Parameters

- **klass** (*type*) – class to be used as a factory for the given type
- **type_url** (*str*) – (Optional) URL naming the type. If not provided, infers the URL from the type descriptor.

Raises **ValueError** – if a registration already exists for the URL.

4.1 Base Client

Base classes for client used to interact with Google Cloud APIs.

class `google.cloud.client.Client` (*credentials=None, _http=None*)
Bases: `google.cloud.client._ClientFactoryMixin`

Client to bundle configuration needed for API requests.

Stores `credentials` and an HTTP object so that subclasses can pass them along to a connection class.

If no value is passed in for `_http`, a `requests.Session` object will be created and authorized with the `credentials`. If not, the `credentials` and `_http` need not be related.

Callers and subclasses may seek to use the private key from `credentials` to sign data.

Parameters

- **credentials** (`Credentials`) – (Optional) The OAuth2 Credentials to use for this client. If not passed (and if no `_http` object is passed), falls back to the default inferred from the environment.
- **_http** (`Session`) – (Optional) HTTP object to make requests. Can be any object that defines `request()` with the same interface as `requests.Session.request()`. If not passed, an `_http` object is created that is bound to the `credentials` for the current object. This parameter should be considered private, and could change in the future.

SCOPE = None

The scopes required for authenticating with a service.

Needs to be set by subclasses.

from_service_account_json (*json_credentials_path, *args, **kwargs*)
Factory to retrieve JSON credentials while creating client.

Parameters

- **json_credentials_path** (*str*) – The path to a private key file (this file was given to you when you created the service account). This file must contain a JSON object with a private key and other credentials information (downloaded from the Google APIs console).
- **args** (*tuple*) – Remaining positional arguments to pass to constructor.
- **kwargs** (*dict*) – Remaining keyword arguments to pass to constructor.

Return type `_ClientFactoryMixin`

Returns The client created with the retrieved JSON credentials.

Raises `TypeError` – if there is a conflict with the kwargs and the credentials created by the factory.

```
class google.cloud.client.ClientWithProject (project=None, credentials=None,
                                             _http=None)
```

Bases: `google.cloud.client.Client`, `google.cloud.client._ClientProjectMixin`

Client that also stores a project.

Parameters

- **project** (*str*) – the project which the client acts on behalf of. If not passed falls back to the default inferred from the environment.
- **credentials** (`Credentials`) – (Optional) The OAuth2 Credentials to use for this client. If not passed (and if no `_http` object is passed), falls back to the default inferred from the environment.
- **_http** (`Session`) – (Optional) HTTP object to make requests. Can be any object that defines `request()` with the same interface as `request()`. If not passed, an `_http` object is created that is bound to the `credentials` for the current object. This parameter should be considered private, and could change in the future.

Raises `ValueError` if the project is neither passed in nor set in the environment.

```
from service_account_json (json_credentials_path, *args, **kwargs)
```

Factory to retrieve JSON credentials while creating client.

Parameters

- **json_credentials_path** (*str*) – The path to a private key file (this file was given to you when you created the service account). This file must contain a JSON object with a private key and other credentials information (downloaded from the Google APIs console).
- **args** (*tuple*) – Remaining positional arguments to pass to constructor.
- **kwargs** (*dict*) – Remaining keyword arguments to pass to constructor.

Return type `_ClientFactoryMixin`

Returns The client created with the retrieved JSON credentials.

Raises `TypeError` – if there is a conflict with the kwargs and the credentials created by the factory.

4.2 Exceptions

Custom exceptions for `google.cloud` package.

`google.cloud.exceptions.GrpcRendezvous`
 Exception class raised by gRPC stable.
 alias of `_Rendezvous`

4.3 Environment Variables

Comprehensive list of environment variables used in google-cloud.

These enable many types of implicit behavior in both production and tests.

`google.cloud.environment_vars.BIGTABLE_EMULATOR = 'BIGTABLE_EMULATOR_HOST'`
 Environment variable defining host for Bigtable emulator.

`google.cloud.environment_vars.DISABLE_GRPC = 'GOOGLE_CLOUD_DISABLE_GRPC'`
 Environment variable acting as flag to disable gRPC.

To be used for APIs where both an HTTP and gRPC implementation exist.

`google.cloud.environment_vars.GCD_DATASET = 'DATASTORE_DATASET'`
 Environment variable defining default dataset ID under GCD.

`google.cloud.environment_vars.GCD_HOST = 'DATASTORE_EMULATOR_HOST'`
 Environment variable defining host for GCD dataset server.

`google.cloud.environment_vars.PUBSUB_EMULATOR = 'PUBSUB_EMULATOR_HOST'`
 Environment variable defining host for Pub/Sub emulator.

4.4 IAM Support

Non-API-specific IAM policy definitions

For allowed roles / permissions, see: <https://cloud.google.com/iam/docs/understanding-roles>

`google.cloud.iam.EDITOR_ROLE = 'roles/editor'`
 Generic role implying rights to modify an object.

`google.cloud.iam.OWNER_ROLE = 'roles/owner'`
 Generic role implying all rights to an object.

class `google.cloud.iam.Policy` (*etag=None, version=None*)
 Bases: `_abcoll.MutableMapping`

IAM Policy

See <https://cloud.google.com/iam/reference/rest/v1/Policy>

Parameters

- **etag** (*str*) – ETag used to identify a unique of the policy
- **version** (*int*) – unique version of the policy

static `all_users()`

Factory method for a member representing all users.

Return type `str`

Returns A member string representing all users.

static authenticated_users ()

Factory method for a member representing all authenticated users.

Return type `str`

Returns A member string representing all authenticated users.

static domain (domain)

Factory method for a domain member.

Parameters **domain** (`str`) – The domain for this member.

Return type `str`

Returns A member string corresponding to the given domain.

editors

Legacy access to editor role.

classmethod from_api_repr (resource)

Create a policy from the resource returned from the API.

Parameters **resource** (`dict`) – resource returned from the `getIamPolicy` API.

Return type `Policy`

Returns the parsed policy

static group (email)

Factory method for a group member.

Parameters **email** (`str`) – An id or e-mail for this particular group.

Return type `str`

Returns A member string corresponding to the given group.

owners

Legacy access to owner role.

static service_account (email)

Factory method for a service account member.

Parameters **email** (`str`) – E-mail for this particular service account.

Return type `str`

Returns A member string corresponding to the given service account.

to_api_repr ()

Construct a Policy resource.

Return type `dict`

Returns a resource to be passed to the `setIamPolicy` API.

static user (email)

Factory method for a user member.

Parameters **email** (`str`) – E-mail for this particular user.

Return type `str`

Returns A member string corresponding to the given user.

viewers

Legacy access to viewer role.

```
google.cloud.iam.VIEWER_ROLE = 'roles/viewer'
```

Generic role implying rights to access an object.

5.1 Client

Client for interacting with the Google BigQuery API.

class `google.cloud.bigquery.client.Client` (*project=None, credentials=None, _http=None*)
Bases: `google.cloud.client.ClientWithProject`

Client to bundle configuration needed for API requests.

Parameters

- **project** (*str*) – the project which the client acts on behalf of. Will be passed when creating a dataset / job. If not passed, falls back to the default inferred from the environment.
- **credentials** (*Credentials*) – (Optional) The OAuth2 Credentials to use for this client. If not passed (and if no `_http` object is passed), falls back to the default inferred from the environment.
- **_http** (*Session*) – (Optional) HTTP object to make requests. Can be any object that defines `request()` with the same interface as `requests.Session.request()`. If not passed, an `_http` object is created that is bound to the `credentials` for the current object. This parameter should be considered private, and could change in the future.

SCOPE = ('<https://www.googleapis.com/auth/bigquery>', '<https://www.googleapis.com/auth/>
The scopes required for authenticating as a BigQuery consumer.

copy_table (*job_name, destination, *sources*)
Construct a job for copying one or more tables into another table.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.copy>

Parameters

- **job_name** (*str*) – Name of the job.
- **destination** (*google.cloud.bigquery.table.Table*) – Table into which data is to be copied.

- **sources** (sequence of `google.cloud.bigquery.table.Table`) – tables to be copied.

Return type `google.cloud.bigquery.job.CopyJob`

Returns a new `CopyJob` instance

dataset (*dataset_name*, *project=None*)

Construct a dataset bound to this client.

Parameters

- **dataset_name** (*str*) – Name of the dataset.
- **project** (*str*) – (Optional) project ID for the dataset (defaults to the project of the client).

Return type `google.cloud.bigquery.dataset.Dataset`

Returns a new `Dataset` instance

extract_table_to_storage (*job_name*, *source*, **destination_uris*)

Construct a job for extracting a table into Cloud Storage files.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.extract>

Parameters

- **job_name** (*str*) – Name of the job.
- **source** (`google.cloud.bigquery.table.Table`) – table to be extracted.
- **destination_uris** (sequence of *string*) – URIs of CloudStorage file(s) into which table data is to be extracted; in format `gs://<bucket_name>/<object_name_or_glob>`.

Return type `google.cloud.bigquery.job.ExtractTableToStorageJob`

Returns a new `ExtractTableToStorageJob` instance

job_from_resource (*resource*)

Detect correct job type from resource and instantiate.

Parameters **resource** (*dict*) – one job resource from API response

Return type One of: `google.cloud.bigquery.job.LoadTableFromStorageJob`, `google.cloud.bigquery.job.CopyJob`, `google.cloud.bigquery.job.ExtractTableToStorageJob`, `google.cloud.bigquery.job.QueryJob`, `google.cloud.bigquery.job.RunSyncQueryJob`

Returns the job instance, constructed via the resource

list_datasets (*include_all=False*, *max_results=None*, *page_token=None*)

List datasets for the project associated with this client.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/datasets/list>

Parameters

- **include_all** (*bool*) – True if results include hidden datasets.
- **max_results** (*int*) – maximum number of datasets to return, If not passed, defaults to a value set by the API.
- **page_token** (*str*) – opaque marker for the next “page” of datasets. If not passed, the API will return the first page of datasets.

Return type `Iterator`

Returns Iterator of `Dataset`. accessible to the current client.

list_jobs (*max_results=None, page_token=None, all_users=None, state_filter=None*)

List jobs for the project associated with this client.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/list>

Parameters

- **max_results** (*int*) – maximum number of jobs to return, If not passed, defaults to a value set by the API.
- **page_token** (*str*) – opaque marker for the next “page” of jobs. If not passed, the API will return the first page of jobs.
- **all_users** (*bool*) – if true, include jobs owned by all users in the project.
- **state_filter** (*str*) – if passed, include only jobs matching the given state. One of
 - "done"
 - "pending"
 - "running"

Return type `Iterator`

Returns Iterable of job instances.

list_projects (*max_results=None, page_token=None*)

List projects for the project associated with this client.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/projects/list>

Parameters

- **max_results** (*int*) – maximum number of projects to return, If not passed, defaults to a value set by the API.
- **page_token** (*str*) – opaque marker for the next “page” of projects. If not passed, the API will return the first page of projects.

Return type `Iterator`

Returns Iterator of `Project` accessible to the current client.

load_table_from_storage (*job_name, destination, *source_uris*)

Construct a job for loading data into a table from CloudStorage.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.load>

Parameters

- **job_name** (*str*) – Name of the job.
- **destination** (*google.cloud.bigquery.table.Table*) – Table into which data is to be loaded.
- **source_uris** (*sequence of string*) – URIs of data files to be loaded; in format `gs://<bucket_name>/<object_name_or_glob>`.

Return type `google.cloud.bigquery.job.LoadTableFromStorageJob`

Returns a new `LoadTableFromStorageJob` instance

run_async_query (*job_name*, *query*, *udf_resources*=(), *query_parameters*=())

Construct a job for running a SQL query asynchronously.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.query>

Parameters

- **job_name** (*str*) – Name of the job.
- **query** (*str*) – SQL query to be executed
- **udf_resources** (*tuple*) – An iterable of `google.cloud.bigquery._helpers.UDFResource` (empty by default)
- **query_parameters** (*tuple*) – An iterable of `google.cloud.bigquery._helpers.AbstractQueryParameter` (empty by default)

Return type `google.cloud.bigquery.job.QueryJob`

Returns a new `QueryJob` instance

run_sync_query (*query*, *udf_resources*=(), *query_parameters*=())

Run a SQL query synchronously.

Parameters

- **query** (*str*) – SQL query to be executed
- **udf_resources** (*tuple*) – An iterable of `google.cloud.bigquery._helpers.UDFResource` (empty by default)
- **query_parameters** (*tuple*) – An iterable of `google.cloud.bigquery._helpers.AbstractQueryParameter` (empty by default)

Return type `google.cloud.bigquery.query.QueryResults`

Returns a new `QueryResults` instance

class `google.cloud.bigquery.client.Project` (*project_id*, *numeric_id*, *friendly_name*)

Bases: `object`

Wrapper for resource describing a BigQuery project.

Parameters

- **project_id** (*str*) – Opaque ID of the project
- **numeric_id** (*int*) – Numeric ID of the project
- **friendly_name** (*str*) – Display name of the project

classmethod `from_api_repr` (*resource*)

Factory: construct an instance from a resource dict.

5.2 Datasets

Define API Datasets.

class `google.cloud.bigquery.dataset.AccessGrant` (*role*, *entity_type*, *entity_id*)

Bases: `object`

Represent grant of an access role to an entity.

Every entry in the access list will have exactly one of `userByEmail`, `groupByEmail`, `domain`, `specialGroup` or `view` set. And if anything but `view` is set, it'll also have a `role` specified. `role` is omitted for a `view`, since `views` are always read-only.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/datasets>.

Parameters

- **role** (*str*) – Role granted to the entity. One of
 - 'OWNER'
 - 'WRITER'
 - 'READER'
 May also be `None` if the `entity_type` is `view`.
- **entity_type** (*str*) – Type of entity being granted the role. One of `ENTITY_TYPES`.
- **entity_id** (*str*) – ID of entity being granted the role.

Raises `ValueError` if the `entity_type` is not among `ENTITY_TYPES`, or if a `view` has `role` set or a non `view` **does not** have a `role` set.

ENTITY_TYPES = `frozenset(['specialGroup', 'groupByEmail', 'userByEmail', 'domain', 'view'])`
 Allowed entity types.

```
class google.cloud.bigquery.dataset.Dataset(name, client, access_grants=(),
                                             project=None)
```

Bases: `object`

Datasets are containers for tables.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/datasets>

Parameters

- **name** (*str*) – the name of the dataset
- **client** (`google.cloud.bigquery.client.Client`) – A client which holds credentials and project configuration for the dataset (which requires a project).
- **access_grants** (list of `AccessGrant`) – roles granted to entities for this dataset
- **project** (*str*) – (Optional) project ID for the dataset (defaults to the project of the client).

access_grants

Dataset's access grants.

Return type list of `AccessGrant`

Returns roles granted to entities for this dataset

create (*client=None*)

API call: create the dataset via a PUT request.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/tables/insert>

Parameters **client** (`Client` or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

created

Datetime at which the dataset was created.

Return type `datetime.datetime`, or `NoneType`

Returns the creation time (`None` until set from the server).

dataset_id

ID for the dataset resource.

Return type str, or NoneType

Returns the ID (None until set from the server).

default_table_expiration_ms

Default expiration time for tables in the dataset.

Return type int, or NoneType

Returns The time in milliseconds, or None (the default).

delete (*client=None*)

API call: delete the dataset via a DELETE request.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/datasets/delete>

Parameters **client** (*Client* or NoneType) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

description

Description of the dataset.

Return type str, or NoneType

Returns The description as set by the user, or None (the default).

etag

ETag for the dataset resource.

Return type str, or NoneType

Returns the ETag (None until set from the server).

exists (*client=None*)

API call: test for the existence of the dataset via a GET request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/datasets/get>

Parameters **client** (*Client* or NoneType) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

Return type bool

Returns Boolean indicating existence of the dataset.

friendly_name

Title of the dataset.

Return type str, or NoneType

Returns The name as set by the user, or None (the default).

classmethod from_api_repr (*resource, client*)

Factory: construct a dataset given its API representation

Parameters

- **resource** (*dict*) – dataset resource representation returned from the API
- **client** (*google.cloud.bigquery.client.Client*) – Client which holds credentials and project configuration for the dataset.

Return type *google.cloud.bigquery.dataset.Dataset*

Returns Dataset parsed from `resource`.

list_tables (*max_results=None, page_token=None*)

List tables for the project associated with this client.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/tables/list>

Parameters

- **max_results** (*int*) – (Optional) Maximum number of tables to return. If not passed, defaults to a value set by the API.
- **page_token** (*str*) – (Optional) Opaque marker for the next “page” of datasets. If not passed, the API will return the first page of datasets.

Return type `Iterator`

Returns Iterator of `Table` contained within the current dataset.

location

Location in which the dataset is hosted.

Return type `str`, or `NoneType`

Returns The location as set by the user, or `None` (the default).

modified

Datetime at which the dataset was last modified.

Return type `datetime.datetime`, or `NoneType`

Returns the modification time (`None` until set from the server).

patch (*client=None, **kw*)

API call: update individual dataset properties via a PATCH request.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/datasets/patch>

Parameters

- **client** (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current dataset.
- **kw** (*dict*) – properties to be patched.

Raises `ValueError` for invalid value types.

path

URL path for the dataset’s APIs.

Return type `str`

Returns the path based on project and dataset name.

project

Project bound to the dataset.

Return type `str`

Returns the project (derived from the client).

reload (*client=None*)

API call: refresh dataset properties via a GET request.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/datasets/get>

Parameters **client** (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

self_link

URL for the dataset resource.

Return type `str`, or `NoneType`

Returns the URL (None until set from the server).

table (*name*, *schema*=())

Construct a table bound to this dataset.

Parameters

- **name** (*str*) – Name of the table.
- **schema** (list of `google.cloud.bigquery.table.SchemaField`) – The table's schema

Return type `google.cloud.bigquery.table.Table`

Returns a new `Table` instance

update (*client*=None)

API call: update dataset properties via a PUT request.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/datasets/update>

Parameters **client** (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

5.3 Jobs

Define API Jobs.

class `google.cloud.bigquery.job.AutoDetectSchema` (*name*, *property_type*)

Bases: `google.cloud.bigquery._helpers._TypedProperty`

Typed Property for autodetect properties.

Raises **ValueError** – on set operation if `instance.schema` is already defined.

class `google.cloud.bigquery.job.Compression` (*name*)

Bases: `google.cloud.bigquery._helpers._EnumProperty`

Pseudo-enum for compression properties.

class `google.cloud.bigquery.job.CopyJob` (*name*, *destination*, *sources*, *client*)

Bases: `google.cloud.bigquery.job._AsyncJob`

Asynchronous job: copy data into a table from other tables.

Parameters

- **name** (*str*) – the name of the job
- **destination** (`google.cloud.bigquery.table.Table`) – Table into which data is to be loaded.
- **sources** (list of `google.cloud.bigquery.table.Table`) – Table into which data is to be loaded.
- **client** (`google.cloud.bigquery.client.Client`) – A client which holds credentials and project configuration for the dataset (which requires a project).

add_done_callback (*fn*)

Add a callback to be executed when the operation is complete.

If the operation is not already complete, this will start a helper thread to poll for the status of the operation in the background.

Parameters *fn* (*Callable*[*Future*]) – The callback to execute when the operation is complete.

begin (*client=None*)

API call: begin the job via a POST request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/insert>

Parameters *client* (*Client* or *NoneType*) – the client to use. If not passed, falls back to the *client* stored on the current dataset.

Raises *ValueError* if the job has already begin.

cancel (*client=None*)

API call: cancel job via a POST request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/cancel>

Parameters *client* (*Client* or *NoneType*) – the client to use. If not passed, falls back to the *client* stored on the current dataset.

Return type *bool*

Returns Boolean indicating that the cancel request was sent.

cancelled ()

Check if the job has been cancelled.

This always returns False. It's not possible to check if a job was cancelled in the API. This method is here to satisfy the interface for `google.api.core.future.Future`.

Return type *bool*

Returns False

create_disposition

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.copy.createDisposition>

created

Datetime at which the job was created.

Return type *datetime.datetime*, or *NoneType*

Returns the creation time (None until set from the server).

done ()

Refresh the job and checks if it is complete.

Return type *bool*

Returns True if the job is complete, False otherwise.

ended

Datetime at which the job finished.

Return type *datetime.datetime*, or *NoneType*

Returns the end time (None until set from the server).

error_result

Error information about the job as a whole.

Return type mapping, or `NoneType`

Returns the error information (None until set from the server).

errors

Information about individual errors generated by the job.

Return type list of mappings, or `NoneType`

Returns the error information (None until set from the server).

etag

ETag for the job resource.

Return type str, or `NoneType`

Returns the ETag (None until set from the server).

exception (*timeout=None*)

Get the exception from the operation, blocking if necessary.

Parameters **timeout** (*int*) – How long to wait for the operation to complete. If None, wait indefinitely.

Returns The operation's error.

Return type Optional[google.gax.GaxError]

exists (*client=None*)

API call: test for the existence of the job via a GET request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/get>

Parameters **client** (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

Return type bool

Returns Boolean indicating existence of the job.

classmethod from_api_repr (*resource, client*)

Factory: construct a job given its API representation

Parameters

- **resource** (*dict*) – dataset job representation returned from the API
- **client** (*google.cloud.bigquery.client.Client*) – Client which holds credentials and project configuration for the dataset.

Return type *google.cloud.bigquery.job.CopyJob*

Returns Job parsed from `resource`.

job_type

Type of job

Return type str

Returns one of 'load', 'copy', 'extract', 'query'

path

URL path for the job's APIs.

Return type str

Returns the path based on project and job name.

project

Project bound to the job.

Return type `str`

Returns the project (derived from the client).

reload (*client=None*)

API call: refresh job properties via a GET request.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/get>

Parameters **client** (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

result (*timeout=None*)

Start the job and wait for it to complete and get the result.

Parameters **timeout** (*int*) – How long to wait for job to complete before raising a `TimeoutError`.

Return type `_AsyncJob`

Returns This instance.

Raises `GoogleCloudError` if the job failed or `TimeoutError` if the job did not complete in the given timeout.

running ()

True if the operation is currently running.

self_link

URL for the job resource.

Return type `str`, or `NoneType`

Returns the URL (None until set from the server).

set_exception (*exception*)

Set the Future's exception.

set_result (*result*)

Set the Future's result.

started

Datetime at which the job was started.

Return type `datetime.datetime`, or `NoneType`

Returns the start time (None until set from the server).

state

Status of the job.

Return type `str`, or `NoneType`

Returns the state (None until set from the server).

user_email

E-mail address of user who submitted the job.

Return type `str`, or `NoneType`

Returns the URL (None until set from the server).

write_disposition

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.copy.writeDisposition>

class google.cloud.bigquery.job.**CreateDisposition** (*name*)

Bases: google.cloud.bigquery._helpers._EnumProperty

Pseudo-enum for create_disposition properties.

class google.cloud.bigquery.job.**DestinationFormat** (*name*)

Bases: google.cloud.bigquery._helpers._EnumProperty

Pseudo-enum for destination_format properties.

class google.cloud.bigquery.job.**Encoding** (*name*)

Bases: google.cloud.bigquery._helpers._EnumProperty

Pseudo-enum for encoding properties.

class google.cloud.bigquery.job.**ExtractTableToStorageJob** (*name*, *source*, *destination_uris*, *client*)

Bases: google.cloud.bigquery.job._AsyncJob

Asynchronous job: extract data from a table into Cloud Storage.

Parameters

- **name** (*str*) – the name of the job
- **source** (*google.cloud.bigquery.table.Table*) – Table into which data is to be loaded.
- **destination_uris** (*list of string*) – URIs describing Cloud Storage blobs into which extracted data will be written, in format `gs://<bucket_name>/<object_name_or_glob>`.
- **client** (*google.cloud.bigquery.client.Client*) – A client which holds credentials and project configuration for the dataset (which requires a project).

add_done_callback (*fn*)

Add a callback to be executed when the operation is complete.

If the operation is not already complete, this will start a helper thread to poll for the status of the operation in the background.

Parameters **fn** (*Callable[Future]*) – The callback to execute when the operation is complete.

begin (*client=None*)

API call: begin the job via a POST request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/insert>

Parameters **client** (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

Raises *ValueError* if the job has already begin.

cancel (*client=None*)

API call: cancel job via a POST request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/cancel>

Parameters **client** (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

Return type *bool*

Returns Boolean indicating that the cancel request was sent.

cancelled()

Check if the job has been cancelled.

This always returns False. It's not possible to check if a job was cancelled in the API. This method is here to satisfy the interface for `google.api.core.future.Future`.

Return type `bool`

Returns False

compression

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.extract.compression>

created

Datetime at which the job was created.

Return type `datetime.datetime`, or `NoneType`

Returns the creation time (None until set from the server).

destination_format

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.extract.destinationFormat>

done()

Refresh the job and checks if it is complete.

Return type `bool`

Returns True if the job is complete, False otherwise.

ended

Datetime at which the job finished.

Return type `datetime.datetime`, or `NoneType`

Returns the end time (None until set from the server).

error_result

Error information about the job as a whole.

Return type mapping, or `NoneType`

Returns the error information (None until set from the server).

errors

Information about individual errors generated by the job.

Return type list of mappings, or `NoneType`

Returns the error information (None until set from the server).

etag

ETag for the job resource.

Return type `str`, or `NoneType`

Returns the ETag (None until set from the server).

exception (timeout=None)

Get the exception from the operation, blocking if necessary.

Parameters `timeout (int)` – How long to wait for the operation to complete. If None, wait indefinitely.

Returns The operation's error.

Return type Optional[google.gax.GaxError]

exists (*client=None*)

API call: test for the existence of the job via a GET request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/get>

Parameters **client** (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

Return type *bool*

Returns Boolean indicating existence of the job.

field_delimiter

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.extract.fieldDelimiter>

classmethod from_api_repr (*resource, client*)

Factory: construct a job given its API representation

Parameters

- **resource** (*dict*) – dataset job representation returned from the API
- **client** (*google.cloud.bigquery.client.Client*) – Client which holds credentials and project configuration for the dataset.

Return type *google.cloud.bigquery.job.ExtractTableToStorageJob*

Returns Job parsed from *resource*.

job_type

Type of job

Return type *str*

Returns one of 'load', 'copy', 'extract', 'query'

path

URL path for the job's APIs.

Return type *str*

Returns the path based on project and job name.

print_header

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.extract.printHeader>

project

Project bound to the job.

Return type *str*

Returns the project (derived from the client).

reload (*client=None*)

API call: refresh job properties via a GET request.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/get>

Parameters **client** (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

result (*timeout=None*)

Start the job and wait for it to complete and get the result.

Parameters `timeout` (*int*) – How long to wait for job to complete before raising a `TimeoutError`.

Return type `_AsyncJob`

Returns This instance.

Raises `GoogleCloudError` if the job failed or `TimeoutError` if the job did not complete in the given timeout.

running()

True if the operation is currently running.

self_link

URL for the job resource.

Return type `str`, or `NoneType`

Returns the URL (None until set from the server).

set_exception(exception)

Set the Future's exception.

set_result(result)

Set the Future's result.

started

Datetime at which the job was started.

Return type `datetime.datetime`, or `NoneType`

Returns the start time (None until set from the server).

state

Status of the job.

Return type `str`, or `NoneType`

Returns the state (None until set from the server).

user_email

E-mail address of user who submitted the job.

Return type `str`, or `NoneType`

Returns the URL (None until set from the server).

```
class google.cloud.bigquery.job.LoadTableFromStorageJob(name, destination,
                                                         source_uris, client,
                                                         schema=())
```

Bases: `google.cloud.bigquery.job._AsyncJob`

Asynchronous job for loading data into a table from CloudStorage.

Parameters

- **name** (*str*) – the name of the job
- **destination** (*google.cloud.bigquery.table.Table*) – Table into which data is to be loaded.
- **source_uris** (*sequence of string*) – URIs of one or more data files to be loaded, in format `gs://<bucket_name>/<object_name_or_glob>`.
- **client** (*google.cloud.bigquery.client.Client*) – A client which holds credentials and project configuration for the dataset (which requires a project).

- **schema** (list of `google.cloud.bigquery.table.SchemaField`) – The job’s schema

add_done_callback (*fn*)

Add a callback to be executed when the operation is complete.

If the operation is not already complete, this will start a helper thread to poll for the status of the operation in the background.

Parameters **fn** (*Callable[Future]*) – The callback to execute when the operation is complete.

allow_jagged_rows

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.load.allowJaggedRows>

allow_quoted_newlines

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.load.allowQuotedNewlines>

autodetect

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.load.autodetect>

begin (*client=None*)

API call: begin the job via a POST request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/insert>

Parameters **client** (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

Raises *ValueError* if the job has already begin.

cancel (*client=None*)

API call: cancel job via a POST request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/cancel>

Parameters **client** (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

Return type *bool*

Returns Boolean indicating that the cancel request was sent.

cancelled ()

Check if the job has been cancelled.

This always returns False. It’s not possible to check if a job was cancelled in the API. This method is here to satisfy the interface for `google.api.core.future.Future`.

Return type *bool*

Returns False

create_disposition

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.load.createDisposition>

created

Datetime at which the job was created.

Return type *datetime.datetime*, or *NoneType*

Returns the creation time (None until set from the server).

done ()

Refresh the job and checks if it is complete.

Return type `bool`

Returns True if the job is complete, False otherwise.

encoding

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.load.encoding>

ended

Datetime at which the job finished.

Return type `datetime.datetime`, or `NoneType`

Returns the end time (None until set from the server).

error_result

Error information about the job as a whole.

Return type `mapping`, or `NoneType`

Returns the error information (None until set from the server).

errors

Information about individual errors generated by the job.

Return type `list of mappings`, or `NoneType`

Returns the error information (None until set from the server).

etag

ETag for the job resource.

Return type `str`, or `NoneType`

Returns the ETag (None until set from the server).

exception (*timeout=None*)

Get the exception from the operation, blocking if necessary.

Parameters `timeout` (`int`) – How long to wait for the operation to complete. If None, wait indefinitely.

Returns The operation's error.

Return type `Optional[google.gax.GaxError]`

exists (*client=None*)

API call: test for the existence of the job via a GET request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/get>

Parameters `client` (`Client` or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

Return type `bool`

Returns Boolean indicating existence of the job.

field_delimiter

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.load.fieldDelimiter>

classmethod from_api_repr (*resource, client*)

Factory: construct a job given its API representation

Parameters

- **resource** (`dict`) – dataset job representation returned from the API

- **client** (*google.cloud.bigquery.client.Client*) – Client which holds credentials and project configuration for the dataset.

Return type *google.cloud.bigquery.job.LoadTableFromStorageJob*

Returns Job parsed from *resource*.

ignore_unknown_values

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.load.ignoreUnknownValues>

input_file_bytes

Count of bytes loaded from source files.

Return type *int*, or *NoneType*

Returns the count (None until set from the server).

input_files

Count of source files.

Return type *int*, or *NoneType*

Returns the count (None until set from the server).

job_type

Type of job

Return type *str*

Returns one of 'load', 'copy', 'extract', 'query'

max_bad_records

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.load.maxBadRecords>

null_marker

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.load.nullMarker>

output_bytes

Count of bytes saved to destination table.

Return type *int*, or *NoneType*

Returns the count (None until set from the server).

output_rows

Count of rows saved to destination table.

Return type *int*, or *NoneType*

Returns the count (None until set from the server).

path

URL path for the job's APIs.

Return type *str*

Returns the path based on project and job name.

project

Project bound to the job.

Return type *str*

Returns the project (derived from the client).

quote_character

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.load.quote>

reload (*client=None*)

API call: refresh job properties via a GET request.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/get>

Parameters **client** (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

result (*timeout=None*)

Start the job and wait for it to complete and get the result.

Parameters **timeout** (*int*) – How long to wait for job to complete before raising a *TimeoutError*.

Return type *_AsyncJob*

Returns This instance.

Raises *GoogleCloudError* if the job failed or *TimeoutError* if the job did not complete in the given timeout.

running ()

True if the operation is currently running.

schema

Table's schema.

Return type list of *SchemaField*

Returns fields describing the schema

self_link

URL for the job resource.

Return type str, or *NoneType*

Returns the URL (None until set from the server).

set_exception (*exception*)

Set the Future's exception.

set_result (*result*)

Set the Future's result.

skip_leading_rows

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.load.skipLeadingRows>

source_format

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.load.sourceFormat>

started

Datetime at which the job was started.

Return type *datetime.datetime*, or *NoneType*

Returns the start time (None until set from the server).

state

Status of the job.

Return type str, or *NoneType*

Returns the state (None until set from the server).

user_email

E-mail address of user who submitted the job.

Return type `str`, or `NoneType`

Returns the URL (None until set from the server).

write_disposition

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.load.writeDisposition>

```
class google.cloud.bigquery.job.QueryJob(name, query, client, udf_resources=(),
                                         query_parameters=())
```

Bases: `google.cloud.bigquery.job._AsyncJob`

Asynchronous job: query tables.

Parameters

- **name** (*str*) – the name of the job
- **query** (*str*) – SQL query string
- **client** (*google.cloud.bigquery.client.Client*) – A client which holds credentials and project configuration for the dataset (which requires a project).
- **udf_resources** (*tuple*) – An iterable of `google.cloud.bigquery._helpers.UDFResource` (empty by default)
- **query_parameters** (*tuple*) – An iterable of `google.cloud.bigquery._helpers.AbstractQueryParameter` (empty by default)

add_done_callback (*fn*)

Add a callback to be executed when the operation is complete.

If the operation is not already complete, this will start a helper thread to poll for the status of the operation in the background.

Parameters **fn** (*Callable[Future]*) – The callback to execute when the operation is complete.

allow_large_results

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.query.allowLargeResults>

begin (*client=None*)

API call: begin the job via a POST request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/insert>

Parameters **client** (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

Raises `ValueError` if the job has already begin.

cancel (*client=None*)

API call: cancel job via a POST request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/cancel>

Parameters **client** (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

Return type `bool`

Returns Boolean indicating that the cancel request was sent.

cancelled()

Check if the job has been cancelled.

This always returns False. It's not possible to check if a job was cancelled in the API. This method is here to satisfy the interface for `google.api.core.future.Future`.

Return type `bool`

Returns False

create_disposition

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.query.createDisposition>

created

Datetime at which the job was created.

Return type `datetime.datetime`, or `NoneType`

Returns the creation time (None until set from the server).

default_dataset

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.query.defaultDataset>

destination

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.query.destinationTable>

done()

Refresh the job and checks if it is complete.

Return type `bool`

Returns True if the job is complete, False otherwise.

dry_run

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.dryRun>

ended

Datetime at which the job finished.

Return type `datetime.datetime`, or `NoneType`

Returns the end time (None until set from the server).

error_result

Error information about the job as a whole.

Return type `mapping`, or `NoneType`

Returns the error information (None until set from the server).

errors

Information about individual errors generated by the job.

Return type list of mappings, or `NoneType`

Returns the error information (None until set from the server).

etag

ETag for the job resource.

Return type `str`, or `NoneType`

Returns the ETag (None until set from the server).

exception (timeout=None)

Get the exception from the operation, blocking if necessary.

Parameters `timeout` (*int*) – How long to wait for the operation to complete. If None, wait indefinitely.

Returns The operation's error.

Return type Optional[google.gax.GaxError]

exists (*client=None*)

API call: test for the existence of the job via a GET request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/get>

Parameters `client` (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

Return type *bool*

Returns Boolean indicating existence of the job.

flatten_results

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.query.flattenResults>

classmethod from_api_repr (*resource, client*)

Factory: construct a job given its API representation

Parameters

- **resource** (*dict*) – dataset job representation returned from the API
- **client** (*google.cloud.bigquery.client.Client*) – Client which holds credentials and project configuration for the dataset.

Return type *google.cloud.bigquery.job.RunAsyncQueryJob*

Returns Job parsed from *resource*.

job_type

Type of job

Return type *str*

Returns one of 'load', 'copy', 'extract', 'query'

maximum_billing_tier

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.query.maximumBillingTier>

maximum_bytes_billed

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.query.maximumBytesBilled>

path

URL path for the job's APIs.

Return type *str*

Returns the path based on project and job name.

priority

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.query.priority>

project

Project bound to the job.

Return type *str*

Returns the project (derived from the client).

query_results()

Construct a QueryResults instance, bound to this job.

Return type `QueryResults`

Returns results instance

reload(client=None)

API call: refresh job properties via a GET request.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/get>

Parameters **client** (`Client` or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

result(timeout=None)

Start the job and wait for it to complete and get the result.

Parameters **timeout** (`int`) – How long to wait for job to complete before raising a `TimeoutError`.

Return type `Iterator`

Returns Iterator of row data tuple`s. During each page, the iterator will have the ``total_rows` attribute set, which counts the total number of rows **in the result set** (this is distinct from the total number of rows in the current page: `iterator.page.num_items`).

Raises `GoogleCloudError` if the job failed or `TimeoutError` if the job did not complete in the given timeout.

running()

True if the operation is currently running.

self_link

URL for the job resource.

Return type `str`, or `NoneType`

Returns the URL (None until set from the server).

set_exception(exception)

Set the Future's exception.

set_result(result)

Set the Future's result.

started

Datetime at which the job was started.

Return type `datetime.datetime`, or `NoneType`

Returns the start time (None until set from the server).

state

Status of the job.

Return type `str`, or `NoneType`

Returns the state (None until set from the server).

use_legacy_sql

See <https://cloud.google.com/bigquery/docs/reference/v2/jobs#configuration.query.useLegacySql>

use_query_cache

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.query.useQueryCache>

user_email

E-mail address of user who submitted the job.

Return type str, or NoneType

Returns the URL (None until set from the server).

write_disposition

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs#configuration.query.writeDisposition>

class google.cloud.bigquery.job.**QueryPriority** (*name*)

Bases: google.cloud.bigquery._helpers._EnumProperty

Pseudo-enum for QueryJob.priority property.

class google.cloud.bigquery.job.**SourceFormat** (*name*)

Bases: google.cloud.bigquery._helpers._EnumProperty

Pseudo-enum for source_format properties.

class google.cloud.bigquery.job.**WriteDisposition** (*name*)

Bases: google.cloud.bigquery._helpers._EnumProperty

Pseudo-enum for write_disposition properties.

5.4 Query

Define API Queries.

class google.cloud.bigquery.query.**QueryResults** (*query*, *client*, *udf_resources=()*,
query_parameters=())

Bases: `object`

Synchronous job: query tables.

Parameters

- **query** (*str*) – SQL query string
- **client** (*google.cloud.bigquery.client.Client*) – A client which holds credentials and project configuration for the dataset (which requires a project).
- **udf_resources** (*tuple*) – An iterable of google.cloud.bigquery.job.UDFResource (empty by default)
- **query_parameters** (*tuple*) – An iterable of google.cloud.bigquery._helpers.AbstractQueryParameter (empty by default)

cache_hit

Query results served from cache.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/query#cacheHit>

Return type bool or NoneType

Returns True if the query results were served from cache (None until set by the server).

complete

Server completed query.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/query#jobComplete>

Return type bool or NoneType

Returns True if the query completed on the server (None until set by the server).

default_dataset

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/query#defaultDataset>

dry_run

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/query#dryRun>

errors

Errors generated by the query.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/query#errors>

Return type list of mapping, or NoneType

Returns Mappings describing errors generated on the server (None until set by the server).

fetch_data (*max_results=None*, *page_token=None*, *start_index=None*, *timeout_ms=None*,
client=None)

API call: fetch a page of query result data via a GET request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/getQueryResults>

Parameters

- **max_results** (*int*) – (Optional) maximum number of rows to return.
- **page_token** (*str*) – (Optional) token representing a cursor into the table's rows.
- **start_index** (*int*) – (Optional) zero-based index of starting row
- **timeout_ms** (*int*) – (Optional) How long to wait for the query to complete, in milliseconds, before the request times out and returns. Note that this is only a timeout for the request, not the query. If the query takes longer to run than the timeout value, the call returns without any results and with the 'jobComplete' flag set to false. You can call `GetQueryResults()` to wait for the query to complete and read the results. The default value is 10000 milliseconds (10 seconds).
- **client** (*Client* or NoneType) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

Return type Iterator

Returns Iterator of row data tuple's. During each page, the iterator will have the `total_rows` attribute set, which counts the total number of rows in the result set (this is distinct from the total number of rows in the current page: `iterator.page.num_items`).

Raises ValueError if the query has not yet been executed.

classmethod **from_query_job** (*job*)

Factory: construct from an existing job.

Parameters **job** (*QueryJob*) – existing job

Return type *QueryResults*

Returns the instance, bound to the job

job

Job instance used to run the query.

Return type *google.cloud.bigquery.job.QueryJob*, or NoneType

Returns Job instance used to run the query (None until `jobReference` property is set by the server).

max_results

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/query#maxResults>

name

Job name, generated by the back-end.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/query#jobReference>

Return type list of mapping, or `NoneType`

Returns Mappings describing errors generated on the server (None until set by the server).

num_dml_affected_rows

Total number of rows affected by a DML query.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/query#numDmlAffectedRows>

Return type int, or `NoneType`

Returns Count generated on the server (None until set by the server).

page_token

Token for fetching next batch of results.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/query#pageToken>

Return type str, or `NoneType`

Returns Token generated on the server (None until set by the server).

preserve_nulls

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/query#preserveNulls>

project

Project bound to the job.

Return type str

Returns the project (derived from the client).

rows

Query results.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/query#rows>

Return type list of tuples of row values, or `NoneType`

Returns fields describing the schema (None until set by the server).

run (*client=None*)

API call: run the query via a POST request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/query>

Parameters **client** (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

schema

Schema for query results.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/query#schema>

Return type list of `SchemaField`, or `NoneType`

Returns fields describing the schema (None until set by the server).

timeout_ms

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/query#timeoutMs>

total_bytes_processed

Total number of bytes processed by the query.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/query#totalBytesProcessed>

Return type `int`, or `NoneType`

Returns Count generated on the server (None until set by the server).

total_rows

Total number of rows returned by the query.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/query#totalRows>

Return type `int`, or `NoneType`

Returns Count generated on the server (None until set by the server).

use_legacy_sql

See <https://cloud.google.com/bigquery/docs/reference/v2/jobs/query#useLegacySql>

use_query_cache

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs/query#useQueryCache>

5.5 Schemas

Schemas for BigQuery tables / queries.

```
class google.cloud.bigquery.schema.SchemaField(name, field_type, mode='NULLABLE',
                                                description=None, fields=())
```

Bases: `object`

Describe a single field within a table schema.

Parameters

- **name** (*str*) – the name of the field.
- **field_type** (*str*) – the type of the field (one of 'STRING', 'INTEGER', 'FLOAT', 'BOOLEAN', 'TIMESTAMP' or 'RECORD').
- **mode** (*str*) – the mode of the field (one of 'NULLABLE', 'REQUIRED', or 'REPEATED').
- **description** (*str*) – optional description for the field.
- **fields** (tuple of *SchemaField*) – subfields (requires `field_type` of 'RECORD').

description

Optional[str] – Description for the field.

field_type

str – The type of the field.

Will be one of 'STRING', 'INTEGER', 'FLOAT', 'BOOLEAN', 'TIMESTAMP' or 'RECORD'.

fields

tuple – Subfields contained in this field.

If `field_type` is not 'RECORD', this property must be empty / unset.

classmethod from_api_repr (*api_repr*)

Return a `SchemaField` object deserialized from a dictionary.

Parameters `api_repr` (*Mapping[str, str]*) – The serialized representation of the SchemaField, such as what is output by `to_api_repr()`.

Returns The SchemaField object.

Return type *SchemaField*

is_nullable

Check whether 'mode' is 'nullable'.

mode

str – The mode of the field.

Will be one of 'NULLABLE', 'REQUIRED', or 'REPEATED'.

name

str – The name of the field.

to_api_repr()

Return a dictionary representing this schema field.

Returns

A dictionary representing the SchemaField in a serialized form.

Return type *dict*

5.6 Tables

Define API Datasets.

class `google.cloud.bigquery.table.Table` (*name, dataset, schema=()*)

Bases: *object*

Tables represent a set of rows whose values correspond to a schema.

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/tables>

Parameters

- **name** (*str*) – the name of the table
- **dataset** (*google.cloud.bigquery.dataset.Dataset*) – The dataset which contains the table.
- **schema** (list of *SchemaField*) – The table's schema

create (*client=None*)

API call: create the table via a PUT request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/tables/insert>

Parameters `client` (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

created

Datetime at which the table was created.

Return type *datetime.datetime*, or *NoneType*

Returns the creation time (None until set from the server).

dataset_name

Name of dataset containing the table.

Return type `str`

Returns the ID (derived from the dataset).

delete (*client=None*)

API call: delete the table via a DELETE request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/tables/delete>

Parameters **client** (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

description

Description of the table.

Return type `str`, or `NoneType`

Returns The description as set by the user, or `None` (the default).

etag

ETag for the table resource.

Return type `str`, or `NoneType`

Returns the ETag (`None` until set from the server).

exists (*client=None*)

API call: test for the existence of the table via a GET request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/tables/get>

Parameters **client** (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

Return type `bool`

Returns Boolean indicating existence of the table.

expires

Datetime at which the table will be removed.

Return type `datetime.datetime`, or `NoneType`

Returns the expiration time, or `None`

fetch_data (*max_results=None, page_token=None, client=None*)

API call: fetch the table data via a GET request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/tabledata/list>

Note: This method assumes that its instance's `schema` attribute is up-to-date with the schema as defined on the back-end: if the two schemas are not identical, the values returned may be incomplete. To ensure that the local copy of the schema is up-to-date, call `reload()`.

Parameters

- **max_results** (*int*) – (Optional) Maximum number of rows to return.
- **page_token** (*str*) – (Optional) Token representing a cursor into the table's rows.
- **client** (*Client*) – (Optional) The client to use. If not passed, falls back to the `client` stored on the current dataset.

Return type `Iterator`

Returns Iterator of row data tuple`s. During each page, the iterator will have the ``total_rows` attribute set, which counts the total number of rows **in the table** (this is distinct from the total number of rows in the current page: `iterator.page.num_items`).

friendly_name

Title of the table.

Return type str, or NoneType

Returns The name as set by the user, or None (the default).

classmethod from_api_repr (*resource*, *dataset*)

Factory: construct a table given its API representation

Parameters

- **resource** (*dict*) – table resource representation returned from the API
- **dataset** (*google.cloud.bigquery.dataset.Dataset*) – The dataset containing the table.

Return type *google.cloud.bigquery.table.Table*

Returns Table parsed from *resource*.

insert_data (*rows*, *row_ids=None*, *skip_invalid_rows=None*, *ignore_unknown_values=None*, *template_suffix=None*, *client=None*)

API call: insert table data via a POST request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/tabledata/insertAll>

Parameters

- **rows** (*list of tuples*) – Row data to be inserted. Each tuple should contain data for each schema field on the current table and in the same order as the schema fields.
- **row_ids** (*list of string*) – Unique ids, one per row being inserted. If not passed, no de-duplication occurs.
- **skip_invalid_rows** (*bool*) – (Optional) Insert all valid rows of a request, even if invalid rows exist. The default value is False, which causes the entire request to fail if any invalid rows exist.
- **ignore_unknown_values** (*bool*) – (Optional) Accept rows that contain values that do not match the schema. The unknown values are ignored. Default is False, which treats unknown values as errors.
- **template_suffix** (*str*) – (Optional) treat name as a template table and provide a suffix. BigQuery will create the table <name> + <template_suffix> based on the schema of the template table. See <https://cloud.google.com/bigquery/streaming-data-into-bigquery#template-tables>
- **client** (*Client* or NoneType) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

Return type list of mappings

Returns One mapping per row with insert errors: the “index” key identifies the row, and the “errors” key contains a list of the mappings describing one or more problems with the row.

Raises ValueError if table’s schema is not set

list_partitions (*client=None*)

List the partitions in a table.

Parameters `client` (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

Return type `list`

Returns a list of time partitions

location

Location in which the table is hosted.

Return type `str`, or `NoneType`

Returns The location as set by the user, or `None` (the default).

modified

Datetime at which the table was last modified.

Return type `datetime.datetime`, or `NoneType`

Returns the modification time (`None` until set from the server).

num_bytes

The size of the table in bytes.

Return type `int`, or `NoneType`

Returns the byte count (`None` until set from the server).

num_rows

The number of rows in the table.

Return type `int`, or `NoneType`

Returns the row count (`None` until set from the server).

partition_expiration

Expiration time in ms for a partition :rtype: `int`, or `NoneType` :returns: Returns the time in ms for partition expiration

partitioning_type

Time partitioning of the table. :rtype: `str`, or `NoneType` :returns: Returns type if the table is partitioned, `None` otherwise.

patch (*client=None*, *friendly_name=<object object>*, *description=<object object>*, *location=<object object>*, *expires=<object object>*, *view_query=<object object>*, *schema=<object object>*)
API call: update individual table properties via a PATCH request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/tables/patch>

Parameters

- **client** (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current dataset.
- **friendly_name** (*str*) – (Optional) a descriptive name for this table.
- **description** (*str*) – (Optional) a description of this table.
- **location** (*str*) – (Optional) the geographic location where the table resides.
- **expires** (*datetime.datetime*) – (Optional) point in time at which the table expires.
- **view_query** (*str*) – SQL query defining the table as a view
- **schema** (list of `SchemaField`) – fields describing the schema

Raises `ValueError` for invalid value types.

path

URL path for the table's APIs.

Return type `str`

Returns the path based on project and dataset name.

project

Project bound to the table.

Return type `str`

Returns the project (derived from the dataset).

reload (*client=None*)

API call: refresh table properties via a GET request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/tables/get>

Parameters **client** (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

row_from_mapping (*mapping*)

Convert a mapping to a row tuple using the schema.

Parameters **mapping** (*dict*) – Mapping of row data: must contain keys for all required fields in the schema. Keys which do not correspond to a field in the schema are ignored.

Return type `tuple`

Returns Tuple whose elements are ordered according to the table's schema.

Raises `ValueError` if table's schema is not set

schema

Table's schema.

Return type list of `SchemaField`

Returns fields describing the schema

self_link

URL for the table resource.

Return type `str`, or `NoneType`

Returns the URL (None until set from the server).

table_id

ID for the table resource.

Return type `str`, or `NoneType`

Returns the ID (None until set from the server).

table_type

The type of the table.

Possible values are "TABLE", "VIEW", or "EXTERNAL".

Return type `str`, or `NoneType`

Returns the URL (None until set from the server).

update (*client=None*)

API call: update table properties via a PUT request

See <https://cloud.google.com/bigquery/docs/reference/rest/v2/tables/update>

Parameters **client** (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current dataset.

```
upload_from_file (file_obj, source_format, rewind=False, size=None, num_retries=6,  
                  allow_jagged_rows=None, allow_quoted_newlines=None, create_disposition=None,  
                  encoding=None, field_delimiter=None, ignore_unknown_values=None,  
                  max_bad_records=None, quote_character=None, skip_leading_rows=None,  
                  write_disposition=None, client=None, job_name=None, null_marker=None)
```

Upload the contents of this table from a file-like object.

Parameters

- **file_obj** (*file*) – A file handle opened in binary mode for reading.
- **source_format** (*str*) – Any supported format. The full list of supported formats is documented under the `configuration.extract.destinationFormat` property on this page: <https://cloud.google.com/bigquery/docs/reference/rest/v2/jobs>
- **rewind** (*bool*) – If True, seek to the beginning of the file handle before writing the file.
- **size** (*int*) – The number of bytes to read from the file handle. If not provided, we'll try to guess the size using `os.fstat()`. (If the file handle is not from the filesystem this won't be possible.)
- **num_retries** (*int*) – Number of upload retries. Defaults to 6.
- **allow_jagged_rows** (*bool*) – job configuration option; see `google.cloud.bigquery.job.LoadJob()`.
- **allow_quoted_newlines** (*bool*) – job configuration option; see `google.cloud.bigquery.job.LoadJob()`.
- **create_disposition** (*str*) – job configuration option; see `google.cloud.bigquery.job.LoadJob()`.
- **encoding** (*str*) – job configuration option; see `google.cloud.bigquery.job.LoadJob()`.
- **field_delimiter** (*str*) – job configuration option; see `google.cloud.bigquery.job.LoadJob()`.
- **ignore_unknown_values** (*bool*) – job configuration option; see `google.cloud.bigquery.job.LoadJob()`.
- **max_bad_records** (*int*) – job configuration option; see `google.cloud.bigquery.job.LoadJob()`.
- **quote_character** (*str*) – job configuration option; see `google.cloud.bigquery.job.LoadJob()`.
- **skip_leading_rows** (*int*) – job configuration option; see `google.cloud.bigquery.job.LoadJob()`.
- **write_disposition** (*str*) – job configuration option; see `google.cloud.bigquery.job.LoadJob()`.
- **client** (*Client*) – (Optional) The client to use. If not passed, falls back to the `client` stored on the current table.
- **job_name** (*str*) – Optional. The id of the job. Generated if not explicitly passed in.
- **null_marker** (*str*) – Optional. A custom null marker (example: "N")

Return type `LoadTableFromStorageJob`

Returns the job instance used to load the data (e.g., for querying status). Note that the job is already started: do not call `job.begin()`.

Raises `ValueError` if `size` is not passed in and can not be determined, or if the `file_obj` can be detected to be a file opened in text mode.

view_query

SQL query defining the table as a view.

Return type `str`, or `NoneType`

Returns The query as set by the user, or `None` (the default).

view_use_legacy_sql

Specifies whether to execute the view with legacy or standard SQL.

If not set, `None` is returned. BigQuery's default mode is equivalent to `useLegacySql = True`.

Return type `bool`, or `NoneType`

Returns The boolean for `view.useLegacySql` as set by the user, or `None` (the default).

5.7 Authentication / Configuration

- Use `Client` objects to configure your applications.
- `Client` objects hold both a `project` and an authenticated connection to the BigQuery service.
- The authentication credentials can be implicitly determined from the environment or directly via `from_service_account_json` and `from_service_account_p12`.
- After setting `GOOGLE_APPLICATION_CREDENTIALS` and `GOOGLE_CLOUD_PROJECT` environment variables, create an instance of `Client`.

```
>>> from google.cloud import bigquery
>>> client = bigquery.Client()
```

5.8 Projects

A project is the top-level container in the BigQuery API: it is tied closely to billing, and can provide default access control across all its datasets. If no `project` is passed to the client container, the library attempts to infer a project using the environment (including explicit environment variables, GAE, and GCE).

To override the project inferred from the environment, pass an explicit `project` to the constructor, or to either of the alternative classmethod factories:

```
>>> from google.cloud import bigquery
>>> client = bigquery.Client(project='PROJECT_ID')
```

5.8.1 Project ACLs

Each project has an access control list granting reader / writer / owner permission to one or more entities. This list cannot be queried or set via the API: it must be managed using the Google Developer Console.

5.9 Datasets

A dataset represents a collection of tables, and applies several default policies to tables as they are created:

- An access control list (ACL). When created, a dataset has an ACL which maps to the ACL inherited from its project.
- A default table expiration period. If set, tables created within the dataset will have the value as their expiration period.

5.9.1 Dataset operations

List datasets for the client's project:

```
for dataset in client.list_datasets(): # API request(s)
    do_something_with(dataset)
```

Create a new dataset for the client's project:

```
dataset = client.dataset(DATASET_NAME)
dataset.create() # API request
```

Check for the existence of a dataset:

```
assert not dataset.exists() # API request
dataset.create() # API request
assert dataset.exists() # API request
```

Refresh metadata for a dataset (to pick up changes made by another client):

```
assert dataset.description == ORIGINAL_DESCRIPTION
dataset.description = LOCALLY_CHANGED_DESCRIPTION
assert dataset.description == LOCALLY_CHANGED_DESCRIPTION
dataset.reload() # API request
assert dataset.description == ORIGINAL_DESCRIPTION
```

Patch metadata for a dataset:

```
ONE_DAY_MS = 24 * 60 * 60 * 1000
assert dataset.description == ORIGINAL_DESCRIPTION
dataset.patch(
    description=PATCHED_DESCRIPTION,
    default_table_expiration_ms=ONE_DAY_MS
) # API request
assert dataset.description == PATCHED_DESCRIPTION
assert dataset.default_table_expiration_ms == ONE_DAY_MS
```

Replace the ACL for a dataset, and update all writeable fields:

```
>>> from google.cloud import bigquery
>>> client = bigquery.Client()
>>> dataset = client.dataset('dataset_name')
>>> dataset.get() # API request
>>> acl = list(dataset.acl)
>>> acl.append(bigquery.Access(role='READER', entity_type='domain', entity='example.
↪com'))
```

```
>>> dataset.acl = acl
>>> dataset.update() # API request
```

Delete a dataset:

```
assert dataset.exists() # API request
dataset.delete()
assert not dataset.exists() # API request
```

5.10 Tables

Tables exist within datasets. List tables for the dataset:

```
tables = list(dataset.list_tables()) # API request(s)
assert len(tables) == 0
table = dataset.table(TABLE_NAME)
table.view_query = QUERY
table.create() # API request
tables = list(dataset.list_tables()) # API request(s)
assert len(tables) == 1
assert tables[0].name == TABLE_NAME
```

Create a table:

```
table = dataset.table(TABLE_NAME, SCHEMA)
table.create() # API request
```

Check for the existence of a table:

```
table = dataset.table(TABLE_NAME, SCHEMA)
assert not table.exists() # API request
table.create() # API request
assert table.exists() # API request
```

Refresh metadata for a table (to pick up changes made by another client):

```
assert table.friendly_name == ORIGINAL_FRIENDLY_NAME
assert table.description == ORIGINAL_DESCRIPTION
table.friendly_name = LOCALLY_CHANGED_FRIENDLY_NAME
table.description = LOCALLY_CHANGED_DESCRIPTION
table.reload() # API request
assert table.friendly_name == ORIGINAL_FRIENDLY_NAME
assert table.description == ORIGINAL_DESCRIPTION
```

Patch specific properties for a table:

```
assert table.friendly_name == ORIGINAL_FRIENDLY_NAME
assert table.description == ORIGINAL_DESCRIPTION
table.patch(
    friendly_name=PATCHED_FRIENDLY_NAME,
    description=PATCHED_DESCRIPTION,
) # API request
assert table.friendly_name == PATCHED_FRIENDLY_NAME
assert table.description == PATCHED_DESCRIPTION
```


Update all writable metadata for a table

```
assert table.friendly_name == ORIGINAL_FRIENDLY_NAME
assert table.description == ORIGINAL_DESCRIPTION
NEW_SCHEMA = table.schema[:]
NEW_SCHEMA.append(SchemaField('phone', 'string'))
table.friendly_name = UPDATED_FRIENDLY_NAME
table.description = UPDATED_DESCRIPTION
table.schema = NEW_SCHEMA
table.update()           # API request
assert table.friendly_name == UPDATED_FRIENDLY_NAME
assert table.description == UPDATED_DESCRIPTION
assert table.schema == NEW_SCHEMA
```

Get rows from a table's data:

```
for row in table.fetch_data():
    do_something(row)
```

Insert rows into a table's data:

```
ROWS_TO_INSERT = [
    (u'Phred Phlyntstone', 32),
    (u'Wylma Phlyntstone', 29),
]

table.insert_data(ROWS_TO_INSERT)
```

Upload table data from a file:

```
writer = csv.writer(csv_file)
writer.writerow((b'full_name', b'age'))
writer.writerow((b'Phred Phlyntstone', b'32'))
writer.writerow((b'Wylma Phlyntstone', b'29'))
csv_file.flush()

with open(csv_file.name, 'rb') as readable:
    table.upload_from_file(
        readable, source_format='CSV', skip_leading_rows=1)
```

Delete a table:

```
assert table.exists()           # API request
table.delete()                  # API request
assert not table.exists()       # API request
```

5.11 Jobs

Jobs describe actions performed on data in BigQuery tables:

- Load data into a table
- Run a query against data in one or more tables
- Extract data from a table
- Copy a table

List jobs for a project:

```
job_iterator = client.list_jobs()
for job in job_iterator:    # API request(s)
    do_something_with(job)
```

5.11.1 Querying data (synchronous)

Run a query which can be expected to complete within bounded time:

```
query = client.run_sync_query(LIMITED)
query.timeout_ms = TIMEOUT_MS
query.run()                # API request

assert query.complete
assert len(query.rows) == LIMIT
assert [field.name for field in query.schema] == ['name']
```

Run a query using a named query parameter:

```
from google.cloud.bigquery import ScalarQueryParameter
param = ScalarQueryParameter('state', 'STRING', 'TX')
query = client.run_sync_query(LIMITED, query_parameters=[param])
query.use_legacy_sql = False
query.timeout_ms = TIMEOUT_MS
query.run()                # API request

assert query.complete
assert len(query.rows) == LIMIT
assert [field.name for field in query.schema] == ['name']
```

If the rows returned by the query do not fit into the initial response, then we need to fetch the remaining rows via `fetch_data()`:

```
query = client.run_sync_query(LIMITED)
query.timeout_ms = TIMEOUT_MS
query.max_results = PAGE_SIZE
query.run()                # API request

assert query.complete
assert query.page_token is not None
assert len(query.rows) == PAGE_SIZE
assert [field.name for field in query.schema] == ['name']

iterator = query.fetch_data()    # API request(s) during iteration
for row in iterator:
    do_something_with(row)
```

If the query takes longer than the timeout allowed, `query.complete` will be `False`. In that case, we need to poll the associated job until it is done, and then fetch the results:

```
query = client.run_sync_query(QUERY)
query.timeout_ms = TIMEOUT_MS
query.use_query_cache = False
query.run()                # API request

assert not query.complete
```

```

job = query.job
job.reload()                                # API request
retry_count = 0

while retry_count < 10 and job.state != u'DONE':
    time.sleep(1.5**retry_count)            # exponential backoff
    retry_count += 1
    job.reload()                            # API request

assert job.state == u'DONE'

iterator = query.fetch_data()                # API request(s) during iteration
for row in iterator:
    do_something_with(row)

```

5.11.2 Querying data (asynchronous)

Background a query, loading the results into a table:

```

>>> from google.cloud import bigquery
>>> client = bigquery.Client()
>>> query = """\
SELECT firstname + ' ' + last_name AS full_name,
       FLOOR(DATEDIFF(CURRENT_DATE(), birth_date) / 365) AS age
FROM dataset_name.persons
"""
>>> dataset = client.dataset('dataset_name')
>>> table = dataset.table(name='person_ages')
>>> job = client.run_async_query('fullname-age-query-job', query)
>>> job.destination = table
>>> job.write_disposition= 'WRITE_TRUNCATE'
>>> job.name
'fullname-age-query-job'
>>> job.job_type
'query'
>>> job.created
None
>>> job.state
None

```

Note:

- The created and state fields are not set until the job is submitted to the BigQuery back-end.
-

Then, begin executing the job on the server:

```

>>> job.begin()    # API call
>>> job.created
datetime.datetime(2015, 7, 23, 9, 30, 20, 268260, tzinfo=<UTC>)
>>> job.state
'RUNNING'

```

Poll until the job is complete:

```
>>> import time
>>> retry_count = 100
>>> while retry_count > 0 and job.state != 'DONE':
...     retry_count -= 1
...     time.sleep(10)
...     job.reload() # API call
>>> job.state
'done'
>>> job.ended
datetime.datetime(2015, 7, 23, 9, 30, 21, 334792, tzinfo=<UTC>)
```

Retrieve the results:

```
>>> results = job.results()
>>> rows, total_count, token = query.fetch_data() # API request
>>> while True:
...     do_something_with(rows)
...     if token is None:
...         break
...     rows, total_count, token = query.fetch_data(
...         page_token=token) # API request
```

5.11.3 Inserting data (asynchronous)

Start a job loading data asynchronously from a set of CSV files, located on Google Cloud Storage, appending rows into an existing table. First, create the job locally:

```
>>> from google.cloud import bigquery
>>> from google.cloud.bigquery import SchemaField
>>> client = bigquery.Client()
>>> table = dataset.table(name='person_ages')
>>> table.schema = [
...     SchemaField('full_name', 'STRING', mode='required'),
...     SchemaField('age', 'INTEGER', mode='required')]
>>> job = client.load_table_from_storage(
...     'load-from-storage-job', table, 'gs://bucket-name/object-prefix*')
>>> job.source_format = 'CSV'
>>> job.skip_leading_rows = 1 # count of skipped header rows
>>> job.write_disposition = 'WRITE_TRUNCATE'
>>> job.name
'load-from-storage-job'
>>> job.job_type
'load'
>>> job.created
None
>>> job.state
None
```

Note:

- `google.cloud.bigquery` generates a UUID for each job.
 - The `created` and `state` fields are not set until the job is submitted to the BigQuery back-end.
-

Then, begin executing the job on the server:

```
>>> job.begin() # API call
>>> job.created
datetime.datetime(2015, 7, 23, 9, 30, 20, 268260, tzinfo=<UTC>)
>>> job.state
'RUNNING'
```

Poll until the job is complete:

```
>>> import time
>>> retry_count = 100
>>> while retry_count > 0 and job.state != 'DONE':
...     retry_count -= 1
...     time.sleep(10)
...     job.reload() # API call
>>> job.state
'done'
>>> job.ended
datetime.datetime(2015, 7, 23, 9, 30, 21, 334792, tzinfo=<UTC>)
```

5.11.4 Exporting data (async)

Start a job exporting a table's data asynchronously to a set of CSV files, located on Google Cloud Storage. First, create the job locally:

```
>>> from google.cloud import bigquery
>>> client = bigquery.Client()
>>> table = dataset.table(name='person_ages')
>>> job = client.extract_table_to_storage(
...     'extract-person-ages-job', table,
...     'gs://bucket-name/export-prefix*.csv')
... job.destination_format = 'CSV'
... job.print_header = True
... job.write_disposition = 'WRITE_TRUNCATE'
>>> job.name
'extract-person-ages-job'
>>> job.job_type
'extract'
>>> job.created
None
>>> job.state
None
```

Note:

- `google.cloud.bigquery` generates a UUID for each job.
- The `created` and `state` fields are not set until the job is submitted to the BigQuery back-end.

Then, begin executing the job on the server:

```
>>> job.begin() # API call
>>> job.created
datetime.datetime(2015, 7, 23, 9, 30, 20, 268260, tzinfo=<UTC>)
>>> job.state
'RUNNING'
```

Poll until the job is complete:

```
>>> import time
>>> retry_count = 100
>>> while retry_count > 0 and job.state != 'DONE':
...     retry_count -= 1
...     time.sleep(10)
...     job.reload() # API call
>>> job.state
'done'
>>> job.ended
datetime.datetime(2015, 7, 23, 9, 30, 21, 334792, tzinfo=<UTC>)
```

5.11.5 Copy tables (async)

First, create the job locally:

```
>>> from google.cloud import bigquery
>>> client = bigquery.Client()
>>> source_table = dataset.table(name='person_ages')
>>> destination_table = dataset.table(name='person_ages_copy')
>>> job = client.copy_table(
...     'copy-table-job', destination_table, source_table)
>>> job.name
'copy-table-job'
>>> job.job_type
'copy'
>>> job.created
None
>>> job.state
None
```

Note:

- `google.cloud.bigquery` generates a UUID for each job.
 - The `created` and `state` fields are not set until the job is submitted to the BigQuery back-end.
-

Then, begin executing the job on the server:

```
>>> job.begin() # API call
>>> job.created
datetime.datetime(2015, 7, 23, 9, 30, 20, 268260, tzinfo=<UTC>)
>>> job.state
'RUNNING'
```

Poll until the job is complete:

```
>>> import time
>>> retry_count = 100
>>> while retry_count > 0 and job.state != 'DONE':
...     retry_count -= 1
...     time.sleep(10)
...     job.reload() # API call
>>> job.state
'done'
```

```
>>> job.ended  
datetime.datetime(2015, 7, 23, 9, 30, 21, 334792, tzinfo=<UTC>)
```


6.1 Base for Everything

To use the API, the *Client* class defines a high-level interface which handles authorization and creating other objects:

```
from google.cloud.bigtable.client import Client
client = Client()
```

6.1.1 Long-lived Defaults

When creating a *Client*, the *user_agent* argument has sensible default (DEFAULT_USER_AGENT). However, you may over-ride it and the value will be used throughout all API requests made with the *client* you create.

6.1.2 Configuration

- For an overview of authentication in `google-cloud-python`, see [Authentication](#).
- In addition to any authentication configuration, you can also set the `GOOGLE_CLOUD_PROJECT` environment variable for the Google Cloud Console project you'd like to interact with. If your code is running in Google App Engine or Google Compute Engine the project will be detected automatically. (Setting this environment variable is not required, you may instead pass the `project` explicitly when constructing a *Client*).
- After configuring your environment, create a *Client*

```
>>> from google.cloud import bigtable
>>> client = bigtable.Client()
```

or pass in credentials and project explicitly

```
>>> from google.cloud import bigtable
>>> client = bigtable.Client(project='my-project', credentials=creds)
```

Tip: Be sure to use the **Project ID**, not the **Project Number**.

6.1.3 Admin API Access

If you'll be using your client to make [Instance Admin](#) and [Table Admin](#) API requests, you'll need to pass the `admin` argument:

```
client = bigtable.Client(admin=True)
```

6.1.4 Read-Only Mode

If, on the other hand, you only have (or want) read access to the data, you can pass the `read_only` argument:

```
client = bigtable.Client(read_only=True)
```

This will ensure that the `READ_ONLY_SCOPE` is used for API requests (so any accidental requests that would modify data will fail).

6.1.5 Next Step

After a [Client](#), the next highest-level object is an [Instance](#). You'll need one before you can interact with tables or data.

Head next to learn about the [Instance Admin API](#).

6.2 Client

Parent client for calling the Google Cloud Bigtable API.

This is the base from which all interactions with the API occur.

In the hierarchy of API concepts

- a [Client](#) owns an [Instance](#)
- an [Instance](#) owns a [Table](#)
- a [Table](#) owns a [ColumnFamily](#)
- a [Table](#) owns a [Row](#) (and all the cells in the row)

```
google.cloud.bigtable.client.ADMIN_SCOPE = 'https://www.googleapis.com/auth/bigtable.admin'
Scope for interacting with the Cluster Admin and Table Admin APIs.
```

```
class google.cloud.bigtable.client.Client (project=None, credentials=None,
                                           read_only=False, admin=False,
                                           user_agent='gcloud-python/0.27.1')
```

Bases: [google.cloud.client.ClientWithProject](#)

Client for interacting with Google Cloud Bigtable API.

Note: Since the Cloud Bigtable API requires the gRPC transport, no `_http` argument is accepted by this class.

Parameters

- **project** (*str* or *unicode*) – (Optional) The ID of the project which owns the instances, tables and data. If not provided, will attempt to determine from the environment.
- **credentials** (*Credentials*) – (Optional) The OAuth2 Credentials to use for this client. If not passed, falls back to the default inferred from the environment.
- **read_only** (*bool*) – (Optional) Boolean indicating if the data scope should be for reading only (or for writing as well). Defaults to *False*.
- **admin** (*bool*) – (Optional) Boolean indicating if the client will be used to interact with the Instance Admin or Table Admin APIs. This requires the *ADMIN_SCOPE*. Defaults to *False*.
- **user_agent** (*str*) – (Optional) The user agent to be used with API request. Defaults to *DEFAULT_USER_AGENT*.

Raises *ValueError* if both *read_only* and *admin* are *True*

copy()

Make a copy of this client.

Copies the local data stored as simple types but does not copy the current state of any open connections with the Cloud Bigtable API.

Return type *Client*

Returns A copy of the current client.

credentials

Getter for client's credentials.

Return type *OAuth2Credentials*

Returns The credentials stored on the client.

instance (*instance_id*, *location*=*'see-existing-cluster'*, *display_name*=*None*, *serve_nodes*=*3*)

Factory to create a instance associated with this client.

Parameters

- **instance_id** (*str*) – The ID of the instance.
- **location** (*str*) – location name, in form *projects/<project>/locations/<location>*; used to set up the instance's cluster.
- **display_name** (*str*) – (Optional) The display name for the instance in the Cloud Console UI. (Must be between 4 and 30 characters.) If this value is not set in the constructor, will fall back to the instance ID.
- **serve_nodes** (*int*) – (Optional) The number of nodes in the instance's cluster; used to set up the instance's cluster.

Return type *Instance*

Returns an instance owned by this client.

list_instances()

List instances owned by the project.

Return type *tuple*

Returns A pair of results, the first is a list of *Instance* objects returned and the second is a list of strings (the failed locations in the request).

project_name

Project name to be used with Instance Admin API.

Note: This property will not change if `project` does not, but the return value is not cached.

The project name is of the form

`"projects/{project}"`

Return type `str`

Returns The project name to be used with the Cloud Bigtable Admin API RPC service.

```
google.cloud.bigtable.client.DATA_API_HOST = 'bigtable.googleapis.com'
```

Data API request host.

```
google.cloud.bigtable.client.DATA_SCOPE = 'https://www.googleapis.com/auth/bigtable.data'
```

Scope for reading and writing table data.

```
google.cloud.bigtable.client.INSTANCE_ADMIN_HOST = 'bigtableadmin.googleapis.com'
```

Cluster Admin API request host.

```
google.cloud.bigtable.client.READ_ONLY_SCOPE = 'https://www.googleapis.com/auth/bigtable.data'
```

Scope for reading table data.

```
google.cloud.bigtable.client.TABLE_ADMIN_HOST = 'bigtableadmin.googleapis.com'
```

Table Admin API request host.

6.3 Cluster

User friendly container for Google Cloud Bigtable Cluster.

```
class google.cloud.bigtable.cluster.Cluster(cluster_id, instance, serve_nodes=3)
```

Bases: `object`

Representation of a Google Cloud Bigtable Cluster.

We can use a *Cluster* to:

- `reload()` itself
- `create()` itself
- `update()` itself
- `delete()` itself

Note: For now, we leave out the `default_storage_type` (an enum) which if not sent will end up as `data_v2_pb2.STORAGE_SSD`.

Parameters

- **cluster_id** (*str*) – The ID of the cluster.
- **instance** (*Instance*) – The instance where the cluster resides.
- **serve_nodes** (*int*) – (Optional) The number of nodes in the cluster. Defaults to `DEFAULT_SERVE_NODES`.

copy()

Make a copy of this cluster.

Copies the local data stored as simple types and copies the client attached to this instance.

Return type *Cluster*

Returns A copy of the current cluster.

create()

Create this cluster.

Note: Uses the `project`, `instance` and `cluster_id` on the current *Cluster* in addition to the `serve_nodes`. To change them before creating, reset the values via

```
cluster.serve_nodes = 8
cluster.cluster_id = 'i-changed-my-mind'
```

before calling `create()`.

Return type *Operation*

Returns The long-running operation corresponding to the create operation.

delete()

Delete this cluster.

Marks a cluster and all of its tables for permanent deletion in 7 days.

Immediately upon completion of the request:

- Billing will cease for all of the cluster's reserved resources.
- The cluster's `delete_time` field will be set 7 days in the future.

Soon afterward:

- All tables within the cluster will become unavailable.

At the cluster's `delete_time`:

- The cluster and **all of its tables** will immediately and irrevocably disappear from the API, and their data will be permanently deleted.

classmethod from_pb(cluster_pb, instance)

Creates a cluster instance from a protobuf.

Parameters

- **cluster_pb** (`instance_pb2.Cluster`) – A cluster protobuf object.
- **instance** (`Instance`) – The instance that owns the cluster.

Return type *Cluster*

Returns The cluster parsed from the protobuf response.

Raises `ValueError` if the cluster name does not match `projects/{project}/instances/{instance}/clusters/{cluster_id}` or if the parsed project ID does not match the project ID on the client.

name

Cluster name used in requests.

Note: This property will not change if `_instance` and `cluster_id` do not, but the return value is not cached.

The cluster name is of the form

```
"projects/{project}/instances/{instance}/clusters/{cluster_id}"
```

Return type `str`

Returns The cluster name.

reload()

Reload the metadata for this cluster.

update()

Update this cluster.

Note: Updates the `serve_nodes`. If you'd like to change them before updating, reset the values via

```
cluster.serve_nodes = 8
```

before calling `update()`.

Return type `Operation`

Returns The long-running operation corresponding to the update operation.

`google.cloud.bigtable.cluster.DEFAULT_SERVE_NODES = 3`

Default number of nodes to use when creating a cluster.

6.4 Instance

User-friendly container for Google Cloud Bigtable Instance.

```
class google.cloud.bigtable.instance.Instance(instance_id, client, location_id='see-  
existing-cluster', display_name=None,  
serve_nodes=3)
```

Bases: `object`

Representation of a Google Cloud Bigtable Instance.

We can use an *Instance* to:

- `reload()` itself
- `create()` itself
- `update()` itself
- `delete()` itself

Note: For now, we leave out the `default_storage_type` (an enum) which if not sent will end up as `data_v2_pb2.STORAGE_SSD`.

Parameters

- **instance_id** (*str*) – The ID of the instance.
- **client** (*Client*) – The client that owns the instance. Provides authorization and a project ID.
- **location_id** (*str*) – ID of the location in which the instance will be created. Required for instances which do not yet exist.
- **display_name** (*str*) – (Optional) The display name for the instance in the Cloud Console UI. (Must be between 4 and 30 characters.) If this value is not set in the constructor, will fall back to the instance ID.
- **serve_nodes** (*int*) – (Optional) The number of nodes in the instance's cluster; used to set up the instance's cluster.

cluster (*cluster_id*, *serve_nodes=3*)

Factory to create a cluster associated with this client.

Parameters

- **cluster_id** (*str*) – The ID of the cluster.
- **serve_nodes** (*int*) – (Optional) The number of nodes in the cluster. Defaults to 3.

Return type *Cluster*

Returns The cluster owned by this client.

copy ()

Make a copy of this instance.

Copies the local data stored as simple types and copies the client attached to this instance.

Return type *Instance*

Returns A copy of the current instance.

create ()

Create this instance.

Note: Uses the `project` and `instance_id` on the current *Instance* in addition to the `display_name`. To change them before creating, reset the values via

```
instance.display_name = 'New display name'
instance.instance_id = 'i-changed-my-mind'
```

before calling *create* ().

Return type *Operation*

Returns The long-running operation corresponding to the create operation.

delete()

Delete this instance.

Marks an instance and all of its tables for permanent deletion in 7 days.

Immediately upon completion of the request:

- Billing will cease for all of the instance's reserved resources.
- The instance's `delete_time` field will be set 7 days in the future.

Soon afterward:

- All tables within the instance will become unavailable.

At the instance's `delete_time`:

- The instance and **all of its tables** will immediately and irrevocably disappear from the API, and their data will be permanently deleted.

classmethod from_pb(instance_pb, client)

Creates an instance instance from a protobuf.

Parameters

- **instance_pb** (`instance_pb2.Instance`) – An instance protobuf object.
- **client** (`Client`) – The client that owns the instance.

Return type `Instance`

Returns The instance parsed from the protobuf response.

Raises `ValueError` if the instance name does not match `projects/{project}/instances/{instance_id}` or if the parsed project ID does not match the project ID on the client.

list_clusters()

Lists clusters in this instance.

Return type `tuple`

Returns A pair of results, the first is a list of `Cluster`s returned and the second is a list of strings (the failed locations in the request).

list_tables()

List the tables in this instance.

Return type list of `Table`

Returns The list of tables owned by the instance.

Raises `ValueError` if one of the returned tables has a name that is not of the expected format.

name

Instance name used in requests.

Note: This property will not change if `instance_id` does not, but the return value is not cached.

The instance name is of the form

`"projects/{project}/instances/{instance_id}"`

Return type `str`

Returns The instance name.

reload()

Reload the metadata for this instance.

table(*table_id*)

Factory to create a table associated with this instance.

Parameters *table_id* (*str*) – The ID of the table.

Return type *Table*

Returns The table owned by this instance.

update()

Update this instance.

Note: Updates the `display_name`. To change that value before updating, reset its values via

```
instance.display_name = 'New display name'
```

before calling `update()`.

6.5 Instance Admin API

After creating a *Client*, you can interact with individual instances for a project.

6.5.1 List Instances

If you want a comprehensive list of all existing instances, make a `ListInstances` API request with *Client*. `list_instances()`:

```
instances = client.list_instances()
```

6.5.2 Instance Factory

To create an *Instance* object:

```
instance = client.instance(instance_id, location_id,
                           display_name=display_name)
```

- `location_id` is the ID of the location in which the instance's cluster will be hosted, e.g. 'us-central1-c'. `location_id` is required for instances which do not already exist.
- `display_name` is optional. When not provided, `display_name` defaults to the `instance_id` value.

You can also use `Client.instance()` to create a local wrapper for instances that have already been created with the API, or through the web console:

```
instance = client.instance(existing_instance_id)
instance.reload()
```

6.5.3 Create a new Instance

After creating the instance object, make a `CreateInstance` API request with `create()`:

```
instance.display_name = 'My very own instance'
instance.create()
```

6.5.4 Check on Current Operation

Note: When modifying an instance (via a `CreateInstance` request), the Bigtable API will return a `long-running operation` and a corresponding `Operation` object will be returned by `create()`.

You can check if a long-running operation (for a `create()` has finished by making a `GetOperation` request with `Operation.finished()`:

```
>>> operation = instance.create()
>>> operation.finished()
True
```

Note: Once an `Operation` object has returned `True` from `finished()`, the object should not be re-used. Subsequent calls to `finished()` will result in a `ValueError`.

6.5.5 Get metadata for an existing Instance

After creating the instance object, make a `GetInstance` API request with `reload()`:

```
instance.reload()
```

This will load `display_name` for the existing instance object.

6.5.6 Update an existing Instance

After creating the instance object, make an `UpdateInstance` API request with `update()`:

```
client.display_name = 'New display_name'
instance.update()
```

6.5.7 Delete an existing Instance

Make a `DeleteInstance` API request with `delete()`:

```
instance.delete()
```

6.5.8 Next Step

Now we go down the hierarchy from *Instance* to a *Table*.

Head next to learn about the *Table Admin API*.

6.6 Table

User-friendly container for Google Cloud Bigtable Table.

class google.cloud.bigtable.table.**Table** (*table_id*, *instance*)

Bases: `object`

Representation of a Google Cloud Bigtable Table.

Note: We don't define any properties on a table other than the name. The only other fields are `column_families` and `granularity`, The `column_families` are not stored locally and `granularity` is an enum with only one value.

We can use a *Table* to:

- `create()` the table
- `rename()` the table
- `delete()` the table
- `list_column_families()` in the table

Parameters

- **table_id** (*str*) – The ID of the table.
- **instance** (*Instance*) – The instance that owns the table.

column_family (*column_family_id*, *gc_rule=None*)

Factory to create a column family associated with this table.

Parameters

- **column_family_id** (*str*) – The ID of the column family. Must be of the form `[_a-zA-Z0-9][_\.a-zA-Z0-9]*`.
- **gc_rule** (*GarbageCollectionRule*) – (Optional) The garbage collection settings for this column family.

Return type *ColumnFamily*

Returns A column family owned by this table.

create (*initial_split_keys=None*, *column_families=()*)

Creates this table.

Note: A create request returns a `_generated.table_pb2.Table` but we don't use this response.

Parameters

- **initial_split_keys** (*list*) – (Optional) List of row keys that will be used to initially split the table into several tablets (Tablets are similar to HBase regions). Given two split keys, "s1" and "s2", three tablets will be created, spanning the key ranges: [, s1), [s1, s2), [s2,).
- **column_families** (*list*) – (Optional) List or other iterable of *ColumnFamily* instances.

delete()

Delete this table.

list_column_families()

List the column families owned by this table.

Return type *dict*

Returns Dictionary of column families attached to this table. Keys are strings (column family names) and values are *ColumnFamily* instances.

Raises *ValueError* if the column family name from the response does not agree with the computed name from the column family ID.

mutate_rows (*rows*)

Mutates multiple rows in bulk.

The method tries to update all specified rows. If some of the rows weren't updated, it would not remove mutations. They can be applied to the row separately. If row mutations finished successfully, they would be cleaned up.

Parameters *rows* (*list*) – List or other iterable of *DirectRow* instances.

Return type *list*

Returns A list of response statuses (*google.rpc.status_pb2.Status*) corresponding to success or failure of each row mutation sent. These will be in the same order as the *rows*.

name

Table name used in requests.

Note: This property will not change if `table_id` does not, but the return value is not cached.

The table name is of the form

"projects/.../instances/.../tables/{table_id}"

Return type *str*

Returns The table name.

read_row (*row_key*, *filter_=None*)

Read a single row from this table.

Parameters

- **row_key** (*bytes*) – The key of the row to read from.
- **filter** (*RowFilter*) – (Optional) The filter to apply to the contents of the row. If unset, returns the entire row.

Return type *PartialRowData*, *NoneType*

Returns The contents of the row if any chunks were returned in the response, otherwise *None*.

Raises `ValueError` if a commit row chunk is never encountered.

read_rows (*start_key=None, end_key=None, limit=None, filter_=None, end_inclusive=False*)
Read rows from this table.

Parameters

- **start_key** (*bytes*) – (Optional) The beginning of a range of row keys to read from. The range will include *start_key*. If left empty, will be interpreted as the empty string.
- **end_key** (*bytes*) – (Optional) The end of a range of row keys to read from. The range will not include *end_key*. If left empty, will be interpreted as an infinite string.
- **limit** (*int*) – (Optional) The read will terminate after committing to N rows' worth of results. The default (zero) is to return all results.
- **filter** (*RowFilter*) – (Optional) The filter to apply to the contents of the specified row(s). If unset, reads every column in each row.
- **end_inclusive** (*bool*) – (Optional) Whether the *end_key* should be considered inclusive. The default is `False` (exclusive).

Return type *PartialRowsData*

Returns A *PartialRowsData* convenience wrapper for consuming the streamed results.

row (*row_key, filter_=None, append=False*)
Factory to create a row associated with this table.

Warning: At most one of *filter_* and *append* can be used in a Row.

Parameters

- **row_key** (*bytes*) – The key for the row being created.
- **filter** (*RowFilter*) – (Optional) Filter to be used for conditional mutations. See *ConditionalRow* for more details.
- **append** (*bool*) – (Optional) Flag to determine if the row should be used for append mutations.

Return type *Row*

Returns A row owned by this table.

Raises `ValueError` if both *filter_* and *append* are used.

sample_row_keys ()
Read a sample of row keys in the table.

The returned row keys will delimit contiguous sections of the table of approximately equal size, which can be used to break up the data for distributed tasks like mapreduces.

The elements in the iterator are a *SampleRowKeys* response and they have the properties *offset_bytes* and *row_key*. They occur in sorted order. The table might have contents before the first row key in the list and after the last one, but a key containing the empty string indicates “end of table” and will be the last response given, if present.

Note: Row keys in this list may not have ever been written to or read from, and users should therefore not make any assumptions about the row key structure that are specific to their use case.

The `offset_bytes` field on a response indicates the approximate total storage space used by all rows in the table which precede `row_key`. Buffering the contents of all rows between two subsequent samples would require space roughly equal to the difference in their `offset_bytes` fields.

Return type `GrpcRendezvous`

Returns A cancel-able iterator. Can be consumed by calling `next()` or by casting to a `list` and can be cancelled by calling `cancel()`.

exception `google.cloud.bigtable.table.TableMismatchError`

Bases: `exceptions.ValueError`

Row from another table.

exception `google.cloud.bigtable.table.TooManyMutationsError`

Bases: `exceptions.ValueError`

The number of mutations for bulk request is too big.

6.7 Table Admin API

After creating an *Instance*, you can interact with individual tables, groups of tables or column families within a table.

6.7.1 List Tables

If you want a comprehensive list of all existing tables in a instance, make a `ListTables` API request with *Instance*. `list_tables()`:

```
>>> instance.list_tables()
[<google.cloud.bigtable.table.Table at 0x7ff6a1de8f50>,
 <google.cloud.bigtable.table.Table at 0x7ff6a1de8350>]
```

6.7.2 Table Factory

To create a *Table* object:

```
table = instance.table(table_id)
```

Even if this *Table* already has been created with the API, you'll want this object to use as a parent of a *ColumnFamily* or *Row*.

6.7.3 Create a new Table

After creating the table object, make a `CreateTable` API request with `create()`:

```
table.create()
```

If you would like to initially split the table into several tablets (tablets are similar to HBase regions):

```
table.create(initial_split_keys=['s1', 's2'])
```

6.7.4 Delete an existing Table

Make a `DeleteTable` API request with `delete()`:

```
table.delete()
```

6.7.5 List Column Families in a Table

Though there is no **official** method for retrieving `column families` associated with a table, the `GetTable` API method returns a table object with the names of the column families.

To retrieve the list of column families use `list_column_families()`:

```
column_families = table.list_column_families()
```

6.7.6 Column Family Factory

To create a `ColumnFamily` object:

```
column_family = table.column_family(column_family_id)
```

There is no real reason to use this factory unless you intend to create or delete a column family.

In addition, you can specify an optional `gc_rule` (a `GarbageCollectionRule` or similar):

```
column_family = table.column_family(column_family_id,  
                                     gc_rule=gc_rule)
```

This rule helps the backend determine when and how to clean up old cells in the column family.

See *Column Families* for more information about `GarbageCollectionRule` and related classes.

6.7.7 Create a new Column Family

After creating the column family object, make a `CreateColumnFamily` API request with `ColumnFamily.create()`

```
column_family.create()
```

6.7.8 Delete an existing Column Family

Make a `DeleteColumnFamily` API request with `ColumnFamily.delete()`

```
column_family.delete()
```

6.7.9 Update an existing Column Family

Make an `UpdateColumnFamily` API request with `ColumnFamily.delete()`

```
column_family.update()
```

6.7.10 Next Step

Now we go down the final step of the hierarchy from *Table* to *Row* as well as streaming data directly via a *Table*. Head next to learn about the *Data API*.

6.8 Column Families

When creating a *ColumnFamily*, it is possible to set garbage collection rules for expired data.

By setting a rule, cells in the table matching the rule will be deleted during periodic garbage collection (which executes opportunistically in the background).

The types *MaxAgeGCRule*, *MaxVersionsGCRule*, *GarbageCollectionRuleUnion* and *GarbageCollectionRuleIntersection* can all be used as the optional *gc_rule* argument in the *ColumnFamily* constructor. This value is then used in the *create()* and *update()* methods.

These rules can be nested arbitrarily, with a *MaxAgeGCRule* or *MaxVersionsGCRule* at the lowest level of the nesting:

```
import datetime

max_age = datetime.timedelta(days=3)
rule1 = MaxAgeGCRule(max_age)
rule2 = MaxVersionsGCRule(1)

# Make a composite that matches anything older than 3 days **AND**
# with more than 1 version.
rule3 = GarbageCollectionIntersection(rules=[rule1, rule2])

# Make another composite that matches our previous intersection
# **OR** anything that has more than 3 versions.
rule4 = GarbageCollectionRule(max_num_versions=3)
rule5 = GarbageCollectionUnion(rules=[rule3, rule4])
```

User friendly container for Google Cloud Bigtable Column Family.

```
class google.cloud.bigtable.column_family.ColumnFamily(column_family_id,    table,
                                                         gc_rule=None)
```

Bases: *object*

Representation of a Google Cloud Bigtable Column Family.

We can use a *ColumnFamily* to:

- *create()* itself
- *update()* itself
- *delete()* itself

Parameters

- **column_family_id** (*str*) – The ID of the column family. Must be of the form `[_a-zA-Z0-9][-_a-zA-Z0-9]*`.
- **table** (*Table*) – The table that owns the column family.

- **gc_rule** (*GarbageCollectionRule*) – (Optional) The garbage collection settings for this column family.

create()

Create this column family.

delete()

Delete this column family.

name

Column family name used in requests.

Note: This property will not change if `column_family_id` does not, but the return value is not cached.

The table name is of the form

"projects/.../zones/.../clusters/.../tables/.../columnFamilies/..."

Return type `str`

Returns The column family name.

to_pb()

Converts the column family to a protobuf.

Return type `table_v2_pb2.ColumnFamily`

Returns The converted current object.

update()

Update this column family.

Note: Only the GC rule can be updated. By changing the column family ID, you will simply be referring to a different column family.

class `google.cloud.bigtable.column_family.GCRuleIntersection(rules)`

Bases: `google.cloud.bigtable.column_family.GarbageCollectionRule`

Intersection of garbage collection rules.

Parameters **rules** (*list*) – List of *GarbageCollectionRule*.

to_pb()

Converts the intersection into a single GC rule as a protobuf.

Return type `table_v2_pb2.GcRule`

Returns The converted current object.

class `google.cloud.bigtable.column_family.GCRuleUnion(rules)`

Bases: `google.cloud.bigtable.column_family.GarbageCollectionRule`

Union of garbage collection rules.

Parameters **rules** (*list*) – List of *GarbageCollectionRule*.

to_pb()

Converts the union into a single GC rule as a protobuf.

Return type `table_v2_pb2.GcRule`

Returns The converted current object.

class `google.cloud.bigtable.column_family.GarbageCollectionRule`

Bases: `object`

Garbage collection rule for column families within a table.

Cells in the column family (within a table) fitting the rule will be deleted during garbage collection.

Note: This class is a do-nothing base class for all GC rules.

Note: A string `gc_expression` can also be used with API requests, but that value would be superseded by a `gc_rule`. As a result, we don't support that feature and instead support via native classes.

class `google.cloud.bigtable.column_family.MaxAgeGCRule(max_age)`

Bases: `google.cloud.bigtable.column_family.GarbageCollectionRule`

Garbage collection limiting the age of a cell.

Parameters `max_age` (`datetime.timedelta`) – The maximum age allowed for a cell in the table.

to_pb()

Converts the garbage collection rule to a protobuf.

Return type `table_v2_pb2.GcRule`

Returns The converted current object.

class `google.cloud.bigtable.column_family.MaxVersionsGCRule(max_num_versions)`

Bases: `google.cloud.bigtable.column_family.GarbageCollectionRule`

Garbage collection limiting the number of versions of a cell.

Parameters `max_num_versions` (`int`) – The maximum number of versions

to_pb()

Converts the garbage collection rule to a protobuf.

Return type `table_v2_pb2.GcRule`

Returns The converted current object.

6.9 Bigtable Row

User-friendly container for Google Cloud Bigtable Row.

class `google.cloud.bigtable.row.AppendRow(row_key, table)`

Bases: `google.cloud.bigtable.row.Row`

Google Cloud Bigtable Row for sending append mutations.

These mutations are intended to augment the value of an existing cell and uses the methods:

- `append_cell_value()`
- `increment_cell_value()`

The first works by appending bytes and the second by incrementing an integer (stored in the cell as 8 bytes). In either case, if the cell is empty, assumes the default empty value (empty string for bytes or 0 for integer).

Parameters

- **row_key** (*bytes*) – The key for the current row.
- **table** (*Table*) – The table that owns the row.

append_cell_value (*column_family_id*, *column*, *value*)

Appends a value to an existing cell.

Note: This method adds a read-modify rule protobuf to the accumulated read-modify rules on this row, but does not make an API request. To actually send an API request (with the rules) to the Google Cloud Bigtable API, call `commit()`.

Parameters

- **column_family_id** (*str*) – The column family that contains the column. Must be of the form `[_a-zA-Z0-9][_._a-zA-Z0-9]*`.
- **column** (*bytes*) – The column within the column family where the cell is located.
- **value** (*bytes*) – The value to append to the existing value in the cell. If the targeted cell is unset, it will be treated as containing the empty string.

clear()

Removes all currently accumulated modifications on current row.

commit()

Makes a ReadModifyWriteRow API request.

This commits modifications made by `append_cell_value()` and `increment_cell_value()`. If no modifications were made, makes no API request and just returns `{}`.

Modifies a row atomically, reading the latest existing timestamp / value from the specified columns and writing a new value by appending / incrementing. The new cell created uses either the current server time or the highest timestamp of a cell in that column (if it exceeds the server time).

After committing the accumulated mutations, resets the local mutations.

```
>>> append_row.commit()
{
  u'col-fam-id': {
    b'col-name1': [
      (b'cell-val', datetime.datetime(...)),
      (b'cell-val-newer', datetime.datetime(...)),
    ],
    b'col-name2': [
      (b'altcol-cell-val', datetime.datetime(...)),
    ],
  },
  u'col-fam-id2': {
    b'col-name3-but-other-fam': [
      (b'foo', datetime.datetime(...)),
    ],
  },
}
```

Return type `dict`

Returns The new contents of all modified cells. Returned as a dictionary of column families, each of which holds a dictionary of columns. Each column contains a list of cells modified. Each cell is represented with a two-tuple with the value (in bytes) and the timestamp for the cell.

Raises `ValueError` if the number of mutations exceeds the `MAX_MUTATIONS`.

increment_cell_value (*column_family_id*, *column*, *int_value*)

Increments a value in an existing cell.

Assumes the value in the cell is stored as a 64 bit integer serialized to bytes.

Note: This method adds a read-modify rule protobuf to the accumulated read-modify rules on this row, but does not make an API request. To actually send an API request (with the rules) to the Google Cloud Bigtable API, call `commit()`.

Parameters

- **column_family_id** (*str*) – The column family that contains the column. Must be of the form `[_a-zA-Z0-9][_\.a-zA-Z0-9]*`.
- **column** (*bytes*) – The column within the column family where the cell is located.
- **int_value** (*int*) – The value to increment the existing value in the cell by. If the targeted cell is unset, it will be treated as containing a zero. Otherwise, the targeted cell must contain an 8-byte value (interpreted as a 64-bit big-endian signed integer), or the entire request will fail.

row_key

Row key.

Return type `bytes`

Returns The key for the current row.

table

Row table.

Return type table: `Table`

Returns table: The table that owns the row.

class `google.cloud.bigtable.row.ConditionalRow` (*row_key*, *table*, *filter_*)

Bases: `google.cloud.bigtable.row._SetDeleteRow`

Google Cloud Bigtable Row for sending mutations conditionally.

Each mutation has an associated state: `True` or `False`. When `commit()`-ed, the mutations for the `True` state will be applied if the filter matches any cells in the row, otherwise the `False` state will be applied.

A `ConditionalRow` accumulates mutations in the same way a `DirectRow` does:

- `set_cell()`
- `delete()`
- `delete_cell()`
- `delete_cells()`

with the only change the extra `state` parameter:

```
>>> row_cond = table.row(b'row-key2', filter_=row_filter)
>>> row_cond.set_cell(u'fam', b'col', b'cell-val', state=True)
>>> row_cond.delete_cell(u'fam', b'col', state=False)
```

Note: As with *DirectRow*, to actually send these mutations to the Google Cloud Bigtable API, you must call `commit()`.

Parameters

- **row_key** (*bytes*) – The key for the current row.
- **table** (*Table*) – The table that owns the row.
- **filter** (*RowFilter*) – Filter to be used for conditional mutations.

`clear()`

Removes all currently accumulated mutations on the current row.

`commit()`

Makes a `CheckAndMutateRow` API request.

If no mutations have been created in the row, no request is made.

The mutations will be applied conditionally, based on whether the filter matches any cells in the *ConditionalRow* or not. (Each method which adds a mutation has a `state` parameter for this purpose.)

Mutations are applied atomically and in order, meaning that earlier mutations can be masked / negated by later ones. Cells already present in the row are left unchanged unless explicitly changed by a mutation.

After committing the accumulated mutations, resets the local mutations.

Return type `bool`

Returns Flag indicating if the filter was matched (which also indicates which set of mutations were applied by the server).

Raises `ValueError` if the number of mutations exceeds the `MAX_MUTATIONS`.

`delete(state=True)`

Deletes this row from the table.

Note: This method adds a mutation to the accumulated mutations on this row, but does not make an API request. To actually send an API request (with the mutations) to the Google Cloud Bigtable API, call `commit()`.

Parameters **state** (*bool*) – (Optional) The state that the mutation should be applied in. Defaults to `True`.

`delete_cell(column_family_id, column, time_range=None, state=True)`

Deletes cell in this row.

Note: This method adds a mutation to the accumulated mutations on this row, but does not make an API request. To actually send an API request (with the mutations) to the Google Cloud Bigtable API, call `commit()`.

Parameters

- **column_family_id** (*str*) – The column family that contains the column or columns with cells being deleted. Must be of the form `[_a-zA-Z0-9] [-_.a-zA-Z0-9]*`.
- **column** (*bytes*) – The column within the column family that will have a cell deleted.
- **time_range** (*TimestampRange*) – (Optional) The range of time within which cells should be deleted.
- **state** (*bool*) – (Optional) The state that the mutation should be applied in. Defaults to `True`.

delete_cells (*column_family_id, columns, time_range=None, state=True*)

Deletes cells in this row.

Note: This method adds a mutation to the accumulated mutations on this row, but does not make an API request. To actually send an API request (with the mutations) to the Google Cloud Bigtable API, call `commit()`.

Parameters

- **column_family_id** (*str*) – The column family that contains the column or columns with cells being deleted. Must be of the form `[_a-zA-Z0-9] [-_.a-zA-Z0-9]*`.
- **columns** (*list of str / unicode, or object*) – The columns within the column family that will have cells deleted. If `ALL_COLUMNS` is used then the entire column family will be deleted from the row.
- **time_range** (*TimestampRange*) – (Optional) The range of time within which cells should be deleted.
- **state** (*bool*) – (Optional) The state that the mutation should be applied in. Defaults to `True`.

row_key

Row key.

Return type `bytes`

Returns The key for the current row.

set_cell (*column_family_id, column, value, timestamp=None, state=True*)

Sets a value in this row.

The cell is determined by the `row_key` of this *ConditionalRow* and the `column`. The `column` must be in an existing *ColumnFamily* (as determined by `column_family_id`).

Note: This method adds a mutation to the accumulated mutations on this row, but does not make an API request. To actually send an API request (with the mutations) to the Google Cloud Bigtable API, call `commit()`.

Parameters

- **column_family_id** (*str*) – The column family that contains the column. Must be of the form `[_a-zA-Z0-9] [-_.a-zA-Z0-9]*`.
- **column** (*bytes*) – The column within the column family where the cell is located.

- **value** (bytes or `int`) – The value to set in the cell. If an integer is used, will be interpreted as a 64-bit big-endian signed integer (8 bytes).
- **timestamp** (`datetime.datetime`) – (Optional) The timestamp of the operation.
- **state** (`bool`) – (Optional) The state that the mutation should be applied in. Defaults to `True`.

table

Row table.

Return type table: `Table`

Returns table: The table that owns the row.

class `google.cloud.bigtable.row.DirectRow` (*row_key*, *table*)

Bases: `google.cloud.bigtable.row._SetDeleteRow`

Google Cloud Bigtable Row for sending “direct” mutations.

These mutations directly set or delete cell contents:

- `set_cell()`
- `delete()`
- `delete_cell()`
- `delete_cells()`

These methods can be used directly:

```
>>> row = table.row(b'row-key1')
>>> row.set_cell(u'fam', b'col1', b'cell-val')
>>> row.delete_cell(u'fam', b'col2')
```

Note: A `DirectRow` accumulates mutations locally via the `set_cell()`, `delete()`, `delete_cell()` and `delete_cells()` methods. To actually send these mutations to the Google Cloud Bigtable API, you must call `commit()`.

Parameters

- **row_key** (*bytes*) – The key for the current row.
- **table** (*Table*) – The table that owns the row.

clear()

Removes all currently accumulated mutations on the current row.

commit()

Makes a MutateRow API request.

If no mutations have been created in the row, no request is made.

Mutations are applied atomically and in order, meaning that earlier mutations can be masked / negated by later ones. Cells already present in the row are left unchanged unless explicitly changed by a mutation.

After committing the accumulated mutations, resets the local mutations to an empty list.

Raises `ValueError` if the number of mutations exceeds the `MAX_MUTATIONS`.

delete()

Deletes this row from the table.

Note: This method adds a mutation to the accumulated mutations on this row, but does not make an API request. To actually send an API request (with the mutations) to the Google Cloud Bigtable API, call `commit()`.

delete_cell(*column_family_id*, *column*, *time_range=None*)

Deletes cell in this row.

Note: This method adds a mutation to the accumulated mutations on this row, but does not make an API request. To actually send an API request (with the mutations) to the Google Cloud Bigtable API, call `commit()`.

Parameters

- **column_family_id**(*str*) – The column family that contains the column or columns with cells being deleted. Must be of the form `[_a-zA-Z0-9][_._a-zA-Z0-9]*`.
- **column**(*bytes*) – The column within the column family that will have a cell deleted.
- **time_range**(*TimestampRange*) – (Optional) The range of time within which cells should be deleted.

delete_cells(*column_family_id*, *columns*, *time_range=None*)

Deletes cells in this row.

Note: This method adds a mutation to the accumulated mutations on this row, but does not make an API request. To actually send an API request (with the mutations) to the Google Cloud Bigtable API, call `commit()`.

Parameters

- **column_family_id**(*str*) – The column family that contains the column or columns with cells being deleted. Must be of the form `[_a-zA-Z0-9][_._a-zA-Z0-9]*`.
- **columns**(*list of str / unicode, or object*) – The columns within the column family that will have cells deleted. If `ALL_COLUMNS` is used then the entire column family will be deleted from the row.
- **time_range**(*TimestampRange*) – (Optional) The range of time within which cells should be deleted.

row_key

Row key.

Return type `bytes`**Returns** The key for the current row.**set_cell**(*column_family_id*, *column*, *value*, *timestamp=None*)

Sets a value in this row.

The cell is determined by the `row_key` of this `DirectRow` and the `column`. The `column` must be in an existing `ColumnFamily` (as determined by `column_family_id`).

Note: This method adds a mutation to the accumulated mutations on this row, but does not make an API request. To actually send an API request (with the mutations) to the Google Cloud Bigtable API, call `commit()`.

Parameters

- **column_family_id** (*str*) – The column family that contains the column. Must be of the form `[_a-zA-Z0-9][_\.a-zA-Z0-9]*`.
- **column** (*bytes*) – The column within the column family where the cell is located.
- **value** (bytes or *int*) – The value to set in the cell. If an integer is used, will be interpreted as a 64-bit big-endian signed integer (8 bytes).
- **timestamp** (*datetime.datetime*) – (Optional) The timestamp of the operation.

table

Row table.

Return type table: *Table*

Returns table: The table that owns the row.

`google.cloud.bigtable.row.MAX_MUTATIONS = 100000`

The maximum number of mutations that a row can accumulate.

class `google.cloud.bigtable.row.Row` (*row_key*, *table*)

Bases: *object*

Base representation of a Google Cloud Bigtable Row.

This class has three subclasses corresponding to the three RPC methods for sending row mutations:

- *DirectRow* for `MutateRow`
- *ConditionalRow* for `CheckAndMutateRow`
- *AppendRow* for `ReadModifyWriteRow`

Parameters

- **row_key** (*bytes*) – The key for the current row.
- **table** (*Table*) – The table that owns the row.

row_key

Row key.

Return type *bytes*

Returns The key for the current row.

table

Row table.

Return type table: *Table*

Returns table: The table that owns the row.

6.10 Row Data

Container for Google Cloud Bigtable Cells and Streaming Row Contents.

class google.cloud.bigtable.row_data.**Cell** (*value*, *timestamp*, *labels=()*)

Bases: `object`

Representation of a Google Cloud Bigtable Cell.

Parameters

- **value** (*bytes*) – The value stored in the cell.
- **timestamp** (*datetime.datetime*) – The timestamp when the cell was stored.
- **labels** (*list*) – (Optional) List of strings. Labels applied to the cell.

classmethod **from_pb** (*cell_pb*)

Create a new cell from a Cell protobuf.

Parameters **cell_pb** (*_generated.data_pb2.Cell*) – The protobuf to convert.

Return type *Cell*

Returns The cell corresponding to the protobuf.

exception google.cloud.bigtable.row_data.**InvalidChunk**

Bases: `exceptions.RuntimeError`

Exception raised to to invalid chunk data from back-end.

exception google.cloud.bigtable.row_data.**InvalidReadRowsResponse**

Bases: `exceptions.RuntimeError`

Exception raised to to invalid response data from back-end.

class google.cloud.bigtable.row_data.**PartialCellData** (*row_key*, *family_name*, *qualifier*, *timestamp_micros*, *labels=()*, *value=""*)

Bases: `object`

Representation of partial cell in a Google Cloud Bigtable Table.

These are expected to be updated directly from a `_generated.bigtable_service_messages_pb2.ReadRowsResponse`

Parameters

- **row_key** (*bytes*) – The key for the row holding the (partial) cell.
- **family_name** (*str*) – The family name of the (partial) cell.
- **qualifier** (*bytes*) – The column qualifier of the (partial) cell.
- **timestamp_micros** (*int*) – The timestamp (in microseconds) of the (partial) cell.
- **labels** (*list of str*) – labels assigned to the (partial) cell
- **value** (*bytes*) – The (accumulated) value of the (partial) cell.

append_value (*value*)

Append bytes from a new chunk to value.

Parameters **value** (*bytes*) – bytes to append

class google.cloud.bigtable.row_data.**PartialRowData** (*row_key*)

Bases: `object`

Representation of partial row in a Google Cloud Bigtable Table.

These are expected to be updated directly from a `_generated.bigtable_service_messages_pb2.ReadRowsResponse`

Parameters **row_key** (*bytes*) – The key for the row holding the (partial) data.

cells

Property returning all the cells accumulated on this partial row.

Return type `dict`

Returns Dictionary of the `Cell` objects accumulated. This dictionary has two-levels of keys (first for column families and second for column names/qualifiers within a family). For a given column, a list of `Cell` objects is stored.

row_key

Getter for the current (partial) row's key.

Return type `bytes`

Returns The current (partial) row's key.

to_dict ()

Convert the cells to a dictionary.

This is intended to be used with HappyBase, so the column family and column qualifiers are combined (with `:`).

Return type `dict`

Returns Dictionary containing all the data in the cells of this row.

class google.cloud.bigtable.row_data.**PartialRowsData** (*response_iterator*)

Bases: `object`

Convenience wrapper for consuming a ReadRows streaming response.

Parameters **response_iterator** (*GrpcRendezvous*) – A streaming iterator returned from a ReadRows request.

cancel ()

Cancels the iterator, closing the stream.

consume_all (*max_loops=None*)

Consume the streamed responses until there are no more.

This simply calls `consume_next ()` until there are no more to consume.

Parameters **max_loops** (*int*) – (Optional) Maximum number of times to try to consume an additional ReadRowsResponse. You can use this to avoid long wait times.

consume_next ()

Consume the next ReadRowsResponse from the stream.

Parse the response and its chunks into a new/existing row in `_rows`. Rows are returned in order by row key.

rows

Property returning all rows accumulated from the stream.

Return type `dict`

Returns row_key -> *PartialRowData*.

state

State machine state.

Return type str

Returns name of state corresponding to current row / chunk processing.

6.11 Bigtable Row Filters

It is possible to use a *RowFilter* when adding mutations to a *ConditionalRow* and when reading row data with *read_row()* or *read_rows()*.

As laid out in the *RowFilter* definition, the following basic filters are provided:

- *SinkFilter*
- *PassAllFilter*
- *BlockAllFilter*
- *RowKeyRegexFilter*
- *RowSampleFilter*
- *FamilyNameRegexFilter*
- *ColumnQualifierRegexFilter*
- *TimestampRangeFilter*
- *ColumnRangeFilter*
- *ValueRegexFilter*
- *ValueRangeFilter*
- *CellsRowOffsetFilter*
- *CellsRowLimitFilter*
- *CellsColumnLimitFilter*
- *StripValueTransformerFilter*
- *ApplyLabelFilter*

In addition, these filters can be combined into composite filters with

- *RowFilterChain*
- *RowFilterUnion*
- *ConditionalRowFilter*

These rules can be nested arbitrarily, with a basic filter at the lowest level. For example:

```
# Filter in a specified column (matching any column family).
coll_filter = ColumnQualifierRegexFilter(b'columnbia')

# Create a filter to label results.
label1 = u'label-red'
label1_filter = ApplyLabelFilter(label1)
```

```
# Combine the filters to label all the cells in columnbia.
chain1 = RowFilterChain(filters=[col1_filter, label1_filter])

# Create a similar filter to label cells blue.
col2_filter = ColumnQualifierRegexFilter(b'columnseeya')
label2 = u'label-blue'
label2_filter = ApplyLabelFilter(label2)
chain2 = RowFilterChain(filters=[col2_filter, label2_filter])

# Bring our two labeled columns together.
row_filter = RowFilterUnion(filters=[chain1, chain2])
```

Filters for Google Cloud Bigtable Row classes.

class google.cloud.bigtable.row_filters.**ApplyLabelFilter** (*label*)

Bases: *google.cloud.bigtable.row_filters.RowFilter*

Filter to apply labels to cells.

Intended to be used as an intermediate filter on a pre-existing filtered result set. This way if two sets are combined, the label can tell where the cell(s) originated. This allows the client to determine which results were produced from which part of the filter.

Note: Due to a technical limitation of the backend, it is not currently possible to apply multiple labels to a cell.

Parameters *label* (*str*) – Label to apply to cells in the output row. Values must be at most 15 characters long, and match the pattern `[a-z0-9\-_]+\.`

to_pb()

Converts the row filter to a protobuf.

Return type `data_v2_pb2.RowFilter`

Returns The converted current object.

class google.cloud.bigtable.row_filters.**BlockAllFilter** (*flag*)

Bases: *google.cloud.bigtable.row_filters._BoolFilter*

Row filter that doesn't match any cells.

Parameters *flag* (*bool*) – Does not match any cells, regardless of input. Useful for temporarily disabling just part of a filter.

to_pb()

Converts the row filter to a protobuf.

Return type `data_v2_pb2.RowFilter`

Returns The converted current object.

class google.cloud.bigtable.row_filters.**CellsColumnLimitFilter** (*num_cells*)

Bases: *google.cloud.bigtable.row_filters._CellCountFilter*

Row filter to limit cells in a column.

Parameters *num_cells* (*int*) – Matches only the most recent N cells within each column. This filters a (family name, column) pair, based on timestamps of each cell.

to_pb()

Converts the row filter to a protobuf.

Return type `data_v2_pb2.RowFilter`

Returns The converted current object.

class `google.cloud.bigtable.row_filters.CellsRowLimitFilter` (*num_cells*)

Bases: `google.cloud.bigtable.row_filters._CellCountFilter`

Row filter to limit cells in a row.

Parameters **num_cells** (*int*) – Matches only the first N cells of the row.

to_pb()

Converts the row filter to a protobuf.

Return type `data_v2_pb2.RowFilter`

Returns The converted current object.

class `google.cloud.bigtable.row_filters.CellsRowOffsetFilter` (*num_cells*)

Bases: `google.cloud.bigtable.row_filters._CellCountFilter`

Row filter to skip cells in a row.

Parameters **num_cells** (*int*) – Skips the first N cells of the row.

to_pb()

Converts the row filter to a protobuf.

Return type `data_v2_pb2.RowFilter`

Returns The converted current object.

class `google.cloud.bigtable.row_filters.ColumnQualifierRegexFilter` (*regex*)

Bases: `google.cloud.bigtable.row_filters._RegexFilter`

Row filter for a column qualifier regular expression.

The regex must be valid RE2 patterns. See Google's [RE2 reference](#) for the accepted syntax.

Note: Special care need be used with the expression used. Since each of these properties can contain arbitrary bytes, the `\C` escape sequence must be used if a true wildcard is desired. The `.` character will not match the new line character `\n`, which may be present in a binary value.

Parameters **regex** (*bytes*) – A regular expression (RE2) to match cells from column that match this regex (irrespective of column family).

to_pb()

Converts the row filter to a protobuf.

Return type `data_v2_pb2.RowFilter`

Returns The converted current object.

class `google.cloud.bigtable.row_filters.ColumnRangeFilter` (*column_family_id*,
start_column=None,
end_column=None,
inclusive_start=None,
inclusive_end=None)

Bases: `google.cloud.bigtable.row_filters.RowFilter`

A row filter to restrict to a range of columns.

Both the start and end column can be included or excluded in the range. By default, we include them both, but this can be changed with optional flags.

Parameters

- **column_family_id** (*str*) – The column family that contains the columns. Must be of the form `[_a-zA-Z0-9][_._a-zA-Z0-9]*`.
- **start_column** (*bytes*) – The start of the range of columns. If no value is used, the backend applies no upper bound to the values.
- **end_column** (*bytes*) – The end of the range of columns. If no value is used, the backend applies no upper bound to the values.
- **inclusive_start** (*bool*) – Boolean indicating if the start column should be included in the range (or excluded). Defaults to `True` if `start_column` is passed and no `inclusive_start` was given.
- **inclusive_end** (*bool*) – Boolean indicating if the end column should be included in the range (or excluded). Defaults to `True` if `end_column` is passed and no `inclusive_end` was given.

Raises `ValueError` if `inclusive_start` is set but no `start_column` is given or if `inclusive_end` is set but no `end_column` is given

`to_pb()`

Converts the row filter to a protobuf.

First converts to a `data_v2_pb2.ColumnRange` and then uses it in the `column_range_filter` field.

Return type `data_v2_pb2.RowFilter`

Returns The converted current object.

```
class google.cloud.bigtable.row_filters.ConditionalRowFilter (base_filter,
                                                         true_filter=None,
                                                         false_filter=None)
```

Bases: `google.cloud.bigtable.row_filters.RowFilter`

Conditional row filter which exhibits ternary behavior.

Executes one of two filters based on another filter. If the `base_filter` returns any cells in the row, then `true_filter` is executed. If not, then `false_filter` is executed.

Note: The `base_filter` does not execute atomically with the true and false filters, which may lead to inconsistent or unexpected results.

Additionally, executing a `ConditionalRowFilter` has poor performance on the server, especially when `false_filter` is set.

Parameters

- **base_filter** (*RowFilter*) – The filter to condition on before executing the true/false filters.
- **true_filter** (*RowFilter*) – (Optional) The filter to execute if there are any cells matching `base_filter`. If not provided, no results will be returned in the true case.

- **false_filter** (*RowFilter*) – (Optional) The filter to execute if there are no cells matching *base_filter*. If not provided, no results will be returned in the false case.

to_pb()

Converts the row filter to a protobuf.

Return type `data_v2_pb2.RowFilter`

Returns The converted current object.

class `google.cloud.bigtable.row_filters.FamilyNameRegexFilter` (*regex*)

Bases: `google.cloud.bigtable.row_filters._RegexFilter`

Row filter for a family name regular expression.

The *regex* must be valid RE2 patterns. See Google’s [RE2 reference](#) for the accepted syntax.

Parameters **regex** (*str*) – A regular expression (RE2) to match cells from columns in a given column family. For technical reasons, the regex must not contain the ' : ' character, even if it is not being used as a literal.

to_pb()

Converts the row filter to a protobuf.

Return type `data_v2_pb2.RowFilter`

Returns The converted current object.

class `google.cloud.bigtable.row_filters.PassAllFilter` (*flag*)

Bases: `google.cloud.bigtable.row_filters._BoolFilter`

Row filter equivalent to not filtering at all.

Parameters **flag** (*bool*) – Matches all cells, regardless of input. Functionally equivalent to leaving *filter* unset, but included for completeness.

to_pb()

Converts the row filter to a protobuf.

Return type `data_v2_pb2.RowFilter`

Returns The converted current object.

class `google.cloud.bigtable.row_filters.RowFilter`

Bases: `object`

Basic filter to apply to cells in a row.

These values can be combined via [RowFilterChain](#), [RowFilterUnion](#) and [ConditionalRowFilter](#).

Note: This class is a do-nothing base class for all row filters.

class `google.cloud.bigtable.row_filters.RowFilterChain` (*filters=None*)

Bases: `google.cloud.bigtable.row_filters._FilterCombination`

Chain of row filters.

Sends rows through several filters in sequence. The filters are “chained” together to process a row. After the first filter is applied, the second is applied to the filtered output and so on for subsequent filters.

Parameters **filters** (*list*) – List of *RowFilter*

to_pb()

Converts the row filter to a protobuf.

Return type `data_v2_pb2.RowFilter`

Returns The converted current object.

class `google.cloud.bigtable.row_filters.RowFilterUnion` (*filters=None*)

Bases: `google.cloud.bigtable.row_filters._FilterCombination`

Union of row filters.

Sends rows through several filters simultaneously, then merges / interleaves all the filtered results together.

If multiple cells are produced with the same column and timestamp, they will all appear in the output row in an unspecified mutual order.

Parameters **filters** (*list*) – List of `RowFilter`

to_pb()

Converts the row filter to a protobuf.

Return type `data_v2_pb2.RowFilter`

Returns The converted current object.

class `google.cloud.bigtable.row_filters.RowKeyRegexFilter` (*regex*)

Bases: `google.cloud.bigtable.row_filters._RegexFilter`

Row filter for a row key regular expression.

The `regex` must be valid RE2 patterns. See Google's [RE2 reference](#) for the accepted syntax.

Note: Special care need be used with the expression used. Since each of these properties can contain arbitrary bytes, the `\C` escape sequence must be used if a true wildcard is desired. The `.` character will not match the new line character `\n`, which may be present in a binary value.

Parameters **regex** (*bytes*) – A regular expression (RE2) to match cells from rows with row keys that satisfy this regex. For a `CheckAndMutateRowRequest`, this filter is unnecessary since the row key is already specified.

to_pb()

Converts the row filter to a protobuf.

Return type `data_v2_pb2.RowFilter`

Returns The converted current object.

class `google.cloud.bigtable.row_filters.RowSampleFilter` (*sample*)

Bases: `google.cloud.bigtable.row_filters.RowFilter`

Matches all cells from a row with probability `p`.

Parameters **sample** (*float*) – The probability of matching a cell (must be in the interval `[0, 1]`).

to_pb()

Converts the row filter to a protobuf.

Return type `data_v2_pb2.RowFilter`

Returns The converted current object.

class google.cloud.bigtable.row_filters.**SinkFilter** (*flag*)

Bases: google.cloud.bigtable.row_filters._BoolFilter

Advanced row filter to skip parent filters.

Parameters **flag** (*bool*) – ADVANCED USE ONLY. Hook for introspection into the row filter. Outputs all cells directly to the output of the read rather than to any parent filter. Cannot be used within the `predicate_filter`, `true_filter`, or `false_filter` of a *ConditionalRowFilter*.

to_pb()

Converts the row filter to a protobuf.

Return type data_v2_pb2.RowFilter

Returns The converted current object.

class google.cloud.bigtable.row_filters.**StripValueTransformerFilter** (*flag*)

Bases: google.cloud.bigtable.row_filters._BoolFilter

Row filter that transforms cells into empty string (0 bytes).

Parameters **flag** (*bool*) – If `True`, replaces each cell's value with the empty string. As the name indicates, this is more useful as a transformer than a generic query / filter.

to_pb()

Converts the row filter to a protobuf.

Return type data_v2_pb2.RowFilter

Returns The converted current object.

class google.cloud.bigtable.row_filters.**TimestampRange** (*start=None, end=None*)

Bases: *object*

Range of time with inclusive lower and exclusive upper bounds.

Parameters

- **start** (*datetime.datetime*) – (Optional) The (inclusive) lower bound of the timestamp range. If omitted, defaults to Unix epoch.
- **end** (*datetime.datetime*) – (Optional) The (exclusive) upper bound of the timestamp range. If omitted, no upper bound is used.

to_pb()

Converts the *TimestampRange* to a protobuf.

Return type data_v2_pb2.TimestampRange

Returns The converted current object.

class google.cloud.bigtable.row_filters.**TimestampRangeFilter** (*range_*)

Bases: *google.cloud.bigtable.row_filters.RowFilter*

Row filter that limits cells to a range of time.

Parameters **range** (*TimestampRange*) – Range of time that cells should match against.

to_pb()

Converts the row filter to a protobuf.

First converts the `range_` on the current object to a protobuf and then uses it in the `timestamp_range_filter` field.

Return type data_v2_pb2.RowFilter

Returns The converted current object.

```
class google.cloud.bigtable.row_filters.ValueRangeFilter(start_value=None,
                                                         end_value=None,    in-
                                                         clusive_start=None,
                                                         inclusive_end=None)
```

Bases: `google.cloud.bigtable.row_filters.RowFilter`

A range of values to restrict to in a row filter.

Will only match cells that have values in this range.

Both the start and end value can be included or excluded in the range. By default, we include them both, but this can be changed with optional flags.

Parameters

- **start_value** (*bytes*) – The start of the range of values. If no value is used, the backend applies no lower bound to the values.
- **end_value** (*bytes*) – The end of the range of values. If no value is used, the backend applies no upper bound to the values.
- **inclusive_start** (*bool*) – Boolean indicating if the start value should be included in the range (or excluded). Defaults to `True` if `start_value` is passed and no `inclusive_start` was given.
- **inclusive_end** (*bool*) – Boolean indicating if the end value should be included in the range (or excluded). Defaults to `True` if `end_value` is passed and no `inclusive_end` was given.

Raises `ValueError` if `inclusive_start` is set but no `start_value` is given or if `inclusive_end` is set but no `end_value` is given

to_pb()

Converts the row filter to a protobuf.

First converts to a `data_v2_pb2.ValueRange` and then uses it to create a row filter protobuf.

Return type `data_v2_pb2.RowFilter`

Returns The converted current object.

```
class google.cloud.bigtable.row_filters.ValueRegexFilter(regex)
```

Bases: `google.cloud.bigtable.row_filters._RegexFilter`

Row filter for a value regular expression.

The `regex` must be valid RE2 patterns. See Google's [RE2 reference](#) for the accepted syntax.

Note: Special care need be used with the expression used. Since each of these properties can contain arbitrary bytes, the `\C` escape sequence must be used if a true wildcard is desired. The `.` character will not match the new line character `\n`, which may be present in a binary value.

Parameters **regex** (*bytes*) – A regular expression (RE2) to match cells with values that match this regex.

to_pb()

Converts the row filter to a protobuf.

Return type `data_v2_pb2.RowFilter`

Returns The converted current object.

6.12 Data API

After creating a *Table* and some column families, you are ready to store and retrieve data.

6.12.1 Cells vs. Columns vs. Column Families

- As explained in the *table overview*, tables can have many column families.
- As described below, a table can also have many rows which are specified by row keys.
- Within a row, data is stored in a cell. A cell simply has a value (as bytes) and a timestamp. The number of cells in each row can be different, depending on what was stored in each row.
- Each cell lies in a column (**not** a column family). A column is really just a more **specific** modifier within a column family. A column can be present in every column family, in only one or anywhere in between.
- Within a column family there can be many columns. For example, within the column family `foo` we could have columns `bar` and `baz`. These would typically be represented as `foo:bar` and `foo:baz`.

6.12.2 Modifying Data

Since data is stored in cells, which are stored in rows, we use the metaphor of a **row** in classes that are used to modify (write, update, delete) data in a *Table*.

Direct vs. Conditional vs. Append

There are three ways to modify data in a table, described by the *MutateRow*, *CheckAndMutateRow* and *ReadModifyWriteRow* API methods.

- The **direct** way is via *MutateRow* which involves simply adding, overwriting or deleting cells. The *DirectRow* class handles direct mutations.
- The **conditional** way is via *CheckAndMutateRow*. This method first checks if some filter is matched in a given row, then applies one of two sets of mutations, depending on if a match occurred or not. (These mutation sets are called the “true mutations” and “false mutations”.) The *ConditionalRow* class handles conditional mutations.
- The **append** way is via *ReadModifyWriteRow*. This simply appends (as bytes) or increments (as an integer) data in a presumed existing cell in a row. The *AppendRow* class handles append mutations.

Row Factory

A single factory can be used to create any of the three row types. To create a *DirectRow*:

```
row = table.row(row_key)
```

Unlike the previous string values we’ve used before, the row key must be bytes.

To create a *ConditionalRow*, first create a *RowFilter* and then

```
cond_row = table.row(row_key, filter_=filter_)
```

To create an *AppendRow*

```
append_row = table.row(row_key, append=True)
```

Building Up Mutations

In all three cases, a set of mutations (or two sets) are built up on a row before they are sent of in a batch via

```
row.commit()
```

Direct Mutations

Direct mutations can be added via one of four methods

- *set_cell()* allows a single value to be written to a column

```
row.set_cell(column_family_id, column, value,
             timestamp=timestamp)
```

If the `timestamp` is omitted, the current time on the Google Cloud Bigtable server will be used when the cell is stored.

The value can either be bytes or an integer, which will be converted to bytes as a signed 64-bit integer.

- *delete_cell()* deletes all cells (i.e. for all timestamps) in a given column

```
row.delete_cell(column_family_id, column)
```

Remember, this only happens in the `row` we are using.

If we only want to delete cells from a limited range of time, a `TimestampRange` can be used

```
row.delete_cell(column_family_id, column,
               time_range=time_range)
```

- *delete_cells()* does the same thing as *delete_cell()*, but accepts a list of columns in a column family rather than a single one.

```
row.delete_cells(column_family_id, [column1, column2],
                 time_range=time_range)
```

In addition, if we want to delete cells from every column in a column family, the special `ALL_COLUMNS` value can be used

```
row.delete_cells(column_family_id, row.ALL_COLUMNS,
                 time_range=time_range)
```

- *delete()* will delete the entire row

```
row.delete()
```

Conditional Mutations

Making **conditional** modifications is essentially identical to **direct** modifications: it uses the exact same methods to accumulate mutations.

However, each mutation added must specify a `state`: will the mutation be applied if the filter matches or if it fails to match.

For example:

```
cond_row.set_cell(column_family_id, column, value,
                  timestamp=timestamp, state=True)
```

will add to the set of true mutations.

Append Mutations

Append mutations can be added via one of two methods

- `append_cell_value()` appends a bytes value to an existing cell:

```
append_row.append_cell_value(column_family_id, column, bytes_value)
```

- `increment_cell_value()` increments an integer value in an existing cell:

```
append_row.increment_cell_value(column_family_id, column, int_value)
```

Since only bytes are stored in a cell, the cell value is decoded as a signed 64-bit integer before being incremented. (This happens on the Google Cloud Bigtable server, not in the library.)

Notice that no timestamp was specified. This is because **append** mutations operate on the latest value of the specified column.

If there are no cells in the specified column, then the empty string (bytes case) or zero (integer case) are the assumed values.

Starting Fresh

If accumulated mutations need to be dropped, use

```
row.clear()
```

6.12.3 Reading Data

Read Single Row from a Table

To make a `ReadRows` API request for a single row key, use `Table.read_row()`:

```
>>> row_data = table.read_row(row_key)
>>> row_data.cells
{
  u'fam1': {
    b'col1': [
      <google.cloud.bigtable.row_data.Cell at 0x7f80d150ef10>,
      <google.cloud.bigtable.row_data.Cell at 0x7f80d150ef10>,
    ],
    b'col2': [
      <google.cloud.bigtable.row_data.Cell at 0x7f80d150ef10>,
    ],
  },
}
```

```

    u'fam2': {
        b'col3': [
            <google.cloud.bigtable.row_data.Cell at 0x7f80d150ef10>,
            <google.cloud.bigtable.row_data.Cell at 0x7f80d150ef10>,
            <google.cloud.bigtable.row_data.Cell at 0x7f80d150ef10>,
        ],
    },
}
>>> cell = row_data.cells[u'fam1'][b'col1'][0]
>>> cell
<google.cloud.bigtable.row_data.Cell at 0x7f80d150ef10>
>>> cell.value
b'vall'
>>> cell.timestamp
datetime.datetime(2016, 2, 27, 3, 41, 18, 122823, tzinfo=<UTC>)

```

Rather than returning a *DirectRow* or similar class, this method returns a *PartialRowData* instance. This class is used for reading and parsing data rather than for modifying data (as *DirectRow* is).

A filter can also be applied to the results:

```
row_data = table.read_row(row_key, filter_=filter_val)
```

The allowable `filter_` values are the same as those used for a *ConditionalRow*. For more information, see the *Table.read_row()* documentation.

Stream Many Rows from a Table

To make a *ReadRows* API request for a stream of rows, use *Table.read_rows()*:

```
row_data = table.read_rows()
```

Using gRPC over HTTP/2, a continual stream of responses will be delivered. In particular

- *consume_next()* pulls the next result from the stream, parses it and stores it on the *PartialRowsData* instance
- *consume_all()* pulls results from the stream until there are no more
- *cancel()* closes the stream

See the *PartialRowsData* documentation for more information.

As with *Table.read_row()*, an optional `filter_` can be applied. In addition a `start_key` and/or `end_key` can be supplied for the stream, a `limit` can be set and a boolean `allow_row_interleaving` can be specified to allow faster streamed results at the potential cost of non-sequential reads.

See the *Table.read_rows()* documentation for more information on the optional arguments.

Sample Keys in a Table

Make a *SampleRowKeys* API request with *Table.sample_row_keys()*:

```
keys_iterator = table.sample_row_keys()
```

The returned row keys will delimit contiguous sections of the table of approximately equal size, which can be used to break up the data for distributed tasks like mapreduces.

As with `Table.read_rows()`, the returned `keys_iterator` is connected to a cancellable HTTP/2 stream.

The next key in the result can be accessed via

```
next_key = keys_iterator.next()
```

or all keys can be iterated over via

```
for curr_key in keys_iterator:
    do_something(curr_key)
```

Just as with reading, the stream can be canceled:

```
keys_iterator.cancel()
```

API requests are sent to the [Google Cloud Bigtable API](#) via RPC over HTTP/2. In order to support this, we'll rely on [gRPC](#). We are working with the gRPC team to rapidly make the install story more user-friendly.

Get started by learning about the *Client* on the *Base for Everything* page.

In the hierarchy of API concepts

- a *Client* owns an *Instance*
- an *Instance* owns a *Table*
- a *Table* owns a *ColumnFamily*
- a *Table* owns a *Row* (and all the cells in the row)

7.1 Datastore Client

Convenience wrapper for invoking APIs/factories w/ a project.

```
class google.cloud.datastore.client.Client (project=None, namespace=None, credentials=None, _http=None, _use_grpc=None)
```

Bases: *google.cloud.client.ClientWithProject*

Convenience wrapper for invoking APIs/factories w/ a project.

```
>>> from google.cloud import datastore
>>> client = datastore.Client()
```

Parameters

- **project** (*str*) – (optional) The project to pass to proxied API methods.
- **namespace** (*str*) – (optional) namespace to pass to proxied API methods.
- **credentials** (*Credentials*) – (Optional) The OAuth2 Credentials to use for this client. If not passed (and if no `_http` object is passed), falls back to the default inferred from the environment.
- **_http** (*Session*) – (Optional) HTTP object to make requests. Can be any object that defines `request()` with the same interface as `requests.Session.request()`. If not passed, an `_http` object is created that is bound to the `credentials` for the current object. This parameter should be considered private, and could change in the future.
- **_use_grpc** (*bool*) – (Optional) Explicitly specifies whether to use the gRPC transport (via GAX) or HTTP. If unset, falls back to the `GOOGLE_CLOUD_DISABLE_GRPC` environment variable. This parameter should be considered private, and could change in the future.

```
SCOPE = ('https://www.googleapis.com/auth/datastore',)
```

The scopes required for authenticating as a Cloud Datastore consumer.

allocate_ids (*incomplete_key*, *num_ids*)

Allocate a list of IDs from a partial key.

Parameters

- **incomplete_key** (*google.cloud.datastore.key.Key*) – Partial key to use as base for allocated IDs.
- **num_ids** (*int*) – The number of IDs to allocate.

Return type list of *google.cloud.datastore.key.Key*

Returns The (complete) keys allocated with *incomplete_key* as root.

Raises *ValueError* if *incomplete_key* is not a partial key.

batch ()

Proxy to *google.cloud.datastore.batch.Batch*.

current_batch

Currently-active batch.

Return type *google.cloud.datastore.batch.Batch*, or an object implementing its API, or *NoneType* (if no batch is active).

Returns The batch/transaction at the top of the batch stack.

current_transaction

Currently-active transaction.

Return type *google.cloud.datastore.transaction.Transaction*, or an object implementing its API, or *NoneType* (if no transaction is active).

Returns The transaction at the top of the batch stack.

delete (*key*)

Delete the key in the Cloud Datastore.

Note: This is just a thin wrapper over *delete_multi()*. The backend API does not make a distinction between a single key or multiple keys in a commit request.

Parameters **key** (*google.cloud.datastore.key.Key*) – The key to be deleted from the datastore.

delete_multi (*keys*)

Delete keys from the Cloud Datastore.

Parameters **keys** (list of *google.cloud.datastore.key.Key*) – The keys to be deleted from the Datastore.

get (*key*, *missing=None*, *deferred=None*, *transaction=None*)

Retrieve an entity from a single key (if it exists).

Note: This is just a thin wrapper over *get_multi()*. The backend API does not make a distinction between a single key or multiple keys in a lookup request.

Parameters

- **key** (*google.cloud.datastore.key.Key*) – The key to be retrieved from the datastore.
- **missing** (*list*) – (Optional) If a list is passed, the key-only entities returned by the backend as “missing” will be copied into it.
- **deferred** (*list*) – (Optional) If a list is passed, the keys returned by the backend as “deferred” will be copied into it.
- **transaction** (*Transaction*) – (Optional) Transaction to use for read consistency. If not passed, uses current transaction, if set.

Return type *google.cloud.datastore.entity.Entity* or *NoneType*

Returns The requested entity if it exists.

get_multi (*keys, missing=None, deferred=None, transaction=None*)

Retrieve entities, along with their attributes.

Parameters

- **keys** (list of *google.cloud.datastore.key.Key*) – The keys to be retrieved from the datastore.
- **missing** (*list*) – (Optional) If a list is passed, the key-only entities returned by the backend as “missing” will be copied into it. If the list is not empty, an error will occur.
- **deferred** (*list*) – (Optional) If a list is passed, the keys returned by the backend as “deferred” will be copied into it. If the list is not empty, an error will occur.
- **transaction** (*Transaction*) – (Optional) Transaction to use for read consistency. If not passed, uses current transaction, if set.

Return type list of *google.cloud.datastore.entity.Entity*

Returns The requested entities.

Raises *ValueError* if one or more of keys has a project which does not match our project.

key (**path_args, **kwargs*)

Proxy to *google.cloud.datastore.key.Key*.

Passes our project.

put (*entity*)

Save an entity in the Cloud Datastore.

Note: This is just a thin wrapper over *put_multi()*. The backend API does not make a distinction between a single entity or multiple entities in a commit request.

Parameters **entity** (*google.cloud.datastore.entity.Entity*) – The entity to be saved to the datastore.

put_multi (*entities*)

Save entities in the Cloud Datastore.

Parameters **entities** (list of *google.cloud.datastore.entity.Entity*) – The entities to be saved to the datastore.

Raises *ValueError* if *entities* is a single entity.

query (**kwargs)

Proxy to `google.cloud.datastore.query.Query`.

Passes our project.

Using query to search a datastore:

```
>>> query = client.query(kind='MyKind')
>>> query.add_filter('property', '=', 'val')
```

Using the query iterator

```
>>> query_iter = query.fetch()
>>> for entity in query_iter:
...     do_something(entity)
```

or manually page through results

```
>>> query_iter = query.fetch(start_cursor=cursor)
>>> pages = query_iter.pages
>>>
>>> first_page = next(pages)
>>> first_page_entities = list(first_page)
>>> query_iter.next_page_token
b'...'
```

Parameters **kwargs** (*dict*) – Parameters for initializing and instance of `Query`.

Return type `Query`

Returns A query object.

transaction ()

Proxy to `google.cloud.datastore.transaction.Transaction`.

7.2 Entities

Class for representing a single entity in the Cloud Datastore.

class `google.cloud.datastore.entity.Entity` (*key=None, exclude_from_indexes=()*)

Bases: `dict`

Entities are akin to rows in a relational database

An entity storing the actual instance of data.

Each entity is officially represented with a `Key`, however it is possible that you might create an entity with only a partial key (that is, a key with a kind, and possibly a parent, but without an ID). In such a case, the datastore service will automatically assign an ID to the partial key.

Entities in this API act like dictionaries with extras built in that allow you to delete or persist the data stored on the entity.

Entities are mutable and act like a subclass of a dictionary. This means you could take an existing entity and change the key to duplicate the object.

Use `get ()` to retrieve an existing entity:

```
>>> client.get(key)
<Entity('EntityKind', 1234) {'property': 'value'}>
```

You can the set values on the entity just like you would on any other dictionary.

```
>>> entity['age'] = 20
>>> entity['name'] = 'JJ'
```

However, not all types are allowed as a value for a Google Cloud Datastore entity. The following basic types are supported by the API:

- `datetime.datetime`
- `Key`
- `bool`
- `float`
- `int` (as well as `long` in Python 2)
- `unicode` (called `str` in Python 3)
- `bytes` (called `str` in Python 2)
- `GeoPoint`
- `None`

In addition, three container types are supported:

- `list`
- `Entity`
- `dict` (will just be treated like an `Entity` without a `key` or `exclude_from_indexes`)

Each entry in a list must be one of the value types (basic or container) and each value in an `Entity` must as well. In this case an `Entity` as a **container** acts as a `dict`, but also has the special annotations of `key` and `exclude_from_indexes`.

And you can treat an entity like a regular Python dictionary:

```
>>> sorted(entity.keys())
['age', 'name']
>>> sorted(entity.items())
[('age', 20), ('name', 'JJ')]
```

Note: When saving an entity to the backend, values which are “text” (`unicode` in Python2, `str` in Python3) will be saved using the ‘text_value’ field, after being encoded to UTF-8. When retrieved from the back-end, such values will be decoded to “text” again. Values which are “bytes” (`str` in Python2, `bytes` in Python3), will be saved using the ‘blob_value’ field, without any decoding / encoding step.

Parameters

- **key** (`google.cloud.datastore.key.Key`) – Optional key to be set on entity.
- **exclude_from_indexes** (*tuple of string*) – Names of fields whose values are not to be indexed for this entity.

exclude_from_indexes = None

Names of fields which are *not* to be indexed for this entity.

kind

Get the kind of the current entity.

Note: This relies entirely on the `google.cloud.datastore.key.Key` set on the entity. That means that we're not storing the kind of the entity at all, just the properties and a pointer to a Key which knows its Kind.

7.3 Keys

Create / interact with Google Cloud Datastore keys.

class google.cloud.datastore.key.**Key**(*path_args, **kwargs)
Bases: `object`

An immutable representation of a datastore Key.

To create a basic key directly:

```
>>> Key('EntityKind', 1234, project=project)
<Key('EntityKind', 1234), project=...>
>>> Key('EntityKind', 'foo', project=project)
<Key('EntityKind', 'foo'), project=...>
```

Though typical usage comes via the `key()` factory:

```
>>> client.key('EntityKind', 1234)
<Key('EntityKind', 1234), project=...>
>>> client.key('EntityKind', 'foo')
<Key('EntityKind', 'foo'), project=...>
```

To create a key with a parent:

```
>>> client.key('Parent', 'foo', 'Child', 1234)
<Key('Parent', 'foo', 'Child', 1234), project=...>
>>> client.key('Child', 1234, parent=parent_key)
<Key('Parent', 'foo', 'Child', 1234), project=...>
```

To create a partial key:

```
>>> client.key('Parent', 'foo', 'Child')
<Key('Parent', 'foo', 'Child'), project=...>
```

Parameters

- **path_args** (*tuple of string and integer*) – May represent a partial (odd length) or full (even length) key path.
- **kwargs** (*dict*) – Keyword arguments to be passed in.

Accepted keyword arguments are

- **namespace** (string): A namespace identifier for the key.

- `project` (string): The project associated with the key.
- `parent` (*Key*): The parent of the key.

The `project` argument is required unless it has been set implicitly.

completed_key (*id_or_name*)

Creates new key from existing partial key by adding final ID/name.

Parameters `id_or_name` (*str* or *integer*) – ID or name to be added to the key.

Return type *google.cloud.datastore.key.Key*

Returns A new *Key* instance with the same data as the current one and an extra ID or name added.

Raises *ValueError* if the current key is not partial or if `id_or_name` is not a string or integer.

flat_path

Getter for the key path as a tuple.

Return type tuple of string and integer

Returns The tuple of elements in the path.

classmethod from_legacy_urlsafe (*urlsafe*)

Convert urlsafe string to *Key*.

This is intended to work with the “legacy” representation of a datastore “Key” used within Google App Engine (a so-called “Reference”). This assumes that `urlsafe` was created within an App Engine app via something like `ndb.Key(...).urlsafe()`.

Parameters `urlsafe` (*bytes* or *unicode*) – The base64 encoded (ASCII) string corresponding to a datastore “Key” / “Reference”.

Return type *Key*.

Returns The key corresponding to `urlsafe`.

id

ID getter. Based on the last element of path.

Return type *int*

Returns The (integer) ID of the key.

id_or_name

Getter. Based on the last element of path.

Return type *int* (if `id`) or *string* (if `name`)

Returns The last element of the key’s path if it is either an `id` or a `name`.

is_partial

Boolean indicating if the key has an ID (or name).

Return type *bool*

Returns `True` if the last element of the key’s path does not have an `id` or a `name`.

kind

Kind getter. Based on the last element of path.

Return type *str*

Returns The kind of the current key.

name

Name getter. Based on the last element of path.

Return type `str`

Returns The (string) name of the key.

namespace

Namespace getter.

Return type `str`

Returns The namespace of the current key.

parent

The parent of the current key.

Return type `google.cloud.datastore.key.Key` or `NoneType`

Returns A new `Key` instance, whose path consists of all but the last element of current path. If the current key has only one path element, returns `None`.

path

Path getter.

Returns a copy so that the key remains immutable.

Return type `list` of `dict`

Returns The (key) path of the current key.

project

Project getter.

Return type `str`

Returns The key's project.

to_legacy_urlsafe()

Convert to a base64 encode urlsafe string for App Engine.

This is intended to work with the “legacy” representation of a datastore “Key” used within Google App Engine (a so-called “Reference”). The returned string can be used as the `urlsafe` argument to `ndb.Key(urlsafe=...)`. The base64 encoded values will have padding removed.

Note: The string returned by `to_legacy_urlsafe` is equivalent, but not identical, to the string returned by `ndb`.

Return type `bytes`

Returns A bytestring containing the key encoded as URL-safe base64.

to_protobuf()

Return a protobuf corresponding to the key.

Return type `entity_pb2.Key`

Returns The protobuf representing the key.

7.4 Queries

Create / interact with Google Cloud Datastore queries.

```
class google.cloud.datastore.query.Iterator(query, client, limit=None, offset=None,
                                             start_cursor=None, end_cursor=None)
```

Bases: `google.api.core.page_iterator.Iterator`

Represent the state of a given execution of a Query.

Parameters

- **query** (*Query*) – Query object holding permanent configuration (i.e. things that don't change on with each page in a results set).
- **client** (*Client*) – The client used to make a request.
- **limit** (*int*) – (Optional) Limit the number of results returned.
- **offset** (*int*) – (Optional) Offset used to begin a query.
- **start_cursor** (*bytes*) – (Optional) Cursor to begin paging through query results.
- **end_cursor** (*bytes*) – (Optional) Cursor to end paging through query results.

```
class google.cloud.datastore.query.Query(client, kind=None, project=None, namespace=None,
                                           ancestor=None, filters=(), projection=(), order=(), distinct_on=())
```

Bases: `object`

A Query against the Cloud Datastore.

This class serves as an abstraction for creating a query over data stored in the Cloud Datastore.

Parameters

- **client** (*google.cloud.datastore.client.Client*) – The client used to connect to Datastore.
- **kind** (*str*) – The kind to query.
- **project** (*str*) – (Optional) The project associated with the query. If not passed, uses the client's value.
- **namespace** (*str*) – (Optional) The namespace to which to restrict results. If not passed, uses the client's value.
- **ancestor** (*Key*) – (Optional) key of the ancestor to which this query's results are restricted.
- **filters** (*tuple[str, str, str]*) – Property filters applied by this query. The sequence is (property_name, operator, value).
- **projection** (*sequence of string*) – fields returned as part of query results.
- **order** (*sequence of string*) – field names used to order query results. Prepend – to a field name to sort it in descending order.
- **distinct_on** (*sequence of string*) – field names used to group query results.

Raises `ValueError` if `project` is not passed and no implicit default is set.

```
OPERATORS = {'>': 3, '<=': 2, '=': 5, '>=': 4, '<': 1}
```

Mapping of operator strings and their protobuf equivalents.

add_filter (*property_name*, *operator*, *value*)

Filter the query based on a property name, operator and a value.

Expressions take the form of:

```
.add_filter('<property>', '<operator>', <value>)
```

where property is a property stored on the entity in the datastore and operator is one of OPERATORS (ie, =, <, <=, >, >=):

```
>>> from google.cloud import datastore
>>> client = datastore.Client()
>>> query = client.query(kind='Person')
>>> query.add_filter('name', '=', 'James')
>>> query.add_filter('age', '>', 50)
```

Parameters

- **property_name** (*str*) – A property name.
- **operator** (*str*) – One of =, <, <=, >, >=.
- **value** (*int*, *str*, *bool*, *float*, *NoneType*, *datetime.datetime*, *google.cloud.datastore.key.Key*) – The value to filter on.

Raises `ValueError` if operation is not one of the specified values, or if a filter names `'__key__'` but passes an invalid value (a key is required).

ancestor

The ancestor key for the query.

Return type *Key* or *None*

Returns The ancestor for the query.

distinct_on

Names of fields used to group query results.

Return type sequence of string

Returns The “distinct on” fields set on the query.

fetch (*limit=None*, *offset=0*, *start_cursor=None*, *end_cursor=None*, *client=None*)

Execute the Query; return an iterator for the matching entities.

For example:

```
>>> from google.cloud import datastore
>>> client = datastore.Client()
>>> query = client.query(kind='Person')
>>> query.add_filter('name', '=', 'Sally')
>>> list(query.fetch())
[<Entity object>, <Entity object>, ...]
>>> list(query.fetch(1))
[<Entity object>]
```

Parameters

- **limit** (*int*) – (Optional) limit passed through to the iterator.
- **offset** (*int*) – (Optional) offset passed through to the iterator.

- **start_cursor** (*bytes*) – (Optional) cursor passed through to the iterator.
- **end_cursor** (*bytes*) – (Optional) cursor passed through to the iterator.
- **client** (*google.cloud.datastore.client.Client*) – client used to connect to datastore. If not supplied, uses the query's value.

Return type *Iterator*

Returns The iterator for the query.

filters

Filters set on the query.

Return type *tuple[str, str, str]*

Returns The filters set on the query. The sequence is (property_name, operator, value).

key_filter (*key*, *operator*='=')

Filter on a key.

Parameters

- **key** (*google.cloud.datastore.key.Key*) – The key to filter on.
- **operator** (*str*) – (Optional) One of =, <, <=, >, >=. Defaults to =.

keys_only ()

Set the projection to include only keys.

kind

Get the Kind of the Query.

Return type *str*

Returns The kind for the query.

namespace

This query's namespace

Return type *str* or *None*

Returns the namespace assigned to this query

order

Names of fields used to sort query results.

Return type sequence of string

Returns The order(s) set on the query.

project

Get the project for this Query.

Return type *str*

Returns The project for the query.

projection

Fields names returned by the query.

Return type sequence of string

Returns Names of fields in query results.

7.5 Transactions

Create / interact with Google Cloud Datastore transactions.

class google.cloud.datastore.transaction.Transaction(*client*)

Bases: *google.cloud.datastore.batch.Batch*

An abstraction representing datastore Transactions.

Transactions can be used to build up a bulk mutation and ensure all or none succeed (transactionally).

For example, the following snippet of code will put the two save operations (either insert or upsert) into the same mutation, and execute those within a transaction:

```
>>> with client.transaction():
...     client.put_multi([entity1, entity2])
```

Because it derives from *Batch*, *Transaction* also provides *put()* and *delete()* methods:

```
>>> with client.transaction() as xact:
...     xact.put(entity1)
...     xact.delete(entity2.key)
```

By default, the transaction is rolled back if the transaction block exits with an error:

```
>>> with client.transaction():
...     do_some_work()
...     raise SomeException # rolls back
Traceback (most recent call last):
...
SomeException
```

If the transaction block exits without an exception, it will commit by default.

Warning:

Inside a transaction, automatically assigned IDs for entities will not be available at save time! That means, if you try:

```
>>> with client.transaction():
...     entity = Entity(key=client.key('Thing'))
...     client.put(entity)
```

entity won't have a complete key until the transaction is committed.

Once you exit the transaction (or call *commit()*), the automatically generated ID will be assigned to the entity:

```
>>> with client.transaction():
...     entity = Entity(key=client.key('Thing'))
...     client.put(entity)
...     print(entity.key.is_partial) # There is no ID on this key.
...
True
>>> print(entity.key.is_partial) # There is an ID.
False
```

If you don't want to use the context manager you can initialize a transaction manually:

```

>>> transaction = client.transaction()
>>> transaction.begin()
>>>
>>> entity = Entity(key=client.key('Thing'))
>>> transaction.put(entity)
>>>
>>> transaction.commit()

```

Parameters `client` (*google.cloud.datastore.client.Client*) – the client used to connect to datastore.

`begin()`

Begins a transaction.

This method is called automatically when entering a `with` statement, however it can be called explicitly if you don't want to use a context manager.

Raises `ValueError` if the transaction has already begun.

`commit()`

Commits the transaction.

This is called automatically upon exiting a `with` statement, however it can be called explicitly if you don't want to use a context manager.

This method has necessary side-effects:

- Sets the current transaction's ID to `None`.

`current()`

Return the topmost transaction.

Note: If the topmost element on the stack is not a transaction, returns `None`.

Return type *google.cloud.datastore.transaction.Transaction* or `None`

Returns The current transaction (if any are active).

`delete(key)`

Remember a key to be deleted during `commit()`.

Parameters `key` (*google.cloud.datastore.key.Key*) – the key to be deleted.

Raises `ValueError` if the batch is not in progress, if key is not complete, or if the key's project does not match ours.

`id`

Getter for the transaction ID.

Return type `str`

Returns The ID of the current transaction.

`mutations`

Getter for the changes accumulated by this batch.

Every batch is committed with a single commit request containing all the work to be done as mutations. Inside a batch, calling `put()` with an entity, or `delete()` with a key, builds up the request by adding a new mutation. This getter returns the protobuf that has been built-up so far.

Return type `iterable`

Returns The list of `datastore_pb2.Mutation` protobufs to be sent in the commit request.

namespace

Getter for namespace in which the batch will run.

Return type `str`

Returns The namespace in which the batch will run.

project

Getter for project in which the batch will run.

Return type `str`

Returns The project in which the batch will run.

put (*entity*)

Remember an entity's state to be saved during `commit()`.

Note: Any existing properties for the entity will be replaced by those currently set on this instance. Already-stored properties which do not correspond to keys set on this instance will be removed from the datastore.

Note: Property values which are “text” (‘unicode’ in Python2, ‘str’ in Python3) map to ‘string_value’ in the datastore; values which are “bytes” (‘str’ in Python2, ‘bytes’ in Python3) map to ‘blob_value’.

When an entity has a partial key, calling `commit()` sends it as an `insert` mutation and the key is completed. On return, the key for the `entity` passed in is updated to match the key ID assigned by the server.

Parameters `entity` (`google.cloud.datastore.entity.Entity`) – the entity to be saved.

Raises `ValueError` if the batch is not in progress, if entity has no key assigned, or if the key's project does not match ours.

rollback ()

Rolls back the current transaction.

This method has necessary side-effects:

- Sets the current transaction's ID to None.

7.6 Batches

Create / interact with a batch of updates / deletes.

Batches provide the ability to execute multiple operations in a single request to the Cloud Datastore API.

See https://cloud.google.com/datastore/docs/concepts/entities#batch_operations

class `google.cloud.datastore.batch.Batch` (*client*)

Bases: `object`

An abstraction representing a collected group of updates / deletes.

Used to build up a bulk mutation.

For example, the following snippet of code will put the two `save` operations and the `delete` operation into the same mutation, and send them to the server in a single API request:

```
>>> from google.cloud import datastore
>>> client = datastore.Client()
>>> batch = client.batch()
>>> batch.put(entity1)
>>> batch.put(entity2)
>>> batch.delete(key3)
>>> batch.commit()
```

You can also use a batch as a context manager, in which case `commit()` will be called automatically if its block exits without raising an exception:

```
>>> with batch:
...     batch.put(entity1)
...     batch.put(entity2)
...     batch.delete(key3)
```

By default, no updates will be sent if the block exits with an error:

```
>>> with batch:
...     do_some_work(batch)
...     raise Exception() # rolls back
```

Parameters `client` (*google.cloud.datastore.client.Client*) – The client used to connect to datastore.

begin()

Begins a batch.

This method is called automatically when entering a `with` statement, however it can be called explicitly if you don't want to use a context manager.

Overridden by *google.cloud.datastore.transaction.Transaction*.

Raises `ValueError` if the batch has already begun.

commit()

Commits the batch.

This is called automatically upon exiting a `with` statement, however it can be called explicitly if you don't want to use a context manager.

Raises `ValueError` if the batch is not in progress.

current()

Return the topmost batch / transaction, or `None`.

delete(key)

Remember a key to be deleted during `commit()`.

Parameters `key` (*google.cloud.datastore.key.Key*) – the key to be deleted.

Raises `ValueError` if the batch is not in progress, if `key` is not complete, or if the key's project does not match ours.

mutations

Getter for the changes accumulated by this batch.

Every batch is committed with a single commit request containing all the work to be done as mutations. Inside a batch, calling `put()` with an entity, or `delete()` with a key, builds up the request by adding a new mutation. This getter returns the protobuf that has been built-up so far.

Return type `iterable`

Returns The list of `datastore_pb2.Mutation` protobufs to be sent in the commit request.

namespace

Getter for namespace in which the batch will run.

Return type `str`

Returns The namespace in which the batch will run.

project

Getter for project in which the batch will run.

Return type `str`

Returns The project in which the batch will run.

put(entity)

Remember an entity's state to be saved during `commit()`.

Note: Any existing properties for the entity will be replaced by those currently set on this instance. Already-stored properties which do not correspond to keys set on this instance will be removed from the datastore.

Note: Property values which are "text" ('unicode' in Python2, 'str' in Python3) map to 'string_value' in the datastore; values which are "bytes" ('str' in Python2, 'bytes' in Python3) map to 'blob_value'.

When an entity has a partial key, calling `commit()` sends it as an `insert` mutation and the key is completed. On return, the key for the `entity` passed in is updated to match the key ID assigned by the server.

Parameters `entity` (`google.cloud.datastore.entity.Entity`) – the entity to be saved.

Raises `ValueError` if the batch is not in progress, if entity has no key assigned, or if the key's project does not match ours.

rollback()

Rolls back the current batch.

Marks the batch as aborted (can't be used again).

Overridden by `google.cloud.datastore.transaction.Transaction`.

Raises `ValueError` if the batch is not in progress.

7.7 Helpers

Helper functions for dealing with Cloud Datastore's Protobuf API.

The non-private functions are part of the API.

class `google.cloud.datastore.helpers.GeoPoint` (*latitude, longitude*)

Bases: `object`

Simple container for a geo point value.

Parameters

- **latitude** (*float*) – Latitude of a point.
- **longitude** (*float*) – Longitude of a point.

to_protobuf ()

Convert the current object to protobuf.

Return type `google.type.latlng_pb2.LatLng`.

Returns The current point as a protobuf.

`google.cloud.datastore.helpers.entity_from_protobuf` (*pb*)

Factory method for creating an entity based on a protobuf.

The protobuf should be one returned from the Cloud Datastore Protobuf API.

Parameters **pb** (`entity_pb2.Entity`) – The Protobuf representing the entity.

Return type `google.cloud.datastore.entity.Entity`

Returns The entity derived from the protobuf.

`google.cloud.datastore.helpers.entity_to_protobuf` (*entity*)

Converts an entity into a protobuf.

Parameters **entity** (`google.cloud.datastore.entity.Entity`) – The entity to be turned into a protobuf.

Return type `entity_pb2.Entity`

Returns The protobuf representing the entity.

`google.cloud.datastore.helpers.key_from_protobuf` (*pb*)

Factory method for creating a key based on a protobuf.

The protobuf should be one returned from the Cloud Datastore Protobuf API.

Parameters **pb** (`entity_pb2.Key`) – The Protobuf representing the key.

Return type `google.cloud.datastore.key.Key`

Returns a new *Key* instance

7.8 Modules

Shortcut methods for getting set up with Google Cloud Datastore.

You'll typically use these to get started with the API:

```
>>> from google.cloud import datastore
>>>
>>> client = datastore.Client()
>>> key = client.key('EntityKind', 1234)
>>> key
<Key('EntityKind', 1234), project=...>
>>> entity = datastore.Entity(key)
>>> entity['answer'] = 42
```

```
>>> entity
<Entity('EntityKind', 1234) {'answer': 42}>
>>> query = client.query(kind='EntityKind')
```

The main concepts with this API are:

- *Client* which represents a project (string) and namespace (string) bundled with a connection and has convenience methods for constructing objects with that project / namespace.
- *Entity* which represents a single entity in the datastore (akin to a row in relational database world).
- *Key* which represents a pointer to a particular entity in the datastore (akin to a unique identifier in relational database world).
- *Query* which represents a lookup or search over the rows in the datastore.
- *Transaction* which represents an all-or-none transaction and enables consistency when race conditions may occur.

class google.cloud.datastore.**Batch** (*client*)

Bases: *object*

An abstraction representing a collected group of updates / deletes.

Used to build up a bulk mutation.

For example, the following snippet of code will put the two `save` operations and the `delete` operation into the same mutation, and send them to the server in a single API request:

```
>>> from google.cloud import datastore
>>> client = datastore.Client()
>>> batch = client.batch()
>>> batch.put(entity1)
>>> batch.put(entity2)
>>> batch.delete(key3)
>>> batch.commit()
```

You can also use a batch as a context manager, in which case `commit()` will be called automatically if its block exits without raising an exception:

```
>>> with batch:
...     batch.put(entity1)
...     batch.put(entity2)
...     batch.delete(key3)
```

By default, no updates will be sent if the block exits with an error:

```
>>> with batch:
...     do_some_work(batch)
...     raise Exception() # rolls back
```

Parameters **client** (*google.cloud.datastore.client.Client*) – The client used to connect to datastore.

begin()

Begins a batch.

This method is called automatically when entering a `with` statement, however it can be called explicitly if you don't want to use a context manager.

Overridden by `google.cloud.datastore.transaction.Transaction`.

Raises `ValueError` if the batch has already begun.

commit()

Commits the batch.

This is called automatically upon exiting a `with` statement, however it can be called explicitly if you don't want to use a context manager.

Raises `ValueError` if the batch is not in progress.

current()

Return the topmost batch / transaction, or `None`.

delete(key)

Remember a key to be deleted during `commit()`.

Parameters `key` (`google.cloud.datastore.key.Key`) – the key to be deleted.

Raises `ValueError` if the batch is not in progress, if `key` is not complete, or if the key's project does not match ours.

mutations

Getter for the changes accumulated by this batch.

Every batch is committed with a single commit request containing all the work to be done as mutations. Inside a batch, calling `put()` with an entity, or `delete()` with a key, builds up the request by adding a new mutation. This getter returns the protobuf that has been built-up so far.

Return type `iterable`

Returns The list of `datastore_pb2.Mutation` protobufs to be sent in the commit request.

namespace

Getter for namespace in which the batch will run.

Return type `str`

Returns The namespace in which the batch will run.

project

Getter for project in which the batch will run.

Return type `str`

Returns The project in which the batch will run.

put(entity)

Remember an entity's state to be saved during `commit()`.

Note: Any existing properties for the entity will be replaced by those currently set on this instance. Already-stored properties which do not correspond to keys set on this instance will be removed from the datastore.

Note: Property values which are "text" ('unicode' in Python2, 'str' in Python3) map to 'string_value' in the datastore; values which are "bytes" ('str' in Python2, 'bytes' in Python3) map to 'blob_value'.

When an entity has a partial key, calling `commit()` sends it as an `insert` mutation and the key is completed. On return, the key for the `entity` passed in is updated to match the key ID assigned by the server.

Parameters `entity` (`google.cloud.datastore.entity.Entity`) – the entity to be saved.

Raises `ValueError` if the batch is not in progress, if entity has no key assigned, or if the key's project does not match ours.

rollback()

Rolls back the current batch.

Marks the batch as aborted (can't be used again).

Overridden by `google.cloud.datastore.transaction.Transaction`.

Raises `ValueError` if the batch is not in progress.

class `google.cloud.datastore.Client` (`project=None`, `namespace=None`, `credentials=None`, `_http=None`, `_use_grpc=None`)

Bases: `google.cloud.client.ClientWithProject`

Convenience wrapper for invoking APIs/factories w/ a project.

```
>>> from google.cloud import datastore
>>> client = datastore.Client()
```

Parameters

- **project** (`str`) – (optional) The project to pass to proxied API methods.
- **namespace** (`str`) – (optional) namespace to pass to proxied API methods.
- **credentials** (`Credentials`) – (Optional) The OAuth2 Credentials to use for this client. If not passed (and if no `_http` object is passed), falls back to the default inferred from the environment.
- **_http** (`Session`) – (Optional) HTTP object to make requests. Can be any object that defines `request()` with the same interface as `requests.Session.request()`. If not passed, an `_http` object is created that is bound to the `credentials` for the current object. This parameter should be considered private, and could change in the future.
- **_use_grpc** (`bool`) – (Optional) Explicitly specifies whether to use the gRPC transport (via GAX) or HTTP. If unset, falls back to the `GOOGLE_CLOUD_DISABLE_GRPC` environment variable. This parameter should be considered private, and could change in the future.

allocate_ids (`incomplete_key`, `num_ids`)

Allocate a list of IDs from a partial key.

Parameters

- **incomplete_key** (`google.cloud.datastore.key.Key`) – Partial key to use as base for allocated IDs.
- **num_ids** (`int`) – The number of IDs to allocate.

Return type list of `google.cloud.datastore.key.Key`

Returns The (complete) keys allocated with `incomplete_key` as root.

Raises `ValueError` if `incomplete_key` is not a partial key.

batch()

Proxy to `google.cloud.datastore.batch.Batch`.

current_batch

Currently-active batch.

Return type `google.cloud.datastore.batch.Batch`, or an object implementing its API, or `NoneType` (if no batch is active).

Returns The batch/transaction at the top of the batch stack.

current_transaction

Currently-active transaction.

Return type `google.cloud.datastore.transaction.Transaction`, or an object implementing its API, or `NoneType` (if no transaction is active).

Returns The transaction at the top of the batch stack.

delete (*key*)

Delete the key in the Cloud Datastore.

Note: This is just a thin wrapper over `delete_multi()`. The backend API does not make a distinction between a single key or multiple keys in a commit request.

Parameters **key** (`google.cloud.datastore.key.Key`) – The key to be deleted from the datastore.

delete_multi (*keys*)

Delete keys from the Cloud Datastore.

Parameters **keys** (list of `google.cloud.datastore.key.Key`) – The keys to be deleted from the Datastore.

get (*key*, *missing=None*, *deferred=None*, *transaction=None*)

Retrieve an entity from a single key (if it exists).

Note: This is just a thin wrapper over `get_multi()`. The backend API does not make a distinction between a single key or multiple keys in a lookup request.

Parameters

- **key** (`google.cloud.datastore.key.Key`) – The key to be retrieved from the datastore.
- **missing** (*list*) – (Optional) If a list is passed, the key-only entities returned by the backend as “missing” will be copied into it.
- **deferred** (*list*) – (Optional) If a list is passed, the keys returned by the backend as “deferred” will be copied into it.
- **transaction** (`Transaction`) – (Optional) Transaction to use for read consistency. If not passed, uses current transaction, if set.

Return type `google.cloud.datastore.entity.Entity` or `NoneType`

Returns The requested entity if it exists.

get_multi (*keys*, *missing=None*, *deferred=None*, *transaction=None*)

Retrieve entities, along with their attributes.

Parameters

- **keys** (list of `google.cloud.datastore.key.Key`) – The keys to be retrieved from the datastore.
- **missing** (`list`) – (Optional) If a list is passed, the key-only entities returned by the backend as “missing” will be copied into it. If the list is not empty, an error will occur.
- **deferred** (`list`) – (Optional) If a list is passed, the keys returned by the backend as “deferred” will be copied into it. If the list is not empty, an error will occur.
- **transaction** (`Transaction`) – (Optional) Transaction to use for read consistency. If not passed, uses current transaction, if set.

Return type list of `google.cloud.datastore.entity.Entity`

Returns The requested entities.

Raises `ValueError` if one or more of keys has a project which does not match our project.

key (`*path_args, **kwargs`)

Proxy to `google.cloud.datastore.key.Key`.

Passes our project.

put (`entity`)

Save an entity in the Cloud Datastore.

Note: This is just a thin wrapper over `put_multi()`. The backend API does not make a distinction between a single entity or multiple entities in a commit request.

Parameters **entity** (`google.cloud.datastore.entity.Entity`) – The entity to be saved to the datastore.

put_multi (`entities`)

Save entities in the Cloud Datastore.

Parameters **entities** (list of `google.cloud.datastore.entity.Entity`) – The entities to be saved to the datastore.

Raises `ValueError` if entities is a single entity.

query (`**kwargs`)

Proxy to `google.cloud.datastore.query.Query`.

Passes our project.

Using query to search a datastore:

```
>>> query = client.query(kind='MyKind')
>>> query.add_filter('property', '=', 'val')
```

Using the query iterator

```
>>> query_iter = query.fetch()
>>> for entity in query_iter:
...     do_something(entity)
```

or manually page through results

```
>>> query_iter = query.fetch(start_cursor=cursor)
>>> pages = query_iter.pages
>>>
>>> first_page = next(pages)
>>> first_page_entities = list(first_page)
>>> query_iter.next_page_token
b'...'
```

Parameters **kwargs** (*dict*) – Parameters for initializing and instance of *Query*.

Return type *Query*

Returns A query object.

transaction()

Proxy to *google.cloud.datastore.transaction.Transaction*.

class *google.cloud.datastore.Entity* (*key=None, exclude_from_indexes=()*)

Bases: *dict*

Entities are akin to rows in a relational database

An entity storing the actual instance of data.

Each entity is officially represented with a *Key*, however it is possible that you might create an entity with only a partial key (that is, a key with a kind, and possibly a parent, but without an ID). In such a case, the datastore service will automatically assign an ID to the partial key.

Entities in this API act like dictionaries with extras built in that allow you to delete or persist the data stored on the entity.

Entities are mutable and act like a subclass of a dictionary. This means you could take an existing entity and change the key to duplicate the object.

Use *get()* to retrieve an existing entity:

```
>>> client.get(key)
<Entity('EntityKind', 1234) {'property': 'value'}>
```

You can the set values on the entity just like you would on any other dictionary.

```
>>> entity['age'] = 20
>>> entity['name'] = 'JJ'
```

However, not all types are allowed as a value for a Google Cloud Datastore entity. The following basic types are supported by the API:

- *datetime.datetime*
- *Key*
- *bool*
- *float*
- *int* (as well as *long* in Python 2)
- *unicode* (called *str* in Python 3)
- *bytes* (called *str* in Python 2)
- *GeoPoint*

- `None`

In addition, three container types are supported:

- `list`
- `Entity`
- `dict` (will just be treated like an `Entity` without a `key` or `exclude_from_indexes`)

Each entry in a list must be one of the value types (basic or container) and each value in an `Entity` must as well. In this case an `Entity` as a **container** acts as a `dict`, but also has the special annotations of `key` and `exclude_from_indexes`.

And you can treat an entity like a regular Python dictionary:

```
>>> sorted(entity.keys())
['age', 'name']
>>> sorted(entity.items())
[('age', 20), ('name', 'JJ')]
```

Note: When saving an entity to the backend, values which are “text” (`unicode` in Python2, `str` in Python3) will be saved using the ‘text_value’ field, after being encoded to UTF-8. When retrieved from the back-end, such values will be decoded to “text” again. Values which are “bytes” (`str` in Python2, `bytes` in Python3), will be saved using the ‘blob_value’ field, without any decoding / encoding step.

Parameters

- **key** (`google.cloud.datastore.key.Key`) – Optional key to be set on entity.
- **exclude_from_indexes** (*tuple of string*) – Names of fields whose values are not to be indexed for this entity.

kind

Get the kind of the current entity.

Note: This relies entirely on the `google.cloud.datastore.key.Key` set on the entity. That means that we’re not storing the kind of the entity at all, just the properties and a pointer to a `Key` which knows its Kind.

class `google.cloud.datastore.Key` (*path_args, **kwargs)

Bases: `object`

An immutable representation of a datastore Key.

To create a basic key directly:

```
>>> Key('EntityKind', 1234, project=project)
<Key('EntityKind', 1234), project=...>
>>> Key('EntityKind', 'foo', project=project)
<Key('EntityKind', 'foo'), project=...>
```

Though typical usage comes via the `key()` factory:

```
>>> client.key('EntityKind', 1234)
<Key('EntityKind', 1234), project=...>
```



```
>>> client.key('EntityKind', 'foo')
<Key('EntityKind', 'foo'), project=...>
```

To create a key with a parent:

```
>>> client.key('Parent', 'foo', 'Child', 1234)
<Key('Parent', 'foo', 'Child', 1234), project=...>
>>> client.key('Child', 1234, parent=parent_key)
<Key('Parent', 'foo', 'Child', 1234), project=...>
```

To create a partial key:

```
>>> client.key('Parent', 'foo', 'Child')
<Key('Parent', 'foo', 'Child'), project=...>
```

Parameters

- **path_args** (*tuple of string and integer*) – May represent a partial (odd length) or full (even length) key path.
- **kwargs** (*dict*) – Keyword arguments to be passed in.

Accepted keyword arguments are

- **namespace** (string): A namespace identifier for the key.
- **project** (string): The project associated with the key.
- **parent** (*Key*): The parent of the key.

The project argument is required unless it has been set implicitly.

completed_key (*id_or_name*)

Creates new key from existing partial key by adding final ID/name.

Parameters **id_or_name** (*str or integer*) – ID or name to be added to the key.

Return type *google.cloud.datastore.key.Key*

Returns A new *Key* instance with the same data as the current one and an extra ID or name added.

Raises *ValueError* if the current key is not partial or if *id_or_name* is not a string or integer.

flat_path

Getter for the key path as a tuple.

Return type tuple of string and integer

Returns The tuple of elements in the path.

classmethod from_legacy_urlsafe (*urlsafe*)

Convert urlsafe string to *Key*.

This is intended to work with the “legacy” representation of a datastore “Key” used within Google App Engine (a so-called “Reference”). This assumes that *urlsafe* was created within an App Engine app via something like `ndb.Key(...).urlsafe()`.

Parameters **urlsafe** (*bytes or unicode*) – The base64 encoded (ASCII) string corresponding to a datastore “Key” / “Reference”.

Return type *Key*.

Returns The key corresponding to `urlsafe`.

id

ID getter. Based on the last element of path.

Return type `int`

Returns The (integer) ID of the key.

id_or_name

Getter. Based on the last element of path.

Return type `int` (if `id`) or `string` (if `name`)

Returns The last element of the key's path if it is either an `id` or a `name`.

is_partial

Boolean indicating if the key has an ID (or name).

Return type `bool`

Returns `True` if the last element of the key's path does not have an `id` or a `name`.

kind

Kind getter. Based on the last element of path.

Return type `str`

Returns The kind of the current key.

name

Name getter. Based on the last element of path.

Return type `str`

Returns The (string) name of the key.

namespace

Namespace getter.

Return type `str`

Returns The namespace of the current key.

parent

The parent of the current key.

Return type `google.cloud.datastore.key.Key` or `NoneType`

Returns A new `Key` instance, whose path consists of all but the last element of current path. If the current key has only one path element, returns `None`.

path

Path getter.

Returns a copy so that the key remains immutable.

Return type `list` of `dict`

Returns The (key) path of the current key.

project

Project getter.

Return type `str`

Returns The key's project.

to_legacy_urlsafe()

Convert to a base64 encode urlsafe string for App Engine.

This is intended to work with the “legacy” representation of a datastore “Key” used within Google App Engine (a so-called “Reference”). The returned string can be used as the `urlsafe` argument to `ndb.Key(urlsafe=...)`. The base64 encoded values will have padding removed.

Note: The string returned by `to_legacy_urlsafe` is equivalent, but not identical, to the string returned by `ndb`.

Return type `bytes`

Returns A bytestring containing the key encoded as URL-safe base64.

to_protobuf()

Return a protobuf corresponding to the key.

Return type `entity_pb2.Key`

Returns The protobuf representing the key.

class `google.cloud.datastore.Query` (*client*, *kind=None*, *project=None*, *namespace=None*, *ancestor=None*, *filters=()*, *projection=()*, *order=()*, *distinct_on=()*)

Bases: `object`

A Query against the Cloud Datastore.

This class serves as an abstraction for creating a query over data stored in the Cloud Datastore.

Parameters

- **client** (`google.cloud.datastore.client.Client`) – The client used to connect to Datastore.
- **kind** (`str`) – The kind to query.
- **project** (`str`) – (Optional) The project associated with the query. If not passed, uses the client’s value.
- **namespace** (`str`) – (Optional) The namespace to which to restrict results. If not passed, uses the client’s value.
- **ancestor** (`Key`) – (Optional) key of the ancestor to which this query’s results are restricted.
- **filters** (`tuple[str, str, str]`) – Property filters applied by this query. The sequence is (property_name, operator, value).
- **projection** (`sequence of string`) – fields returned as part of query results.
- **order** (`sequence of string`) – field names used to order query results. Prepend – to a field name to sort it in descending order.
- **distinct_on** (`sequence of string`) – field names used to group query results.

Raises `ValueError` if `project` is not passed and no implicit default is set.

add_filter(property_name, operator, value)

Filter the query based on a property name, operator and a value.

Expressions take the form of:

```
.add_filter('<property>', '<operator>', <value>)
```

where `property` is a property stored on the entity in the datastore and `operator` is one of `OPERATORS` (ie, `=`, `<`, `<=`, `>`, `>=`):

```
>>> from google.cloud import datastore
>>> client = datastore.Client()
>>> query = client.query(kind='Person')
>>> query.add_filter('name', '=', 'James')
>>> query.add_filter('age', '>', 50)
```

Parameters

- **property_name** (*str*) – A property name.
- **operator** (*str*) – One of `=`, `<`, `<=`, `>`, `>=`.
- **value** (*int*, *str*, *bool*, *float*, *NoneType*, *datetime.datetime*, *google.cloud.datastore.key.Key*) – The value to filter on.

Raises `ValueError` if operation is not one of the specified values, or if a filter names `'__key__'` but passes an invalid value (a key is required).

ancestor

The ancestor key for the query.

Return type *Key* or *None*

Returns The ancestor for the query.

distinct_on

Names of fields used to group query results.

Return type sequence of string

Returns The “distinct on” fields set on the query.

fetch (limit=None, offset=0, start_cursor=None, end_cursor=None, client=None)

Execute the Query; return an iterator for the matching entities.

For example:

```
>>> from google.cloud import datastore
>>> client = datastore.Client()
>>> query = client.query(kind='Person')
>>> query.add_filter('name', '=', 'Sally')
>>> list(query.fetch())
[<Entity object>, <Entity object>, ...]
>>> list(query.fetch(1))
[<Entity object>]
```

Parameters

- **limit** (*int*) – (Optional) limit passed through to the iterator.
- **offset** (*int*) – (Optional) offset passed through to the iterator.
- **start_cursor** (*bytes*) – (Optional) cursor passed through to the iterator.
- **end_cursor** (*bytes*) – (Optional) cursor passed through to the iterator.

- **client** (*google.cloud.datastore.client.Client*) – client used to connect to datastore. If not supplied, uses the query's value.

Return type `Iterator`

Returns The iterator for the query.

filters

Filters set on the query.

Return type `tuple[str, str, str]`

Returns The filters set on the query. The sequence is (property_name, operator, value).

key_filter (*key*, *operator*='=')

Filter on a key.

Parameters

- **key** (*google.cloud.datastore.key.Key*) – The key to filter on.
- **operator** (*str*) – (Optional) One of =, <, <=, >, >=. Defaults to =.

keys_only ()

Set the projection to include only keys.

kind

Get the Kind of the Query.

Return type `str`

Returns The kind for the query.

namespace

This query's namespace

Return type `str` or `None`

Returns the namespace assigned to this query

order

Names of fields used to sort query results.

Return type sequence of string

Returns The order(s) set on the query.

project

Get the project for this Query.

Return type `str`

Returns The project for the query.

projection

Fields names returned by the query.

Return type sequence of string

Returns Names of fields in query results.

class `google.cloud.datastore.Transaction` (*client*)

Bases: `google.cloud.datastore.batch.Batch`

An abstraction representing datastore Transactions.

Transactions can be used to build up a bulk mutation and ensure all or none succeed (transactionally).

For example, the following snippet of code will put the two save operations (either insert or upsert) into the same mutation, and execute those within a transaction:

```
>>> with client.transaction():
...     client.put_multi([entity1, entity2])
```

Because it derives from *Batch*, *Transaction* also provides `put()` and `delete()` methods:

```
>>> with client.transaction() as xact:
...     xact.put(entity1)
...     xact.delete(entity2.key)
```

By default, the transaction is rolled back if the transaction block exits with an error:

```
>>> with client.transaction():
...     do_some_work()
...     raise SomeException # rolls back
Traceback (most recent call last):
...
SomeException
```

If the transaction block exits without an exception, it will commit by default.

Warning:

Inside a transaction, automatically assigned IDs for entities will not be available at save time! That means, if you try:

```
>>> with client.transaction():
...     entity = Entity(key=client.key('Thing'))
...     client.put(entity)
```

entity won't have a complete key until the transaction is committed.

Once you exit the transaction (or call `commit()`), the automatically generated ID will be assigned to the entity:

```
>>> with client.transaction():
...     entity = Entity(key=client.key('Thing'))
...     client.put(entity)
...     print(entity.key.is_partial) # There is no ID on this key.
...
True
>>> print(entity.key.is_partial) # There is an ID.
False
```

If you don't want to use the context manager you can initialize a transaction manually:

```
>>> transaction = client.transaction()
>>> transaction.begin()
>>>
>>> entity = Entity(key=client.key('Thing'))
>>> transaction.put(entity)
>>>
>>> transaction.commit()
```

Parameters `client` (*google.cloud.datastore.client.Client*) – the client used to connect to datastore.

begin()

Begins a transaction.

This method is called automatically when entering a `with` statement, however it can be called explicitly if you don't want to use a context manager.

Raises `ValueError` if the transaction has already begun.

commit()

Commits the transaction.

This is called automatically upon exiting a `with` statement, however it can be called explicitly if you don't want to use a context manager.

This method has necessary side-effects:

- Sets the current transaction's ID to `None`.

current()

Return the topmost transaction.

Note: If the topmost element on the stack is not a transaction, returns `None`.

Return type *google.cloud.datastore.transaction.Transaction* or `None`

Returns The current transaction (if any are active).

id

Getter for the transaction ID.

Return type `str`

Returns The ID of the current transaction.

rollback()

Rolls back the current transaction.

This method has necessary side-effects:

- Sets the current transaction's ID to `None`.

8.1 DNS Client

Client for interacting with the Google Cloud DNS API.

class `google.cloud.dns.client.Client` (*project=None, credentials=None, _http=None*)
Bases: `google.cloud.client.ClientWithProject`

Client to bundle configuration needed for API requests.

Parameters

- **project** (*str*) – the project which the client acts on behalf of. Will be passed when creating a zone. If not passed, falls back to the default inferred from the environment.
- **credentials** (*Credentials*) – (Optional) The OAuth2 Credentials to use for this client. If not passed (and if no `_http` object is passed), falls back to the default inferred from the environment.
- **_http** (*Session*) – (Optional) HTTP object to make requests. Can be any object that defines `request()` with the same interface as `requests.Session.request()`. If not passed, an `_http` object is created that is bound to the `credentials` for the current object. This parameter should be considered private, and could change in the future.

SCOPE = ('<https://www.googleapis.com/auth/ndev.cloudndns.readwrite>',)

The scopes required for authenticating as a Cloud DNS consumer.

list_zones (*max_results=None, page_token=None*)

List zones for the project associated with this client.

See <https://cloud.google.com/dns/api/v1/managedZones/list>

Parameters

- **max_results** (*int*) – maximum number of zones to return, If not passed, defaults to a value set by the API.

- **page_token** (*str*) – opaque marker for the next “page” of zones. If not passed, the API will return the first page of zones.

Return type `Iterator`

Returns Iterator of `ManagedZone` belonging to this project.

quotas ()

Return DNS quotas for the project associated with this client.

See <https://cloud.google.com/dns/api/v1/projects/get>

Return type mapping

Returns keys for the mapping correspond to those of the `quota` sub-mapping of the project resource.

zone (*name*, *dns_name=None*, *description=None*)

Construct a zone bound to this client.

Parameters

- **name** (*str*) – Name of the zone.
- **dns_name** (*str*) – (Optional) DNS name of the zone. If not passed, then calls to `zone.create()` will fail.
- **description** (*str*) – (Optional) the description for the zone. If not passed, defaults to the value of ‘dns_name’.

Return type `google.cloud.dns.zone.ManagedZone`

Returns a new `ManagedZone` instance.

8.2 Managed Zones

Define API `ManagedZones`.

```
class google.cloud.dns.zone.ManagedZone (name, dns_name=None, client=None, description=None)
```

Bases: `object`

`ManagedZones` are containers for DNS resource records.

See <https://cloud.google.com/dns/api/v1/managedZones>

Parameters

- **name** (*str*) – the name of the zone
- **dns_name** (*str*) – (Optional) the DNS name of the zone. If not passed, then calls to `create()` will fail.
- **client** (`google.cloud.dns.client.Client`) – A client which holds credentials and project configuration for the zone (which requires a project).
- **description** (*str*) – (Optional) the description for the zone. If not passed, defaults to the value of ‘dns_name’.

changes ()

Construct a change set bound to this zone.

Return type `google.cloud.dns.changes.Changes`

Returns a new Changes instance

create (*client=None*)

API call: create the zone via a PUT request

See <https://cloud.google.com/dns/api/v1/managedZones/create>

Parameters **client** (*google.cloud.dns.client.Client*) – (Optional) the client to use. If not passed, falls back to the `client` stored on the current zone.

created

Datetime at which the zone was created.

Return type `datetime.datetime`, or `NoneType`

Returns the creation time (None until set from the server).

delete (*client=None*)

API call: delete the zone via a DELETE request

See <https://cloud.google.com/dns/api/v1/managedZones/delete>

Parameters **client** (*google.cloud.dns.client.Client*) – (Optional) the client to use. If not passed, falls back to the `client` stored on the current zone.

description

Description of the zone.

Return type `str`, or `NoneType`

Returns The description as set by the user, or None (the default).

exists (*client=None*)

API call: test for the existence of the zone via a GET request

See <https://cloud.google.com/dns/api/v1/managedZones/get>

Parameters **client** (*google.cloud.dns.client.Client*) – (Optional) the client to use. If not passed, falls back to the `client` stored on the current zone.

Return type `bool`

Returns Boolean indicating existence of the managed zone.

classmethod from_api_repr (*resource, client*)

Factory: construct a zone given its API representation

Parameters

- **resource** (*dict*) – zone resource representation returned from the API
- **client** (*google.cloud.dns.client.Client*) – Client which holds credentials and project configuration for the zone.

Return type *google.cloud.dns.zone.ManagedZone*

Returns Zone parsed from `resource`.

list_changes (*max_results=None, page_token=None, client=None*)

List change sets for this zone.

See <https://cloud.google.com/dns/api/v1/resourceRecordSets/list>

Parameters

- **max_results** (*int*) – maximum number of zones to return, If not passed, defaults to a value set by the API.

- **page_token** (*str*) – opaque marker for the next “page” of zones. If not passed, the API will return the first page of zones.
- **client** (*google.cloud.dns.client.Client*) – (Optional) the client to use. If not passed, falls back to the `client` stored on the current zone.

Return type `Iterator`

Returns Iterator of *Changes* belonging to this zone.

list_resource_record_sets (*max_results=None, page_token=None, client=None*)

List resource record sets for this zone.

See <https://cloud.google.com/dns/api/v1/resourceRecordSets/list>

Parameters

- **max_results** (*int*) – maximum number of zones to return, If not passed, defaults to a value set by the API.
- **page_token** (*str*) – opaque marker for the next “page” of zones. If not passed, the API will return the first page of zones.
- **client** (*google.cloud.dns.client.Client*) – (Optional) the client to use. If not passed, falls back to the `client` stored on the current zone.

Return type `Iterator`

Returns Iterator of *ResourceRecordSet* belonging to this zone.

name_server_set

Named set of DNS name servers that all host the same ManagedZones.

Most users will leave this blank.

See <https://cloud.google.com/dns/api/v1/managedZones#nameServerSet>

Return type `str`, or `NoneType`

Returns The name as set by the user, or `None` (the default).

name_servers

Datetime at which the zone was created.

Return type list of strings, or `NoneType`.

Returns the assigned name servers (`None` until set from the server).

path

URL path for the zone’s APIs.

Return type `str`

Returns the path based on project and dataste name.

project

Project bound to the zone.

Return type `str`

Returns the project (derived from the client).

reload (*client=None*)

API call: refresh zone properties via a GET request

See <https://cloud.google.com/dns/api/v1/managedZones/get>

Parameters `client` (*google.cloud.dns.client.Client*) – (Optional) the client to use. If not passed, falls back to the `client` stored on the current zone.

resource_record_set (*name, record_type, ttl, rrdatas*)

Construct a resource record set bound to this zone.

Parameters

- **name** (*str*) – Name of the record set.
- **record_type** (*str*) – RR type
- **ttl** (*int*) – TTL for the RR, in seconds
- **rrdatas** (*list of string*) – resource data for the RR

Return type *google.cloud.dns.resource_record_set.ResourceRecordSet*

Returns a new `ResourceRecordSet` instance

zone_id

ID for the zone resource.

Return type *str*, or `NoneType`

Returns the ID (None until set from the server).

8.3 Resource Record Sets

Define API `ResourceRecordSets`.

class `google.cloud.dns.resource_record_set.ResourceRecordSet` (*name, record_type, ttl, rrdatas, zone*)

Bases: `object`

`ResourceRecordSets` are DNS resource records.

RRS are owned by a *google.cloud.dns.zone.ManagedZone* instance.

See <https://cloud.google.com/dns/api/v1/resourceRecordSets>

Parameters

- **name** (*str*) – the name of the record set.
- **record_type** (*str*) – the RR type of the zone.
- **ttl** (*int*) – TTL (in seconds) for caching the record sets.
- **rrdatas** (*list of string*) – one or more lines containing the resource data.
- **zone** (*google.cloud.dns.zone.ManagedZone*) – A zone which holds one or more record sets.

classmethod `from_api_repr` (*resource, zone*)

Factory: construct a record set given its API representation

Parameters

- **resource** (*dict*) – record sets representation returned from the API
- **zone** (*google.cloud.dns.zone.ManagedZone*) – A zone which holds one or more record sets.

Return type *google.cloud.dns.zone.ResourceRecordSet*

Returns RRS parsed from `resource`.

8.4 Change Sets

Define API ResourceRecordSets.

class `google.cloud.dns.changes.Changes` (*zone*)

Bases: `object`

Changes are bundled additions / deletions of DNS resource records.

Changes are owned by a `google.cloud.dns.zone.ManagedZone` instance.

See <https://cloud.google.com/dns/api/v1/changes>

Parameters `zone` (`google.cloud.dns.zone.ManagedZone`) – A zone which holds one or more record sets.

add_record_set (*record_set*)

Append a record set to the ‘additions’ for the change set.

Parameters `record_set` (`google.cloud.dns.resource_record_set.ResourceRecordSet`) – the record set to append.

Raises `ValueError` if `record_set` is not of the required type.

additions

Resource record sets to be added to the zone.

Return type sequence of `google.cloud.dns.resource_record_set.ResourceRecordSet`.

Returns record sets appended via `add_record_set()`.

create (*client=None*)

API call: create the change set via a POST request.

See <https://cloud.google.com/dns/api/v1/changes/create>

Parameters `client` (`google.cloud.dns.client.Client`) – (Optional) the client to use. If not passed, falls back to the `client` stored on the current zone.

delete_record_set (*record_set*)

Append a record set to the ‘deletions’ for the change set.

Parameters `record_set` (`google.cloud.dns.resource_record_set.ResourceRecordSet`) – the record set to append.

Raises `ValueError` if `record_set` is not of the required type.

deletions

Resource record sets to be deleted from the zone.

Return type sequence of `google.cloud.dns.resource_record_set.ResourceRecordSet`.

Returns record sets appended via `delete_record_set()`.

exists (*client=None*)

API call: test for the existence of the change set via a GET request.

See <https://cloud.google.com/dns/api/v1/changes/get>

Parameters `client` (*google.cloud.dns.client.Client*) – (Optional) the client to use. If not passed, falls back to the `client` stored on the current zone.

Return type `bool`

Returns Boolean indicating existence of the changes.

classmethod `from_api_repr(resource, zone)`

Factory: construct a change set given its API representation

Parameters

- **resource** (*dict*) – change set representation returned from the API.
- **zone** (*google.cloud.dns.zone.ManagedZone*) – A zone which holds zero or more change sets.

Return type *google.cloud.dns.changes.Changes*

Returns RRS parsed from `resource`.

name

Name of the change set.

Return type `str` or `NoneType`

Returns Name, as set by the back-end, or `None`.

path

URL path for change set APIs.

Return type `str`

Returns the path based on project, zone, and change set names.

reload (*client=None*)

API call: refresh zone properties via a GET request.

See <https://cloud.google.com/dns/api/v1/changes/get>

Parameters `client` (*google.cloud.dns.client.Client*) – (Optional) the client to use. If not passed, falls back to the `client` stored on the current zone.

started

Time when the change set was started.

Return type `datetime.datetime` or `NoneType`

Returns Time, as set by the back-end, or `None`.

status

Status of the change set.

Return type `str` or `NoneType`

Returns Status, as set by the back-end, or `None`.

8.5 Client

Client objects provide a means to configure your DNS applications. Each instance holds both a `project` and an authenticated connection to the DNS service.

For an overview of authentication in `google-cloud-python`, see [Authentication](#).

Assuming your environment is set up as described in that document, create an instance of *Client*.

```
>>> from google.cloud import dns
>>> client = dns.Client()
```

8.6 Projects

A project is the top-level container in the DNS API: it is tied closely to billing, and can provide default access control across all its datasets. If no `project` is passed to the client container, the library attempts to infer a project using the environment (including explicit environment variables, GAE, or GCE).

To override the project inferred from the environment, pass an explicit `project` to the constructor, or to either of the alternative classmethod factories:

```
>>> from google.cloud import dns
>>> client = dns.Client(project='PROJECT_ID')
```

8.7 Project Quotas

Query the quotas for a given project:

```
>>> from google.cloud import dns
>>> client = dns.Client(project='PROJECT_ID')
>>> quotas = client.quotas() # API request
>>> for key, value in sorted(quotas.items()):
...     print('%s: %s' % (key, value))
managedZones: 10000
resourceRecordsPerRrset: 100
rrsetsPerManagedZone: 10000
rrsetAdditionsPerChange: 100
rrsetDeletionsPerChange: 100
totalRrdataSizePerChange: 10000
```

8.7.1 Project ACLs

Each project has an access control list granting reader / writer / owner permission to one or more entities. This list cannot be queried or set via the API: it must be managed using the Google Developer Console.

8.8 Managed Zones

A “managed zone” is the container for DNS records for the same DNS name suffix and has a set of name servers that accept and responds to queries:

```
>>> from google.cloud import dns
>>> client = dns.Client(project='PROJECT_ID')
>>> zone = client.zone('acme-co', 'example.com',
...                   description='Acme Company zone')

>>> zone.exists() # API request
False
>>> zone.create() # API request
```



```
>>> zone.exists() # API request
True
```

List the zones for a given project:

```
>>> from google.cloud import dns
>>> client = dns.Client(project='PROJECT_ID')
>>> zones = client.list_zones() # API request
>>> [zone.name for zone in zones]
['acme-co']
```

8.9 Resource Record Sets

Each managed zone exposes a read-only set of resource records:

```
>>> from google.cloud import dns
>>> client = dns.Client(project='PROJECT_ID')
>>> zone = client.zone('acme-co', 'example.com')
>>> records, page_token = zone.list_resource_record_sets() # API request
>>> [(record.name, record.record_type, record.ttl, record.rdatas)
...   for record in records]
[('example.com.', 'SOA', 21600, ['ns-cloud1.googlecomains.com dns-admin.
↪google.com 1 21600 3600 1209600 300'])]
```

Note: The `page_token` returned from `zone.list_resource_record_sets()` will be an opaque string if there are more resources than can be returned in a single request. To enumerate them all, repeat calling `zone.list_resource_record_sets()`, passing the `page_token`, until the token is `None`. E.g.

```
>>> records, page_token = zone.list_resource_record_sets() # API request
>>> while page_token is not None:
...     next_batch, page_token = zone.list_resource_record_sets(
...         page_token=page_token) # API request
...     records.extend(next_batch)
```

8.10 Change requests

Update the resource record set for a zone by creating a change request bundling additions to or deletions from the set.

```
>>> import time
>>> from google.cloud import dns
>>> client = dns.Client(project='PROJECT_ID')
>>> zone = client.zone('acme-co', 'example.com')
>>> TWO_HOURS = 2 * 60 * 60 # seconds
>>> record_set = zone.resource_record_set(
...     'www.example.com.', 'CNAME', TWO_HOURS, ['www1.example.com.'])
>>> changes = zone.changes()
>>> changes.add_record_set(record_set)
>>> changes.create() # API request
>>> while changes.status != 'done':
...     print('Waiting for changes to complete')
```

```
...     time.sleep(60)         # or whatever interval is appropriate
...     changes.reload()      # API request
```

List changes made to the resource record set for a given zone:

```
>>> from google.cloud import dns
>>> client = dns.Client(project='PROJECT_ID')
>>> zone = client.zone('acme-co', 'example.com')
>>> changes = []
>>> changes, page_token = zone.list_changes() # API request
```

Note: The `page_token` returned from `zone.list_changes()` will be an opaque string if there are more changes than can be returned in a single request. To enumerate them all, repeat calling `zone.list_changes()`, passing the `page_token`, until the token is `None`. E.g.:

```
>>> changes, page_token = zone.list_changes() # API request
>>> while page_token is not None:
...     next_batch, page_token = zone.list_changes(
...         page_token=page_token) # API request
...     changes.extend(next_batch)
```

The [Google Natural Language API](#) can be used to reveal the structure and meaning of text via powerful machine learning models. You can use it to extract information about people, places, events and much more, mentioned in text documents, news articles or blog posts. You can use it to understand sentiment about your product on social media or parse intent from customer conversations happening in a call center or a messaging app. You can analyze text uploaded in your request or integrate with your document storage on Google Cloud Storage.

9.1 Authentication and Configuration

- For an overview of authentication in `google-cloud-python`, see [Authentication](#).
- In addition to any authentication configuration, you should also set the `GOOGLE_CLOUD_PROJECT` environment variable for the project you'd like to interact with. If the `GOOGLE_CLOUD_PROJECT` environment variable is not present, the project ID from JSON file credentials is used.

If you are using Google App Engine or Google Compute Engine this will be detected automatically.

- After configuring your environment, create a `LanguageServiceClient`.

```
>>> from google.cloud import language
>>> client = language.LanguageServiceClient()
```

or pass in `credentials` explicitly.

```
>>> from google.cloud import language
>>> client = language.LanguageServiceClient(
...     credentials=creds,
... )
```

9.2 Documents

The Google Natural Language API has three supported methods

- `analyzeEntities`
- `analyzeSentiment`
- `annotateText`

and each method uses a *Document* for representing text.

```
>>> document = language.types.Document(  
...     content='Google, headquartered in Mountain View, unveiled the '  
...             'new Android phone at the Consumer Electronic Show. '  
...             'Sundar Pichai said in his keynote that users love '  
...             'their new Android phones.',  
...     language='en',  
...     type='PLAIN_TEXT',  
... )
```

The document's language defaults to `None`, which will cause the API to auto-detect the language.

In addition, you can construct an HTML document:

```
>>> html_content = """\  
... <html>  
...   <head>  
...     <title>El Tiempo de las Historias</title>  
...   </head>  
...   <body>  
...     <p>La vaca saltó sobre la luna.</p>  
...   </body>  
... </html>  
... """  
>>> document = language.types.Document(  
...     content=html_content,  
...     language='es',  
...     type='HTML',  
... )
```

The language argument can be either ISO-639-1 or BCP-47 language codes. The API reference page contains the full list of [supported languages](#).

In addition to supplying the text / HTML content, a document can refer to content stored in [Google Cloud Storage](#).

```
>>> document = language.types.Document(  
...     gcs_content_uri='gs://my-text-bucket/sentiment-me.txt',  
...     type=language.enums.HTML,  
... )
```

9.3 Analyze Entities

The `analyze_entities()` method finds named entities (i.e. proper names) in the text. This method returns a *AnalyzeEntitiesResponse*.

```
>>> document = language.types.Document(  
...     content='Michelangelo Caravaggio, Italian painter, is '  
...             'known for "The Calling of Saint Matthew".',  
...     type=language.enums.Type.PLAIN_TEXT,  
... )  
>>> response = client.analyze_entities(  
...     document=
```

```

...     document=document,
...     encoding_type='UTF32',
... )
>>> for entity in response.entities:
...     print('=' * 20)
...     print('         name: {}'.format(entity.name))
...     print('         type: {}'.format(entity.entity_type))
...     print('         metadata: {}'.format(entity.metadata))
...     print('         salience: {}'.format(entity.salience))
=====
         name: Michelangelo Caravaggio
         type: PERSON
         metadata: {'wikipedia_url': 'https://en.wikipedia.org/wiki/Caravaggio'}
         salience: 0.7615959
=====
         name: Italian
         type: LOCATION
         metadata: {'wikipedia_url': 'https://en.wikipedia.org/wiki/Italy'}
         salience: 0.19960518
=====
         name: The Calling of Saint Matthew
         type: EVENT
         metadata: {'wikipedia_url': 'https://en.wikipedia.org/wiki/The_Calling_
↳of_St_Matthew_(Caravaggio)'}
         salience: 0.038798928

```

Note: It is recommended to send an `encoding_type` argument to Natural Language methods, so they provide useful offsets for the data they return. While the correct value varies by environment, in Python you *usually* want UTF32.

9.4 Analyze Sentiment

The `analyze_sentiment()` method analyzes the sentiment of the provided text. This method returns a `AnalyzeSentimentResponse`.

```

>>> document = language.types.Document(
...     content='Jogging is not very fun.',
...     type='PLAIN_TEXT',
... )
>>> response = client.analyze_sentiment(
...     document=document,
...     encoding_type='UTF32',
... )
>>> sentiment = response.document_sentiment
>>> print(sentiment.score)
-1
>>> print(sentiment.magnitude)
0.8

```

Note: It is recommended to send an `encoding_type` argument to Natural Language methods, so they provide useful offsets for the data they return. While the correct value varies by environment, in Python you *usually* want

UTF32.

9.5 Annotate Text

The `annotate_text()` method analyzes a document and is intended for users who are familiar with machine learning and need in-depth text features to build upon. This method returns a `AnnotateTextResponse`.

9.6 API Reference

This package includes clients for multiple versions of the Natural Language API. By default, you will get v1, the latest GA version.

9.6.1 Natural Language Client API

```
class google.cloud.language_v1.LanguageServiceClient (channel=None,          cre-  
                                                    dentials=None,  
                                                    ssl_credentials=None,  
                                                    scopes=None,  
                                                    client_config=None,  
                                                    lib_name=None,  
                                                    lib_version="",          met-  
                                                    rics_headers=())
```

Provides text analysis operations such as sentiment analysis and entity recognition.

Constructor.

Parameters

- **channel** (*Channel*) – A `Channel` instance through which to make calls.
- **credentials** (*Credentials*) – The authorization credentials to attach to requests. These credentials identify this application to the service.
- **ssl_credentials** (*ChannelCredentials*) – A `ChannelCredentials` instance for use with an SSL-enabled channel.
- **scopes** (*Sequence[str]*) – A list of OAuth2 scopes to attach to requests.
- **client_config** (*dict*) – A dictionary for call options for each method. See `google.gax.construct_settings()` for the structure of this data. Falls back to the default config if not specified or the specified config is missing data points.
- **lib_name** (*str*) – The API library software used for calling the service. (Unless you are writing an API client itself, leave this as default.)
- **lib_version** (*str*) – The API library software version used for calling the service. (Unless you are writing an API client itself, leave this as default.)
- **metrics_headers** (*dict*) – A dictionary of values for tracking client library metrics. Ultimately serializes to a string (e.g. 'foo/1.2.3 bar/3.14.1'). This argument should be considered private.

analyze_entities (*document, encoding_type=None, options=None*)

Finds named entities (currently proper names and common nouns) in the text along with entity types, salience, mentions for each entity, and other properties.

Example

```
>>> from google.cloud import language_v1
>>>
>>> client = language_v1.LanguageServiceClient()
>>>
>>> document = {}
>>>
>>> response = client.analyze_entities(document)
```

Parameters

- **document** (*Union[dict, Document]*) – Input document. If a dict is provided, it must be of the same form as the protobuf message *Document*
- **encoding_type** (*EncodingType*) – The encoding type used by the API to calculate offsets.
- **options** (*CallOptions*) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A *AnalyzeEntitiesResponse* instance.

Raises

- *google.gax.errors.GaxError* if the RPC is aborted.
- *ValueError* if the parameters are invalid.

analyze_entity_sentiment (*document, encoding_type=None, options=None*)

Finds entities, similar to *AnalyzeEntities* in the text and analyzes sentiment associated with each entity and its mentions.

Example

```
>>> from google.cloud import language_v1
>>>
>>> client = language_v1.LanguageServiceClient()
>>>
>>> document = {}
>>>
>>> response = client.analyze_entity_sentiment(document)
```

Parameters

- **document** (*Union[dict, Document]*) – Input document. If a dict is provided, it must be of the same form as the protobuf message *Document*
- **encoding_type** (*EncodingType*) – The encoding type used by the API to calculate offsets.
- **options** (*CallOptions*) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A *AnalyzeEntitySentimentResponse* instance.

Raises

- *google.gax.errors.GaxError* if the RPC is aborted.

- `ValueError` if the parameters are invalid.

analyze_sentiment (*document*, *encoding_type=None*, *options=None*)
Analyzes the sentiment of the provided text.

Example

```
>>> from google.cloud import language_v1
>>>
>>> client = language_v1.LanguageServiceClient()
>>>
>>> document = {}
>>>
>>> response = client.analyze_sentiment(document)
```

Parameters

- **document** (*Union[dict, Document]*) – Input document. If a dict is provided, it must be of the same form as the protobuf message *Document*
- **encoding_type** (*EncodingType*) – The encoding type used by the API to calculate sentence offsets.
- **options** (*CallOptions*) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A *AnalyzeSentimentResponse* instance.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

analyze_syntax (*document*, *encoding_type=None*, *options=None*)
Analyzes the syntax of the text and provides sentence boundaries and tokenization along with part of speech tags, dependency trees, and other properties.

Example

```
>>> from google.cloud import language_v1
>>>
>>> client = language_v1.LanguageServiceClient()
>>>
>>> document = {}
>>>
>>> response = client.analyze_syntax(document)
```

Parameters

- **document** (*Union[dict, Document]*) – Input document. If a dict is provided, it must be of the same form as the protobuf message *Document*
- **encoding_type** (*EncodingType*) – The encoding type used by the API to calculate offsets.

- **options** (*CallOptions*) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A *AnalyzeSyntaxResponse* instance.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

annotate_text (*document, features, encoding_type=None, options=None*)

A convenience method that provides all the features that `analyzeSentiment`, `analyzeEntities`, and `analyzeSyntax` provide in one call.

Example

```
>>> from google.cloud import language_v1
>>>
>>> client = language_v1.LanguageServiceClient()
>>>
>>> document = {}
>>> features = {}
>>>
>>> response = client.annotate_text(document, features)
```

Parameters

- **document** (*Union[dict, Document]*) – Input document. If a dict is provided, it must be of the same form as the protobuf message *Document*
- **features** (*Union[dict, Features]*) – The enabled features. If a dict is provided, it must be of the same form as the protobuf message *Features*
- **encoding_type** (*EncodingType*) – The encoding type used by the API to calculate offsets.
- **options** (*CallOptions*) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A *AnnotateTextResponse* instance.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

```
enums = <module 'google.cloud.language_v1.gapic.enums' from '/home/docs/checkouts/read
```

9.6.2 Natural Language Client Types

class `google.cloud.language_v1.types.AnalyzeEntitiesRequest`

The entity analysis request message.

document

Input document.

encoding_type

The encoding type used by the API to calculate offsets.

class google.cloud.language_v1.types.**AnalyzeEntitiesResponse**

The entity analysis response message.

entities

The recognized entities in the input document.

language

The language of the text, which will be the same as the language specified in the request or, if not specified, the automatically-detected language. See [Document.language][google.cloud.language.v1.Document.language] field for more details.

class google.cloud.language_v1.types.**AnalyzeEntitySentimentRequest**

The entity-level sentiment analysis request message.

document

Input document.

encoding_type

The encoding type used by the API to calculate offsets.

class google.cloud.language_v1.types.**AnalyzeEntitySentimentResponse**

The entity-level sentiment analysis response message.

entities

The recognized entities in the input document with associated sentiments.

language

The language of the text, which will be the same as the language specified in the request or, if not specified, the automatically-detected language. See [Document.language][google.cloud.language.v1.Document.language] field for more details.

class google.cloud.language_v1.types.**AnalyzeSentimentRequest**

The sentiment analysis request message.

document

Input document.

encoding_type

The encoding type used by the API to calculate sentence offsets.

class google.cloud.language_v1.types.**AnalyzeSentimentResponse**

The sentiment analysis response message.

document_sentiment

The overall sentiment of the input document.

language

The language of the text, which will be the same as the language specified in the request or, if not specified, the automatically-detected language. See [Document.language][google.cloud.language.v1.Document.language] field for more details.

sentences

The sentiment for all the sentences in the document.

class google.cloud.language_v1.types.**AnalyzeSyntaxRequest**

The syntax analysis request message.

document

Input document.

encoding_type

The encoding type used by the API to calculate offsets.

class google.cloud.language_v1.types.**AnalyzeSyntaxResponse**

The syntax analysis response message.

sentences

Sentences in the input document.

tokens

Tokens, along with their syntactic information, in the input document.

language

The language of the text, which will be the same as the language specified in the request or, if not specified, the automatically-detected language. See [Document.language][google.cloud.language.v1.Document.language] field for more details.

class google.cloud.language_v1.types.**AnnotateTextRequest**

The request message for the text annotation API, which can perform multiple analysis types (sentiment, entities, and syntax) in one call.

extract_syntax

Extract syntax information.

extract_entities

Extract entities.

extract_document_sentiment

Extract document-level sentiment.

extract_entity_sentiment

Extract entities and their associated sentiment.

document

Input document.

features

The enabled features.

encoding_type

The encoding type used by the API to calculate offsets.

class Features

All available features for sentiment, syntax, and semantic analysis. Setting each one to true will enable that specific analysis for the input.

class google.cloud.language_v1.types.**AnnotateTextResponse**

The text annotations response message.

sentences

Sentences in the input document. Populated if the user enables [AnnotateTextRequest.Features.extract_syntax][google.cloud.language.v1.AnnotateTextRequest.Features.extract_syntax].

tokens

Tokens, along with their syntactic information, in the input document. Populated if the user enables [AnnotateTextRequest.Features.extract_syntax][google.cloud.language.v1.AnnotateTextRequest.Features.extract_syntax].

entities

Entities, along with their semantic information, in the input document. Populated if the user enables [AnnotateTextRequest.Features.extract_entities][google.cloud.language.v1.AnnotateTextRequest.Features.extract_entities].

document_sentiment

The overall sentiment for the document. Populated if the user enables [AnnotateTextRequest.Features.extract_document_sentiment][google.cloud.language.v1.AnnotateTextRequest.Features.extract_document_sentiment].

language

The language of the text, which will be the same as the language specified in the request or, if not specified, the automatically-detected language. See [Document.language][google.cloud.language.v1.Document.language] field for more details.

class google.cloud.language_v1.types.DependencyEdge

Represents dependency parse tree information for a token. (For more information on dependency labels, see <http://www.aclweb.org/anthology/P13-2017>)

head_token_index

Represents the head of this token in the dependency tree. This is the index of the token which has an arc going to this token. The index is the position of the token in the array of tokens returned by the API method. If this token is a root token, then the `head_token_index` is its own index.

label

The parse label for the token.

class google.cloud.language_v1.types.Document

Represents the input to API methods.

type

Required. If the type is not set or is `TYPE_UNSPECIFIED`, returns an `INVALID_ARGUMENT` error.

source

The source of the document: a string containing the content or a Google Cloud Storage URI.

content

The content of the input in string format.

gcs_content_uri

The Google Cloud Storage URI where the file content is located. This URI must be of the form: `gs://bucket_name/object_name`. For more details, see <https://cloud.google.com/storage/docs/reference-uris>. NOTE: Cloud Storage object versioning is not supported.

language

The language of the document (if not specified, the language is automatically detected). Both ISO and BCP-47 language codes are accepted. [Language Support](#) lists currently supported languages for each API method. If the language (either specified by the caller or automatically detected) is not supported by the called API method, an `INVALID_ARGUMENT` error is returned.

class google.cloud.language_v1.types.Entity

Represents a phrase in the text that is a known entity, such as a person, an organization, or location. The API associates information, such as salience and mentions, with entities.

name

The representative name for the entity.

type

The entity type.

metadata

Metadata associated with the entity. Currently, Wikipedia URLs and Knowledge Graph MIDs are provided, if available. The associated keys are “`wikipedia_url`” and “`mid`”, respectively.

salience

The salience score associated with the entity in the [0, 1.0] range. The salience score for an entity provides

information about the importance or centrality of that entity to the entire document text. Scores closer to 0 are less salient, while scores closer to 1.0 are highly salient.

mentions

The mentions of this entity in the input document. The API currently supports proper noun mentions.

sentiment

For calls to `[AnalyzeEntitySentiment][]` or if `[AnnotateTextRequest.Features.extract_entity_sentiment][google.cloud.language.v1.AnnotateTextRequest.Features.extract_entity_sentiment]` is set to true, this field will contain the aggregate sentiment expressed for this entity in the provided document.

class `google.cloud.language_v1.types.EntityMention`

Represents a mention for an entity in the text. Currently, proper noun mentions are supported.

text

The mention text.

type

The type of the entity mention.

sentiment

For calls to `[AnalyzeEntitySentiment][]` or if `[AnnotateTextRequest.Features.extract_entity_sentiment][google.cloud.language.v1.AnnotateTextRequest.Features.extract_entity_sentiment]` is set to true, this field will contain the sentiment expressed for this mention of the entity in the provided document.

class `google.cloud.language_v1.types.PartOfSpeech`

Represents part of speech information for a token. Parts of speech are as defined in http://www.lrec-conf.org/proceedings/lrec2012/pdf/274_Paper.pdf

tag

The part of speech tag.

aspect

The grammatical aspect.

case

The grammatical case.

form

The grammatical form.

gender

The grammatical gender.

mood

The grammatical mood.

number

The grammatical number.

person

The grammatical person.

proper

The grammatical properness.

reciprocity

The grammatical reciprocity.

tense

The grammatical tense.

voice

The grammatical voice.

class google.cloud.language_v1.types.Sentence

Represents a sentence in the input document.

text

The sentence text.

sentiment

For calls to [AnalyzeSentiment][] or if [AnnotateTextRequest.Features.extract_document_sentiment][google.cloud.language.v1.AnnotateTextRequest.Features.extract_document_sentiment] is set to true, this field will contain the sentiment for the sentence.

class google.cloud.language_v1.types.Sentiment

Represents the feeling associated with the entire text or entities in the text.

magnitude

A non-negative number in the [0, +inf) range, which represents the absolute magnitude of sentiment regardless of score (positive or negative).

score

Sentiment score between -1.0 (negative sentiment) and 1.0 (positive sentiment).

class google.cloud.language_v1.types.TextSpan

Represents an output piece of text.

content

The content of the output text.

begin_offset

The API calculates the beginning offset of the content in the original document according to the [EncodingType][google.cloud.language.v1.EncodingType] specified in the API request.

class google.cloud.language_v1.types.Token

Represents the smallest syntactic building block of the text.

text

The token text.

part_of_speech

Parts of speech tag for this token.

dependency_edge

Dependency tree parse for this token.

lemma

[Lemma](#) of the token.

If you are interested in beta features ahead of the latest GA, you may opt-in to the v1.1 beta, which is spelled `v1beta2`. In order to do this, you will want to import from `google.cloud.language_v1beta2` in lieu of `google.cloud.language`.

An API and type reference is provided for the v1.1 beta also:

9.6.3 Natural Language Beta Client API

```
class google.cloud.language_v1beta2.LanguageServiceClient (channel=None,
                                                            credentials=None,
                                                            ssl_credentials=None,
                                                            scopes=None,
                                                            client_config=None,
                                                            lib_name=None,
                                                            lib_version="",
                                                            metrics_headers=())
```

Provides text analysis operations such as sentiment analysis and entity recognition.

Constructor.

Parameters

- **channel** (*Channel*) – A *Channel* instance through which to make calls.
- **credentials** (*Credentials*) – The authorization credentials to attach to requests. These credentials identify this application to the service.
- **ssl_credentials** (*ChannelCredentials*) – A *ChannelCredentials* instance for use with an SSL-enabled channel.
- **scopes** (*Sequence[str]*) – A list of OAuth2 scopes to attach to requests.
- **client_config** (*dict*) – A dictionary for call options for each method. See `google.gax.construct_settings()` for the structure of this data. Falls back to the default config if not specified or the specified config is missing data points.
- **lib_name** (*str*) – The API library software used for calling the service. (Unless you are writing an API client itself, leave this as default.)
- **lib_version** (*str*) – The API library software version used for calling the service. (Unless you are writing an API client itself, leave this as default.)
- **metrics_headers** (*dict*) – A dictionary of values for tracking client library metrics. Ultimately serializes to a string (e.g. 'foo/1.2.3 bar/3.14.1'). This argument should be considered private.

Returns: *LanguageServiceClient*

analyze_entities (*document*, *encoding_type=None*, *options=None*)

Finds named entities (currently proper names and common nouns) in the text along with entity types, salience, mentions for each entity, and other properties.

Example

```
>>> from google.cloud import language_v1beta2
>>>
>>> client = language_v1beta2.LanguageServiceClient()
>>>
>>> document = {}
>>>
>>> response = client.analyze_entities(document)
```

Parameters

- **document** (*Union[dict, Document]*) – Input document. If a dict is provided, it must be of the same form as the protobuf message *Document*
- **encoding_type** (*EncodingType*) – The encoding type used by the API to calculate offsets.
- **options** (*CallOptions*) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A *AnalyzeEntitiesResponse* instance.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

analyze_entity_sentiment (*document, encoding_type=None, options=None*)

Finds entities, similar to *AnalyzeEntities* in the text and analyzes sentiment associated with each entity and its mentions.

Example

```
>>> from google.cloud import language_v1beta2
>>>
>>> client = language_v1beta2.LanguageServiceClient()
>>>
>>> document = {}
>>>
>>> response = client.analyze_entity_sentiment(document)
```

Parameters

- **document** (*Union[dict, Document]*) – Input document. If a dict is provided, it must be of the same form as the protobuf message *Document*
- **encoding_type** (*EncodingType*) – The encoding type used by the API to calculate offsets.
- **options** (*CallOptions*) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A *AnalyzeEntitySentimentResponse* instance.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

analyze_sentiment (*document, encoding_type=None, options=None*)

Analyzes the sentiment of the provided text.

Example


```
>>> from google.cloud import language_v1beta2
>>>
>>> client = language_v1beta2.LanguageServiceClient()
>>>
>>> document = {}
>>>
>>> response = client.analyze_sentiment(document)
```

Parameters

- **document** (*Union[dict, Document]*) – Input document. If a dict is provided, it must be of the same form as the protobuf message *Document*
- **encoding_type** (*EncodingType*) – The encoding type used by the API to calculate sentence offsets for the sentence sentiment.
- **options** (*CallOptions*) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A *AnalyzeSentimentResponse* instance.

Raises

- *google.gax.errors.GaxError* if the RPC is aborted.
- *ValueError* if the parameters are invalid.

analyze_syntax (*document, encoding_type=None, options=None*)

Analyzes the syntax of the text and provides sentence boundaries and tokenization along with part of speech tags, dependency trees, and other properties.

Example

```
>>> from google.cloud import language_v1beta2
>>>
>>> client = language_v1beta2.LanguageServiceClient()
>>>
>>> document = {}
>>>
>>> response = client.analyze_syntax(document)
```

Parameters

- **document** (*Union[dict, Document]*) – Input document. If a dict is provided, it must be of the same form as the protobuf message *Document*
- **encoding_type** (*EncodingType*) – The encoding type used by the API to calculate offsets.
- **options** (*CallOptions*) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A *AnalyzeSyntaxResponse* instance.

Raises

- *google.gax.errors.GaxError* if the RPC is aborted.
- *ValueError* if the parameters are invalid.

annotate_text (*document, features, encoding_type=None, options=None*)

A convenience method that provides all syntax, sentiment, entity, and classification features in one call.

Example

```
>>> from google.cloud import language_v1beta2
>>>
>>> client = language_v1beta2.LanguageServiceClient()
>>>
>>> document = {}
>>> features = {}
>>>
>>> response = client.annotate_text(document, features)
```

Parameters

- **document** (*Union[dict, Document]*) – Input document. If a dict is provided, it must be of the same form as the protobuf message *Document*
- **features** (*Union[dict, Features]*) – The enabled features. If a dict is provided, it must be of the same form as the protobuf message *Features*
- **encoding_type** (*EncodingType*) – The encoding type used by the API to calculate offsets.
- **options** (*CallOptions*) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A *AnnotateTextResponse* instance.

Raises

- *google.gax.errors.GaxError* if the RPC is aborted.
- *ValueError* if the parameters are invalid.

classify_text (*document, options=None*)

Classifies a document into categories.

Example

```
>>> from google.cloud import language_v1beta2
>>>
>>> client = language_v1beta2.LanguageServiceClient()
>>>
>>> document = {}
>>>
>>> response = client.classify_text(document)
```

Parameters

- **document** (*Union[dict, Document]*) – Input document. If a dict is provided, it must be of the same form as the protobuf message *Document*
- **options** (*CallOptions*) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A *ClassifyTextResponse* instance.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

```
enums = <module 'google.cloud.language_v1beta2.gapic.enums' from '/home/docs/checkouts
```

9.6.4 Natural Language Beta Client Types

class `google.cloud.language_v1beta2.types.AnalyzeEntitiesRequest`

The entity analysis request message.

document

Input document.

encoding_type

The encoding type used by the API to calculate offsets.

class `google.cloud.language_v1beta2.types.AnalyzeEntitiesResponse`

The entity analysis response message.

entities

The recognized entities in the input document.

language

The language of the text, which will be the same as the language specified in the request or, if not specified, the automatically-detected language. See `[Document.language][google.cloud.language.v1beta2.Document.language]` field for more details.

class `google.cloud.language_v1beta2.types.AnalyzeEntitySentimentRequest`

The entity-level sentiment analysis request message.

document

Input document.

encoding_type

The encoding type used by the API to calculate offsets.

class `google.cloud.language_v1beta2.types.AnalyzeEntitySentimentResponse`

The entity-level sentiment analysis response message.

entities

The recognized entities in the input document with associated sentiments.

language

The language of the text, which will be the same as the language specified in the request or, if not specified, the automatically-detected language. See `[Document.language][google.cloud.language.v1beta2.Document.language]` field for more details.

class `google.cloud.language_v1beta2.types.AnalyzeSentimentRequest`

The sentiment analysis request message.

document

Input document.

encoding_type

The encoding type used by the API to calculate sentence offsets for the sentence sentiment.

class google.cloud.language_v1beta2.types.**AnalyzeSentimentResponse**

The sentiment analysis response message.

document_sentiment

The overall sentiment of the input document.

language

The language of the text, which will be the same as the language specified in the request or, if not specified, the automatically-detected language. See [Document.language][google.cloud.language.v1beta2.Document.language] field for more details.

sentences

The sentiment for all the sentences in the document.

class google.cloud.language_v1beta2.types.**AnalyzeSyntaxRequest**

The syntax analysis request message.

document

Input document.

encoding_type

The encoding type used by the API to calculate offsets.

class google.cloud.language_v1beta2.types.**AnalyzeSyntaxResponse**

The syntax analysis response message.

sentences

Sentences in the input document.

tokens

Tokens, along with their syntactic information, in the input document.

language

The language of the text, which will be the same as the language specified in the request or, if not specified, the automatically-detected language. See [Document.language][google.cloud.language.v1beta2.Document.language] field for more details.

class google.cloud.language_v1beta2.types.**AnnotateTextRequest**

The request message for the text annotation API, which can perform multiple analysis types (sentiment, entities, and syntax) in one call.

extract_syntax

Extract syntax information.

extract_entities

Extract entities.

extract_document_sentiment

Extract document-level sentiment.

extract_entity_sentiment

Extract entities and their associated sentiment.

classify_text

Classify the full document into categories.

document

Input document.

features

The enabled features.

encoding_type

The encoding type used by the API to calculate offsets.

class Features

All available features for sentiment, syntax, and semantic analysis. Setting each one to true will enable that specific analysis for the input.

class google.cloud.language_v1beta2.types.**AnnotateTextResponse**

The text annotations response message.

sentences

Sentences in the input document. Populated if the user enables [AnnotateTextRequest.Features.extract_syntax][google.cloud.language_v1beta2.AnnotateTextRequest.Features.extract_syntax].

tokens

Tokens, along with their syntactic information, in the input document. Populated if the user enables [AnnotateTextRequest.Features.extract_syntax][google.cloud.language_v1beta2.AnnotateTextRequest.Features.extract_syntax].

entities

Entities, along with their semantic information, in the input document. Populated if the user enables [AnnotateTextRequest.Features.extract_entities][google.cloud.language_v1beta2.AnnotateTextRequest.Features.extract_entities].

document_sentiment

The overall sentiment for the document. Populated if the user enables [AnnotateTextRequest.Features.extract_document_sentiment][google.cloud.language_v1beta2.AnnotateTextRequest.Features.extract_document_sentiment].

language

The language of the text, which will be the same as the language specified in the request or, if not specified, the automatically-detected language. See [Document.language][google.cloud.language_v1beta2.Document.language] field for more details.

categories

Categories identified in the input document.

class google.cloud.language_v1beta2.types.**ClassificationCategory**

Represents a category returned from the text classifier.

name

The name of the category representing the document.

confidence

The classifier's confidence of the category. Number represents how certain the classifier is that this category represents the given text.

class google.cloud.language_v1beta2.types.**ClassifyTextRequest**

The document classification request message.

document

Input document.

class google.cloud.language_v1beta2.types.**ClassifyTextResponse**

The document classification response message.

categories

Categories representing the input document.

class google.cloud.language_v1beta2.types.**DependencyEdge**

Represents dependency parse tree information for a token.

head_token_index

Represents the head of this token in the dependency tree. This is the index of the token which has an arc going to this token. The index is the position of the token in the array of tokens returned by the API method. If this token is a root token, then the `head_token_index` is its own index.

label

The parse label for the token.

class `google.cloud.language_v1beta2.types.Document`

Represents the input to API methods.

type

Required. If the type is not set or is `TYPE_UNSPECIFIED`, returns an `INVALID_ARGUMENT` error.

source

The source of the document: a string containing the content or a Google Cloud Storage URI.

content

The content of the input in string format.

gcs_content_uri

The Google Cloud Storage URI where the file content is located. This URI must be of the form: `gs://bucket_name/object_name`. For more details, see <https://cloud.google.com/storage/docs/reference-uris>. NOTE: Cloud Storage object versioning is not supported.

language

The language of the document (if not specified, the language is automatically detected). Both ISO and BCP-47 language codes are accepted. [Language Support](#) lists currently supported languages for each API method. If the language (either specified by the caller or automatically detected) is not supported by the called API method, an `INVALID_ARGUMENT` error is returned.

class `google.cloud.language_v1beta2.types.Entity`

Represents a phrase in the text that is a known entity, such as a person, an organization, or location. The API associates information, such as salience and mentions, with entities.

name

The representative name for the entity.

type

The entity type.

metadata

Metadata associated with the entity. Currently, Wikipedia URLs and Knowledge Graph MIDs are provided, if available. The associated keys are `"wikipedia_url"` and `"mid"`, respectively.

salience

The salience score associated with the entity in the `[0, 1.0]` range. The salience score for an entity provides information about the importance or centrality of that entity to the entire document text. Scores closer to 0 are less salient, while scores closer to 1.0 are highly salient.

mentions

The mentions of this entity in the input document. The API currently supports proper noun mentions.

sentiment

For calls to `[AnalyzeEntitySentiment][]` or if `[AnnotateTextRequest.Features.extract_entity_sentiment][google.cloud.language_v1beta2.AnnotateTextRequest.Features.extract_entity_sentiment]` is set to true, this field will contain the aggregate sentiment expressed for this entity in the provided document.

class `google.cloud.language_v1beta2.types.EntityMention`

Represents a mention for an entity in the text. Currently, proper noun mentions are supported.

text

The mention text.

type

The type of the entity mention.

sentiment

For calls to `[AnalyzeEntitySentiment][]` or if `[AnnotateTextRequest.Features.extract_entity_sentiment][google.cloud.language.v1beta2.AnnotateTextRequest.Features.extract_entity_sentiment]` is set to true, this field will contain the sentiment expressed for this mention of the entity in the provided document.

class `google.cloud.language_v1beta2.types.PartOfSpeech`

Represents part of speech information for a token.

tag

The part of speech tag.

aspect

The grammatical aspect.

case

The grammatical case.

form

The grammatical form.

gender

The grammatical gender.

mood

The grammatical mood.

number

The grammatical number.

person

The grammatical person.

proper

The grammatical properness.

reciprocity

The grammatical reciprocity.

tense

The grammatical tense.

voice

The grammatical voice.

class `google.cloud.language_v1beta2.types.Sentence`

Represents a sentence in the input document.

text

The sentence text.

sentiment

For calls to `[AnalyzeSentiment][]` or if `[AnnotateTextRequest.Features.extract_document_sentiment][google.cloud.language.v1beta2.AnnotateTextRequest.Features.extract_document_sentiment]` is set to true, this field will contain the sentiment for the sentence.

class google.cloud.language_v1beta2.types.Sentiment

Represents the feeling associated with the entire text or entities in the text.

magnitude

A non-negative number in the [0, +inf) range, which represents the absolute magnitude of sentiment regardless of score (positive or negative).

score

Sentiment score between -1.0 (negative sentiment) and 1.0 (positive sentiment).

class google.cloud.language_v1beta2.types.TextSpan

Represents an output piece of text.

content

The content of the output text.

begin_offset

The API calculates the beginning offset of the content in the original document according to the [EncodingType][google.cloud.language.v1beta2.EncodingType] specified in the API request.

class google.cloud.language_v1beta2.types.Token

Represents the smallest syntactic building block of the text.

text

The token text.

part_of_speech

Parts of speech tag for this token.

dependency_edge

Dependency tree parse for this token.

lemma

[Lemma](#) of the token.

Note: The client for the beta API is provided on a provisional basis. The API surface is subject to change, and it is possible that this client will be deprecated or removed after its features become GA.

Google Cloud Pub/Sub is a fully-managed real-time messaging service that allows you to send and receive messages between independent applications. You can leverage Cloud Pub/Sub's flexibility to decouple systems and components hosted on Google Cloud Platform or elsewhere on the Internet. By building on the same technology Google uses, Cloud Pub/Sub is designed to provide “at least once” delivery at low latency with on-demand scalability to 1 million messages per second (and beyond).

10.1 Authentication and Configuration

- For an overview of authentication in `google-cloud-python`, see [Authentication](#).
- In addition to any authentication configuration, you should also set the `GOOGLE_CLOUD_PROJECT` environment variable for the project you'd like to interact with. If the `GOOGLE_CLOUD_PROJECT` environment variable is not present, the project ID from JSON file credentials is used.

If you are using Google App Engine or Google Compute Engine this will be detected automatically.

- After configuring your environment, create a `PublisherClient` or `SubscriberClient`.

```
>>> from google.cloud import pubsub
>>> publisher = pubsub.PublisherClient()
>>> subscriber = pubsub.SubscriberClient()
```

or pass in credentials explicitly.

```
>>> from google.cloud import pubsub
>>> client = pubsub.PublisherClient(
...     credentials=creds,
... )
```

10.2 Publishing

To publish data to Cloud Pub/Sub you must create a topic, and then publish messages to it

```
>>> import os
>>> from google.cloud import pubsub
>>>
>>> publisher = pubsub.PublisherClient()
>>> topic = 'projects/{project_id}/topics/{topic}'.format(
...     project_id=os.getenv('GOOGLE_CLOUD_PROJECT'),
...     topic='MY_TOPIC_NAME', # Set this to something appropriate.
... )
>>> publisher.create_topic()
>>> publisher.publish(topic, b'My first message!', spam='eggs')
```

To learn more, consult the *[publishing documentation](#)*.

10.3 Subscribing

To subscribe to data in Cloud Pub/Sub, you create a subscription based on the topic, and subscribe to that.

```
>>> import os
>>> from google.cloud import pubsub
>>>
>>> subscriber = pubsub.SubscriberClient()
>>> topic = 'projects/{project_id}/topics/{topic}'.format(
...     project_id=os.getenv('GOOGLE_CLOUD_PROJECT'),
...     topic='MY_TOPIC_NAME', # Set this to something appropriate.
... )
>>> subscription_name = 'projects/{project_id}/subscriptions/{sub}'.format(
...     project_id=os.getenv('GOOGLE_CLOUD_PROJECT'),
...     sub='MY_SUBSCRIPTION_NAME', # Set this to something appropriate.
... )
>>> subscription = subscriber.create_subscription(subscription_name, topic)
```

The subscription is opened asynchronously, and messages are processed by use of a callback.

```
>>> def callback(message):
...     print(message.data)
...     message.ack()
>>> subscription.open(callback)
```

To learn more, consult the *[subscriber documentation](#)*.

10.4 Learn More

10.4.1 Publishing Messages

Publishing messages is handled through the *[Client](#)* class (aliased as `google.cloud.pubsub.PublisherClient`). This class provides methods to create topics, and (most importantly) a *[publish\(\)](#)* method that publishes messages to Pub/Sub.

Instantiating a publishing client is straightforward:

```
from google.cloud import pubsub
publish_client = pubsub.PublisherClient()
```

Publish a Message

To publish a message, use the `publish()` method. This method accepts two positional arguments: the topic to publish to, and the body of the message. It also accepts arbitrary keyword arguments, which are passed along as attributes of the message.

The topic is passed along as a string; all topics have the canonical form of `projects/{project_name}/topics/{topic_name}`.

Therefore, a very basic publishing call looks like:

```
topic = 'projects/{project}/topics/{topic}'
publish_client.publish(topic, b'This is my message.')
```

Note: The message data in Pub/Sub is an opaque blob of bytes, and as such, you *must* send a bytes object in Python 3 (str object in Python 2). If you send a text string (str in Python 3, unicode in Python 2), the method will raise `TypeError`.

The reason it works this way is because there is no reasonable guarantee that the same language or environment is being used by the subscriber, and so it is the responsibility of the publisher to properly encode the payload.

If you want to include attributes, simply add keyword arguments:

```
topic = 'projects/{project}/topics/{topic}'
publish_client.publish(topic, b'This is my message.', foo='bar')
```

Batching

Whenever you publish a message, a Batch is automatically created. This way, if you publish a large volume of messages, it reduces the number of requests made to the server.

The way that this works is that on the first message that you send, a new Batch is created automatically. For every subsequent message, if there is already a valid batch that is still accepting messages, then that batch is used. When the batch is created, it begins a countdown that publishes the batch once sufficient time has elapsed (by default, this is 0.05 seconds).

If you need different batching settings, simply provide a `BatchSettings` object when you instantiate the `Client`:

```
from google.cloud import pubsub
from google.cloud.pubsub import types

client = pubsub.PublisherClient(
    batch_settings=BatchSettings(max_messages=500),
)
```

Pub/Sub accepts a maximum of 1,000 messages in a batch, and the size of a batch can not exceed 10 megabytes.

Futures

Every call to `publish()` will return a class that conforms to the `Future` interface. You can use this to ensure that the publish succeeded:

```
# The .result() method will block until the future is complete.
# If there is an error, it will raise an exception.
future = client.publish(topic, b'My awesome message.')
message_id = future.result()
```

You can also attach a callback to the future:

```
# Callbacks receive the future as their only argument, as defined in
# the Future interface.
def callback(future):
    message_id = future.result()
    do_something_with(message_id)

# The callback is added once you get the future. If you add a callback
# and the future is already done, it will simply be executed immediately.
future = client.publish(topic, b'My awesome message.')
future.add_done_callback(callback)
```

API Reference

Publisher Client API

```
class google.cloud.pubsub_v1.publisher.client.Client (batch_settings=(),
                                                    batch_class=<class
                                                    'google.cloud.pubsub_v1.publisher.batch.thread.Batch'>,
                                                    **kwargs)
```

A publisher client for Google Cloud Pub/Sub.

This creates an object that is capable of publishing messages. Generally, you can instantiate this client with no arguments, and you get sensible defaults.

Parameters

- **batch_settings** (`BatchSettings`) – The settings for batch publishing.
- **batch_class** (`class`) – A class that describes how to handle batches. You may subclass the `pubsub_v1.publisher.batch.base.BaseBatch` class in order to define your own batcher. This is primarily provided to allow use of different concurrency models; the default is based on `threading.Thread`.
- **kwargs** (`dict`) – Any additional arguments provided are sent as keyword arguments to the underlying `PublisherClient`. Generally, you should not need to set additional keyword arguments.

batch (`topic`, `message`, `create=True`, `autocommit=True`)

Return the current batch for the provided topic.

This will create a new batch only if no batch currently exists.

Parameters

- **topic** (`str`) – A string representing the topic.
- **message** (`PubsubMessage`) – The message that will be committed.

- **create** (*bool*) – Whether to create a new batch if no batch is found. Defaults to True.
- **autocommit** (*bool*) – Whether to autocommit this batch. This is primarily useful for debugging.

Returns The batch object.

Return type Batch

create_topic (*a, **kw)

Creates the given topic with the given name.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import publisher_client
>>> client = publisher_client.PublisherClient()
>>> name = client.topic_path('[PROJECT]', '[TOPIC]')
>>> response = client.create_topic(name)
```

Parameters

- **name** (*string*) – The name of the topic. It must have the format "projects/{project}/topics/{topic}". {topic} must start with a letter, and contain only letters ([A-Za-z]), numbers ([0-9]), dashes (-), underscores (_), periods (.), tildes (~), plus (+) or percent signs (%). It must be between 3 and 255 characters in length, and it must not start with "goog".
- **options** (*google.gax.CallOptions*) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A `google.cloud.proto.pubsub.v1.pubsub_pb2.Topic` instance.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

delete_topic (*a, **kw)

Deletes the topic with the given name. Returns `NOT_FOUND` if the topic does not exist. After a topic is deleted, a new topic may be created with the same name; this is an entirely new topic with none of the old configuration or subscriptions. Existing subscriptions to this topic are not deleted, but their `topic` field is set to `_deleted-topic_`.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import publisher_client
>>> client = publisher_client.PublisherClient()
>>> topic = client.topic_path('[PROJECT]', '[TOPIC]')
>>> client.delete_topic(topic)
```

Parameters

- **topic** (*string*) – Name of the topic to delete. Format is `projects/{project}/topics/{topic}`.

- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries etc.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

get_iam_policy (*a, **kw)

Gets the access control policy for a resource. Returns an empty policy if the resource exists and does not have a policy set.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import publisher_client
>>> client = publisher_client.PublisherClient()
>>> resource = client.topic_path('[PROJECT]', '[TOPIC]')
>>> response = client.get_iam_policy(resource)
```

Parameters

- **resource** (*string*) – REQUIRED: The resource for which the policy is being requested. `resource` is usually specified as a path. For example, a Project resource is specified as `projects/{project}`.
- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A `google.iam.v1.policy_pb2.Policy` instance.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

get_topic (*a, **kw)

Gets the configuration of a topic.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import publisher_client
>>> client = publisher_client.PublisherClient()
>>> topic = client.topic_path('[PROJECT]', '[TOPIC]')
>>> response = client.get_topic(topic)
```

Parameters

- **topic** (*string*) – The name of the topic to get. Format is `projects/{project}/topics/{topic}`.
- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A `google.cloud.proto.pubsub.v1.pubsub_pb2.Topic` instance.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

list_topic_subscriptions (*a, **kw)

Lists the name of the subscriptions for this topic.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import publisher_client
>>> from google.gax import CallOptions, INITIAL_PAGE
>>> client = publisher_client.PublisherClient()
>>> topic = client.topic_path('[PROJECT]', '[TOPIC]')
>>>
>>> # Iterate over all results
>>> for element in client.list_topic_subscriptions(topic):
>>>     # process element
>>>     pass
>>>
>>> # Or iterate over results one page at a time
>>> for page in client.list_topic_subscriptions(topic,
↳ options=CallOptions(page_token=INITIAL_PAGE)):
>>>     for element in page:
>>>         # process element
>>>         pass
```

Parameters

- **topic** (*string*) – The name of the topic that subscriptions are attached to. Format is `projects/{project}/topics/{topic}`.
- **page_size** (*int*) – The maximum number of resources contained in the underlying API response. If page streaming is performed per-resource, this parameter does not affect the return value. If page streaming is performed per-page, this determines the maximum number of resources in a page.
- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A `google.gax.PageIterator` instance. By default, this is an iterable of string instances. This object can also be configured to iterate over the pages of the response through the `CallOptions` parameter.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

list_topics (*a, **kw)

Lists matching topics.

Example

```

>>> from google.cloud.gapic.pubsub.v1 import publisher_client
>>> from google.gax import CallOptions, INITIAL_PAGE
>>> client = publisher_client.PublisherClient()
>>> project = client.project_path('[PROJECT]')
>>>
>>> # Iterate over all results
>>> for element in client.list_topics(project):
>>>     # process element
>>>     pass
>>>
>>> # Or iterate over results one page at a time
>>> for page in client.list_topics(project, options=CallOptions(page_
↳ token=INITIAL_PAGE)):
>>>     for element in page:
>>>         # process element
>>>         pass

```

Parameters

- **project** (*string*) – The name of the cloud project that topics belong to. Format is `projects/{project}`.
- **page_size** (*int*) – The maximum number of resources contained in the underlying API response. If page streaming is performed per-resource, this parameter does not affect the return value. If page streaming is performed per-page, this determines the maximum number of resources in a page.
- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g. timeout, retries etc.

Returns A `google.gax.PageIterator` instance. By default, this is an iterable of `google.cloud.proto.pubsub.v1.pubsub_pb2.Topic` instances. This object can also be configured to iterate over the pages of the response through the *CallOptions* parameter.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

match_project_from_project_name (**a, **kw*)

Parses the project from a project resource.

Parameters **project_name** (*string*) – A fully-qualified path representing a project resource.

Returns A string representing the project.

match_project_from_topic_name (**a, **kw*)

Parses the project from a topic resource.

Parameters **topic_name** (*string*) – A fully-qualified path representing a topic resource.

Returns A string representing the project.

match_topic_from_topic_name (**a, **kw*)

Parses the topic from a topic resource.

Parameters **topic_name** (*string*) – A fully-qualified path representing a topic resource.

Returns A string representing the topic.

project_path (*a, **kw)

Returns a fully-qualified project resource name string.

publish (topic, data, **attrs)

Publish a single message.

Note: Messages in Pub/Sub are blobs of bytes. They are *binary* data, not text. You must send data as a bytestring (bytes in Python 3; str in Python 2), and this library will raise an exception if you send a text string.

The reason that this is so important (and why we do not try to coerce for you) is because Pub/Sub is also platform independent and there is no way to know how to decode messages properly on the other side; therefore, encoding and decoding is a required exercise for the developer.

Add the given message to this object; this will cause it to be published once the batch either has enough messages or a sufficient period of time has elapsed.

Example

```
>>> from google.cloud.pubsub_v1 import publisher_client
>>> client = publisher_client.PublisherClient()
>>> topic = client.topic_path('[PROJECT]', '[TOPIC]')
>>> data = b'The rain in Wales falls mainly on the snails.'
>>> response = client.publish(topic, data, username='guido')
```

Parameters

- **topic** (*str*) – The topic to publish messages to.
- **data** (*bytes*) – A bytestring representing the message body. This must be a bytestring.
- **attrs** (*Mapping[str, str]*) – A dictionary of attributes to be sent as metadata. (These may be text strings or byte strings.)

Returns An object conforming to the `concurrent.futures.Future` interface.

Return type `Future`

set_iam_policy (*a, **kw)

Sets the access control policy on the specified resource. Replaces any existing policy.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import publisher_client
>>> from google.iam.v1 import policy_pb2
>>> client = publisher_client.PublisherClient()
>>> resource = client.topic_path('[PROJECT]', '[TOPIC]')
>>> policy = policy_pb2.Policy()
>>> response = client.set_iam_policy(resource, policy)
```

Parameters

- **resource** (*string*) – REQUIRED: The resource for which the policy is being specified. `resource` is usually specified as a path. For example, a Project resource is specified as `projects/{project}`.
- **policy** (`google.iam.v1.policy_pb2.Policy`) – REQUIRED: The complete policy to be applied to the resource. The size of the policy is limited to a few 10s of KB. An empty policy is a valid policy but certain Cloud Platform services (such as Projects) might reject them.
- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A `google.iam.v1.policy_pb2.Policy` instance.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

test_iam_permissions (*a, **kw)

Returns permissions that a caller has on the specified resource. If the resource does not exist, this will return an empty set of permissions, not a NOT_FOUND error.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import publisher_client
>>> client = publisher_client.PublisherClient()
>>> resource = client.topic_path('[PROJECT]', '[TOPIC]')
>>> permissions = []
>>> response = client.test_iam_permissions(resource, permissions)
```

Parameters

- **resource** (*string*) – REQUIRED: The resource for which the policy detail is being requested. `resource` is usually specified as a path. For example, a Project resource is specified as `projects/{project}`.
- **permissions** (*list[string]*) – The set of permissions to check for the resource. Permissions with wildcards (such as '*' or 'storage.*') are not allowed. For more information see [IAM Overview](#).
- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A `google.iam.v1.iam_policy_pb2.TestIamPermissionsResponse` instance.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

topic_path (*a, **kw)

Returns a fully-qualified topic resource name string.

10.4.2 Subscribing to Messages

Subscribing to messages is handled through the `Client` class (aliased as `google.cloud.pubsub.SubscriberClient`). This class provides a `subscribe()` method to attach to subscriptions on existing topics, and (most importantly) a `open()` method that consumes messages from Pub/Sub.

Instantiating a subscriber client is straightforward:

```
from google.cloud import pubsub
subscriber = pubsub.SubscriberClient()
```

Creating a Subscription

In Pub/Sub, a **subscription** is a discrete pull of messages from a topic. If multiple clients pull the same subscription, then messages are split between them. If multiple clients create a subscription each, then each client will get every message.

Note: Remember that Pub/Sub operates under the principle of “everything at least once”. Even in the case where multiple clients pull the same subscription, *some* redundancy is likely.

Creating a subscription requires that you already know what topic you want to subscribe to, and it must already exist. Once you have that, it is easy:

```
# Substitute {project}, {topic}, and {subscription} with appropriate
# values for your application.
topic_name = 'projects/{project}/topics/{topic}'
sub_name = 'projects/{project}/subscriptions/{subscription}'
subscriber.create_subscription(topic_name, sub_name)
```

Pulling a Subscription

Once you have created a subscription (or if you already had one), the next step is to pull data from it. This entails two steps: first you must call `subscribe()`, passing in the subscription string.

```
# As before, substitute {project} and {subscription} with appropriate
# values for your application.
subscription = subscriber.subscribe(
    'projects/{project}/subscriptions/{subscription}',
)
```

This will return an object with an `open()` method; calling this method will actually begin consumption of the subscription.

Subscription Callbacks

Because subscriptions in this Pub/Sub client are opened asynchronously, processing the messages that are yielded by the subscription is handled through **callbacks**.

The basic idea: Define a function that takes one argument; this argument will be a `Message` instance. This function should do whatever processing is necessary. At the end, the function should `ack()` the message.

When you call `open()`, you must pass the callback that will be used.

Here is an example:

```
# Define the callback.
# Note that the callback is defined before the subscription is opened.
def callback(message):
    do_something_with(message) # Replace this with your actual logic.
    message.ack()

# Open the subscription, passing the callback.
subscription.open(callback)
```

Explaining Ack

In Pub/Sub, the term **ack** stands for “acknowledge”. You should ack a message when your processing of that message *has completed*. When you ack a message, you are telling Pub/Sub that you do not need to see it again.

It might be tempting to ack messages immediately on receipt. While there are valid use cases for this, in general it is unwise. The reason why: If there is some error or edge case in your processing logic, and processing of the message fails, you will have already told Pub/Sub that you successfully processed the message. By contrast, if you ack only upon completion, then Pub/Sub will eventually re-deliver the unacknowledged message.

It is also possible to **nack** a message, which is the opposite. When you nack, it tells Pub/Sub that you are unable or unwilling to deal with the message, and that the service should redeliver it.

API Reference

Subscriber Client API

```
class google.cloud.pubsub_v1.subscriber.client.Client (policy_class=<class
                                                         'google.cloud.pubsub_v1.subscriber.policy.thread.Poli
                                                         **kwargs)
```

A subscriber client for Google Cloud Pub/Sub.

This creates an object that is capable of subscribing to messages. Generally, you can instantiate this client with no arguments, and you get sensible defaults.

Parameters

- **policy_class** (*class*) – A class that describes how to handle subscriptions. You may subclass the `pubsub_v1.subscriber.policy.base.BasePolicy` class in order to define your own consumer. This is primarily provided to allow use of different concurrency models; the default is based on `threading.Thread`.
- **kwargs** (*dict*) – Any additional arguments provided are sent as keyword keyword arguments to the underlying `SubscriberClient`. Generally, you should not need to set additional keyword arguments.

acknowledge (*a, **kw)

Acknowledges the messages associated with the `ack_ids` in the `AcknowledgeRequest`. The Pub/Sub system can remove the relevant messages from the subscription.

Acknowledging a message whose ack deadline has expired may succeed, but such a message may be redelivered later. Acknowledging a message more than once will not result in an error.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import subscriber_client
>>> client = subscriber_client.SubscriberClient()
>>> subscription = client.subscription_path('[PROJECT]', '[SUBSCRIPTION]')
>>> ack_ids = []
>>> client.acknowledge(subscription, ack_ids)
```

Parameters

- **subscription** (*string*) – The subscription whose message is being acknowledged. Format is `projects/{project}/subscriptions/{sub}`.
- **ack_ids** (*list[string]*) – The acknowledgment ID for the messages being acknowledged that was returned by the Pub/Sub system in the Pull response. Must not be empty.
- **options** (*google.gax.CallOptions*) – Overrides the default settings for this call, e.g. timeout, retries etc.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

`create_snapshot` (*a, **kw)

Creates a snapshot from the requested subscription. If the snapshot already exists, returns `ALREADY_EXISTS`. If the requested subscription doesn't exist, returns `NOT_FOUND`.

If the name is not provided in the request, the server will assign a random name for this snapshot on the same project as the subscription, conforming to the [resource name format](#). The generated name is populated in the returned Snapshot object. Note that for REST API requests, you must specify a name in the request.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import subscriber_client
>>> client = subscriber_client.SubscriberClient()
>>> name = client.snapshot_path('[PROJECT]', '[SNAPSHOT]')
>>> subscription = client.subscription_path('[PROJECT]', '[SUBSCRIPTION]')
>>> response = client.create_snapshot(name, subscription)
```

Parameters

- **name** (*string*) – Optional user-provided name for this snapshot. If the name is not provided in the request, the server will assign a random name for this snapshot on the same project as the subscription. Note that for REST API requests, you must specify a name. Format is `projects/{project}/snapshots/{snap}`.
- **subscription** (*string*) – The subscription whose backlog the snapshot retains. Specifically, the created snapshot is guaranteed to retain:
 - The existing backlog on the subscription. More precisely, this is defined as the messages in the subscription's backlog that are unacknowledged upon the successful completion of the `CreateSnapshot` request; as well as:

- Any messages published to the subscription’s topic following the successful completion of the CreateSnapshot request.

Format is `projects/{project}/subscriptions/{sub}`.

- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A `google.cloud.proto.pubsub.v1.pubsub_pb2.Snapshot` instance.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

create_subscription (*a, **kw)

Creates a subscription to a given topic. If the subscription already exists, returns `ALREADY_EXISTS`. If the corresponding topic doesn’t exist, returns `NOT_FOUND`.

If the name is not provided in the request, the server will assign a random name for this subscription on the same project as the topic, conforming to the [resource name format](#). The generated name is populated in the returned Subscription object. Note that for REST API requests, you must specify a name in the request.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import subscriber_client
>>> client = subscriber_client.SubscriberClient()
>>> name = client.subscription_path('[PROJECT]', '[SUBSCRIPTION]')
>>> topic = client.topic_path('[PROJECT]', '[TOPIC]')
>>> response = client.create_subscription(name, topic)
```

Parameters

- **name** (*string*) – The name of the subscription. It must have the format `"projects/{project}/subscriptions/{subscription}"`. {subscription} must start with a letter, and contain only letters (`[A-Za-z]`), numbers (`[0-9]`), dashes (`-`), underscores (`_`), periods (`.`), tildes (`~`), plus (`+`) or percent signs (`%`). It must be between 3 and 255 characters in length, and it must not start with `"goog"`.
- **topic** (*string*) – The name of the topic from which this subscription is receiving messages. Format is `projects/{project}/topics/{topic}`. The value of this field will be `_deleted-topic_` if the topic has been deleted.
- **push_config** (`google.cloud.proto.pubsub.v1.pubsub_pb2.PushConfig`) – If push delivery is used with this subscription, this field is used to configure it. An empty `pushConfig` signifies that the subscriber will pull and ack messages using API methods.
- **ack_deadline_seconds** (*int*) – This value is the maximum time after a subscriber receives a message before the subscriber should acknowledge the message. After message delivery but before the ack deadline expires and before the message is acknowledged, it is an outstanding message and will not be delivered again during that time (on a best-effort basis).

For pull subscriptions, this value is used as the initial value for the ack deadline. To override this value for a given message, call `ModifyAckDeadline` with the corresponding `ack_id` if using pull. The minimum custom deadline you can specify is 10 seconds. The

maximum custom deadline you can specify is 600 seconds (10 minutes). If this parameter is 0, a default value of 10 seconds is used.

For push delivery, this value is also used to set the request timeout for the call to the push endpoint.

If the subscriber never acknowledges the message, the Pub/Sub system will eventually redeliver the message.

- **retain_acked_messages** (*bool*) – Indicates whether to retain acknowledged messages. If true, then messages are not expunged from the subscription's backlog, even if they are acknowledged, until they fall out of the `message_retention_duration` window.
- **message_retention_duration** (`google.protobuf.duration_pb2.Duration`) – How long to retain unacknowledged messages in the subscription's backlog, from the moment a message is published. If `retain_acked_messages` is true, then this also configures the retention of acknowledged messages, and thus configures how far back in time a `Seek` can be done. Defaults to 7 days. Cannot be more than 7 days or less than 10 minutes.
- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A `google.cloud.proto.pubsub.v1.pubsub_pb2.Subscription` instance.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

`delete_snapshot (*a, **kw)`

Removes an existing snapshot. All messages retained in the snapshot are immediately dropped. After a snapshot is deleted, a new one may be created with the same name, but the new one has no association with the old snapshot or its subscription, unless the same subscription is specified.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import subscriber_client
>>> client = subscriber_client.SubscriberClient()
>>> snapshot = client.snapshot_path('[PROJECT]', '[SNAPSHOT]')
>>> client.delete_snapshot(snapshot)
```

Parameters

- **snapshot** (*string*) – The name of the snapshot to delete. Format is `projects/{project}/snapshots/{snap}`.
- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries etc.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

delete_subscription (*a, **kw)

Deletes an existing subscription. All messages retained in the subscription are immediately dropped. Calls to `Pull` after deletion will return `NOT_FOUND`. After a subscription is deleted, a new one may be created with the same name, but the new one has no association with the old subscription or its topic unless the same topic is specified.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import subscriber_client
>>> client = subscriber_client.SubscriberClient()
>>> subscription = client.subscription_path('[PROJECT]', '[SUBSCRIPTION]')
>>> client.delete_subscription(subscription)
```

Parameters

- **subscription** (*string*) – The subscription to delete. Format is `projects/{project}/subscriptions/{sub}`.
- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries etc.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

get_iam_policy (*a, **kw)

Gets the access control policy for a resource. Returns an empty policy if the resource exists and does not have a policy set.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import subscriber_client
>>> client = subscriber_client.SubscriberClient()
>>> resource = client.subscription_path('[PROJECT]', '[SUBSCRIPTION]')
>>> response = client.get_iam_policy(resource)
```

Parameters

- **resource** (*string*) – REQUIRED: The resource for which the policy is being requested. `resource` is usually specified as a path. For example, a Project resource is specified as `projects/{project}`.
- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A `google.iam.v1.policy_pb2.Policy` instance.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

get_subscription (*a, **kw)

Gets the configuration details of a subscription.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import subscriber_client
>>> client = subscriber_client.SubscriberClient()
>>> subscription = client.subscription_path('[PROJECT]', '[SUBSCRIPTION]')
>>> response = client.get_subscription(subscription)
```

Parameters

- **subscription** (*string*) – The name of the subscription to get. Format is projects/{project}/subscriptions/{sub}.
- **options** (*google.gax.CallOptions*) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A `google.cloud.proto.pubsub.v1.pubsub_pb2.Subscription` instance.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

list_snapshots (*a, **kw)
Lists the existing snapshots.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import subscriber_client
>>> from google.gax import CallOptions, INITIAL_PAGE
>>> client = subscriber_client.SubscriberClient()
>>> project = client.project_path('[PROJECT]')
>>>
>>> # Iterate over all results
>>> for element in client.list_snapshots(project):
>>>     # process element
>>>     pass
>>>
>>> # Or iterate over results one page at a time
>>> for page in client.list_snapshots(project, options=CallOptions(page_
↪ token=INITIAL_PAGE)):
>>>     for element in page:
>>>         # process element
>>>         pass
```

Parameters

- **project** (*string*) – The name of the cloud project that snapshots belong to. Format is projects/{project}.
- **page_size** (*int*) – The maximum number of resources contained in the underlying API response. If page streaming is performed per- resource, this parameter does not affect the return value. If page streaming is performed per-page, this determines the maximum number of resources in a page.

- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g. timeout, retries etc.

Returns A `google.gax.PageIterator` instance. By default, this is an iterable of `google.cloud.proto.pubsub.v1.pubsub_pb2.Snapshot` instances. This object can also be configured to iterate over the pages of the response through the *CallOptions* parameter.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

list_subscriptions (*a, **kw)

Lists matching subscriptions.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import subscriber_client
>>> from google.gax import CallOptions, INITIAL_PAGE
>>> client = subscriber_client.SubscriberClient()
>>> project = client.project_path('[PROJECT]')
>>>
>>> # Iterate over all results
>>> for element in client.list_subscriptions(project):
>>>     # process element
>>>     pass
>>>
>>> # Or iterate over results one page at a time
>>> for page in client.list_subscriptions(project, options=CallOptions(page_
↵token=INITIAL_PAGE)):
>>>     for element in page:
>>>         # process element
>>>         pass
```

Parameters

- **project** (*string*) – The name of the cloud project that subscriptions belong to. Format is `projects/{project}`.
- **page_size** (*int*) – The maximum number of resources contained in the underlying API response. If page streaming is performed per-resource, this parameter does not affect the return value. If page streaming is performed per-page, this determines the maximum number of resources in a page.
- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g. timeout, retries etc.

Returns A `google.gax.PageIterator` instance. By default, this is an iterable of `google.cloud.proto.pubsub.v1.pubsub_pb2.Subscription` instances. This object can also be configured to iterate over the pages of the response through the *CallOptions* parameter.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

match_project_from_project_name (*a, **kw)

Parses the project from a project resource.

Parameters **project_name** (*string*) – A fully-qualified path representing a project resource.

Returns A string representing the project.

match_project_from_snapshot_name (*a, **kw)

Parses the project from a snapshot resource.

Parameters **snapshot_name** (*string*) – A fully-qualified path representing a snapshot resource.

Returns A string representing the project.

match_project_from_subscription_name (*a, **kw)

Parses the project from a subscription resource.

Parameters **subscription_name** (*string*) – A fully-qualified path representing a subscription resource.

Returns A string representing the project.

match_project_from_topic_name (*a, **kw)

Parses the project from a topic resource.

Parameters **topic_name** (*string*) – A fully-qualified path representing a topic resource.

Returns A string representing the project.

match_snapshot_from_snapshot_name (*a, **kw)

Parses the snapshot from a snapshot resource.

Parameters **snapshot_name** (*string*) – A fully-qualified path representing a snapshot resource.

Returns A string representing the snapshot.

match_subscription_from_subscription_name (*a, **kw)

Parses the subscription from a subscription resource.

Parameters **subscription_name** (*string*) – A fully-qualified path representing a subscription resource.

Returns A string representing the subscription.

match_topic_from_topic_name (*a, **kw)

Parses the topic from a topic resource.

Parameters **topic_name** (*string*) – A fully-qualified path representing a topic resource.

Returns A string representing the topic.

modify_ack_deadline (*a, **kw)

Modifies the ack deadline for a specific message. This method is useful to indicate that more time is needed to process a message by the subscriber, or to make the message available for redelivery if the processing was interrupted. Note that this does not modify the subscription-level `ackDeadlineSeconds` used for subsequent messages.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import subscriber_client
>>> client = subscriber_client.SubscriberClient()
>>> subscription = client.subscription_path('[PROJECT]', '[SUBSCRIPTION]')
>>> ack_ids = []
>>> ack_deadline_seconds = 0
>>> client.modify_ack_deadline(subscription, ack_ids, ack_deadline_seconds)
```

Parameters

- **subscription** (*string*) – The name of the subscription. Format is `projects/{project}/subscriptions/{sub}`.
- **ack_ids** (*list[string]*) – List of acknowledgment IDs.
- **ack_deadline_seconds** (*int*) – The new ack deadline with respect to the time this request was sent to the Pub/Sub system. For example, if the value is 10, the new ack deadline will expire 10 seconds after the `ModifyAckDeadline` call was made. Specifying zero may immediately make the message available for another pull request. The minimum deadline you can specify is 0 seconds. The maximum deadline you can specify is 600 seconds (10 minutes).
- **options** (*google.gax.CallOptions*) – Overrides the default settings for this call, e.g. timeout, retries etc.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

modify_push_config (*a, **kw)

Modifies the `PushConfig` for a specified subscription.

This may be used to change a push subscription to a pull one (signified by an empty `PushConfig`) or vice versa, or change the endpoint URL and other attributes of a push subscription. Messages will accumulate for delivery continuously through the call regardless of changes to the `PushConfig`.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import subscriber_client
>>> from google.cloud.proto.pubsub.v1 import pubsub_pb2
>>> client = subscriber_client.SubscriberClient()
>>> subscription = client.subscription_path('[PROJECT]', '[SUBSCRIPTION]')
>>> push_config = pubsub_pb2.PushConfig()
>>> client.modify_push_config(subscription, push_config)
```

Parameters

- **subscription** (*string*) – The name of the subscription. Format is `projects/{project}/subscriptions/{sub}`.
- **push_config** (*google.cloud.proto.pubsub.v1.pubsub_pb2.PushConfig*) – The push configuration for future deliveries.

An empty `pushConfig` indicates that the Pub/Sub system should stop pushing messages from the given subscription and allow messages to be pulled and acknowledged - effectively pausing the subscription if `Pull` is not called.

- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries etc.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

project_path (*a, **kw)

Returns a fully-qualified project resource name string.

seek (*a, **kw)

Seeks an existing subscription to a point in time or to a given snapshot, whichever is provided in the request.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import subscriber_client
>>> client = subscriber_client.SubscriberClient()
>>> subscription = client.subscription_path('[PROJECT]', '[SUBSCRIPTION]')
>>> response = client.seek(subscription)
```

Parameters

- **subscription** (*string*) – The subscription to affect.
- **time** (`google.protobuf.timestamp_pb2.Timestamp`) – The time to seek to. Messages retained in the subscription that were published before this time are marked as acknowledged, and messages retained in the subscription that were published after this time are marked as unacknowledged. Note that this operation affects only those messages retained in the subscription (configured by the combination of `message_retention_duration` and `retain_acked_messages`). For example, if `time` corresponds to a point before the message retention window (or to a point before the system's notion of the subscription creation time), only retained messages will be marked as unacknowledged, and already-expunged messages will not be restored.
- **snapshot** (*string*) – The snapshot to seek to. The snapshot's topic must be the same as that of the provided subscription. Format is `projects/{project}/snapshots/{snap}`.
- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A `google.cloud.proto.pubsub.v1.pubsub_pb2.SeekResponse` instance.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

set_iam_policy (*a, **kw)

Sets the access control policy on the specified resource. Replaces any existing policy.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import subscriber_client
>>> from google.iam.v1 import policy_pb2
>>> client = subscriber_client.SubscriberClient()
>>> resource = client.subscription_path('[PROJECT]', '[SUBSCRIPTION]')
>>> policy = policy_pb2.Policy()
>>> response = client.set_iam_policy(resource, policy)
```

Parameters

- **resource** (*string*) – REQUIRED: The resource for which the policy is being specified. `resource` is usually specified as a path. For example, a Project resource is specified as `projects/{project}`.
- **policy** (`google.iam.v1.policy_pb2.Policy`) – REQUIRED: The complete policy to be applied to the `resource`. The size of the policy is limited to a few 10s of KB. An empty policy is a valid policy but certain Cloud Platform services (such as Projects) might reject them.
- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A `google.iam.v1.policy_pb2.Policy` instance.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

snapshot_path (**a, **kw*)

Returns a fully-qualified snapshot resource name string.

subscribe (*subscription, callback=None, flow_control=()*)

Return a representation of an individual subscription.

This method creates and returns a `Consumer` object (that is, a `BaseConsumer`) subclass bound to the topic. It does *not* create the subscription on the backend (or do any API call at all); it simply returns an object capable of doing these things.

If the `callback` argument is provided, then the `open()` method is automatically called on the returned object. If `callback` is not provided, the subscription is returned unopened.

Note: It only makes sense to provide `callback` here if you have already created the subscription manually in the API.

Parameters

- **subscription** (*str*) – The name of the subscription. The subscription should have already been created (for example, by using `create_subscription()`).
- **callback** (*function*) – The callback function. This function receives the `PubsubMessage` as its only argument.
- **flow_control** (`FlowControl`) – The flow control settings. Use this to prevent situations where you are inundated with too many messages at once.

Returns

An instance of the defined `consumer_class` on the client.

Return type BaseConsumer

subscription_path (*a, **kw)

Returns a fully-qualified subscription resource name string.

test_iam_permissions (*a, **kw)

Returns permissions that a caller has on the specified resource. If the resource does not exist, this will return an empty set of permissions, not a NOT_FOUND error.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import subscriber_client
>>> client = subscriber_client.SubscriberClient()
>>> resource = client.subscription_path('[PROJECT]', '[SUBSCRIPTION]')
>>> permissions = []
>>> response = client.test_iam_permissions(resource, permissions)
```

Parameters

- **resource** (*string*) – REQUIRED: The resource for which the policy detail is being requested. `resource` is usually specified as a path. For example, a Project resource is specified as `projects/{project}`.
- **permissions** (*list[string]*) – The set of permissions to check for the resource. Permissions with wildcards (such as `*` or `storage.*`) are not allowed. For more information see [IAM Overview](#).
- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g. timeout, retries etc.

Returns A `google.iam.v1.iam_policy_pb2.TestIamPermissionsResponse` instance.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

topic_path (*a, **kw)

Returns a fully-qualified topic resource name string.

update_subscription (*a, **kw)

Updates an existing subscription. Note that certain properties of a subscription, such as its topic, are not modifiable.

Example

```
>>> from google.cloud.gapic.pubsub.v1 import subscriber_client
>>> from google.cloud.proto.pubsub.v1 import pubsub_pb2
>>> from google.protobuf import field_mask_pb2
>>> client = subscriber_client.SubscriberClient()
>>> subscription = pubsub_pb2.Subscription()
>>> update_mask = field_mask_pb2.FieldMask()
>>> response = client.update_subscription(subscription, update_mask)
```

Parameters

- **subscription** (`google.cloud.proto.pubsub.v1.pubsub_pb2.Subscription`) – The updated subscription object.
- **update_mask** (`google.protobuf.field_mask_pb2.FieldMask`) – Indicates which fields in the provided subscription to update. Must be specified and non-empty.
- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A `google.cloud.proto.pubsub.v1.pubsub_pb2.Subscription` instance.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

Subscriptions

```
class google.cloud.pubsub_v1.subscriber.policy.thread.Policy (client, subscription,  
                                                         flow_control=FlowControl(max_bytes=827,  
                                                         max_messages=inf,  
                                                         re-  
                                                         sume_threshold=0.8),  
                                                         executor=None,  
                                                         queue=None)
```

A consumer class based on `threading.Thread`.

This consumer handles the connection to the Pub/Sub service and all of the concurrency needs.

Instantiate the policy.

Parameters

- **client** (*client*) – The subscriber client used to create this instance.
- **subscription** (*str*) – The name of the subscription. The canonical format for this is `projects/{project}/subscriptions/{subscription}`.
- **flow_control** (`FlowControl`) – The flow control settings.
- **executor** (`ThreadPoolExecutor`) – (Optional.) A `ThreadPoolExecutor` instance, or anything duck-type compatible with it.
- **queue** (`Queue`) – (Optional.) A `Queue` instance, appropriate for crossing the concurrency boundary implemented by `executor`.

close()

Close the existing connection.

open(callback)

Open a streaming pull connection and begin receiving messages.

For each message received, the `callback` function is fired with a `Message` as its only argument.

Parameters **callback** (`Callable`) – The callback function.

Messages

class google.cloud.pubsub_v1.subscriber.message.**Message** (*message*, *ack_id*, *request_queue*)

A representation of a single Pub/Sub message.

The common way to interact with *Message* objects is to receive them in callbacks on subscriptions; most users should never have a need to instantiate them by hand. (The exception to this is if you are implementing a custom subclass to *BaseConsumer*.)

message_id

str – The message ID. In general, you should not need to use this directly.

data

bytes – The data in the message. Note that this will be a *bytes*, not a text string.

attributes

dict – The attributes sent along with the message.

publish_time

datetime – The time that this message was originally published.

Construct the Message.

Note: This class should not be constructed directly; it is the responsibility of *BasePolicy* subclasses to do so.

Parameters

- **message** (*PubsubMessage*) – The message received from Pub/Sub.
- **ack_id** (*str*) – The *ack_id* received from Pub/Sub.
- **request_queue** (*queue.Queue*) – A queue provided by the policy that can accept requests; the policy is responsible for handling those requests.

ack()

Acknowledge the given message.

Acknowledging a message in Pub/Sub means that you are done with it, and it will not be delivered to this subscription again. You should avoid acknowledging messages until you have *finished* processing them, so that in the event of a failure, you receive the message again.

Warning: Acks in Pub/Sub are best effort. You should always ensure that your processing code is idempotent, as you may receive any given message more than once.

attributes

Return the attributes of the underlying Pub/Sub Message.

Returns The message's attributes.

Return type *dict*

data

Return the data for the underlying Pub/Sub Message.

Returns

The message data. This is always a `bytes` string; if you want a text string, call `bytes.decode()`.

Return type `bytes`

nack()

Decline to acknowledge the given message.

This will cause the message to be re-delivered to the subscription.

publish_time

Return the time that the message was originally published.

Returns The date and time that the message was published.

Return type `datetime`

10.4.3 Pub/Sub Client Types

class `google.cloud.pubsub_v1.types.AcknowledgeRequest`

Request for the Acknowledge method.

subscription

The subscription whose message is being acknowledged. Format is `projects/{project}/subscriptions/{sub}`.

ack_ids

The acknowledgment ID for the messages being acknowledged that was returned by the Pub/Sub system in the Pull response. Must not be empty.

class `google.cloud.pubsub_v1.types.BatchSettings` (*max_bytes*, *max_latency*,
max_messages)
Create new instance of BatchSettings(max_bytes, max_latency, max_messages)

max_bytes

Alias for field number 0

max_latency

Alias for field number 1

max_messages

Alias for field number 2

class `google.cloud.pubsub_v1.types.CreateSnapshotRequest`

Request for the CreateSnapshot method.

name

Optional user-provided name for this snapshot. If the name is not provided in the request, the server will assign a random name for this snapshot on the same project as the subscription. Note that for REST API requests, you must specify a name. Format is `projects/{project}/snapshots/{snap}`.

subscription

The subscription whose backlog the snapshot retains. Specifically, the created snapshot is guaranteed to retain: (a) The existing backlog on the subscription. More precisely, this is defined as the messages in the subscription's backlog that are unacknowledged upon the successful completion of the CreateSnapshot request; as well as: (b) Any messages published to the subscription's topic following the successful completion of the CreateSnapshot request. Format is `projects/{project}/subscriptions/{sub}`.

class `google.cloud.pubsub_v1.types.DeleteSnapshotRequest`

Request for the DeleteSnapshot method.

snapshot

The name of the snapshot to delete. Format is `projects/{project}/snapshots/{snap}`.

class `google.cloud.pubsub_v1.types.DeleteSubscriptionRequest`

Request for the `DeleteSubscription` method.

subscription

The subscription to delete. Format is `projects/{project}/subscriptions/{sub}`.

class `google.cloud.pubsub_v1.types.DeleteTopicRequest`

Request for the `DeleteTopic` method.

topic

Name of the topic to delete. Format is `projects/{project}/topics/{topic}`.

class `google.cloud.pubsub_v1.types.FlowControl` (*max_bytes*, *max_messages*, *resume_threshold*)

Create new instance of `FlowControl(max_bytes, max_messages, resume_threshold)`

max_bytes

Alias for field number 0

max_messages

Alias for field number 1

resume_threshold

Alias for field number 2

class `google.cloud.pubsub_v1.types.GetSubscriptionRequest`

Request for the `GetSubscription` method.

subscription

The name of the subscription to get. Format is `projects/{project}/subscriptions/{sub}`.

class `google.cloud.pubsub_v1.types.GetTopicRequest`

Request for the `GetTopic` method.

topic

The name of the topic to get. Format is `projects/{project}/topics/{topic}`.

class `google.cloud.pubsub_v1.types.ListSnapshotsRequest`

Request for the `ListSnapshots` method.

project

The name of the cloud project that snapshots belong to. Format is `projects/{project}`.

page_size

Maximum number of snapshots to return.

page_token

The value returned by the last `ListSnapshotsResponse`; indicates that this is a continuation of a prior `ListSnapshots` call, and that the system should return the next page of data.

class `google.cloud.pubsub_v1.types.ListSnapshotsResponse`

Response for the `ListSnapshots` method.

snapshots

The resulting snapshots.

next_page_token

If not empty, indicates that there may be more snapshot that match the request; this value should be passed in a new `ListSnapshotsRequest`.

class google.cloud.pubsub_v1.types.**ListSubscriptionsRequest**

Request for the ListSubscriptions method.

project

The name of the cloud project that subscriptions belong to. Format is projects/{project}.

page_size

Maximum number of subscriptions to return.

page_token

The value returned by the last ListSubscriptionsResponse; indicates that this is a continuation of a prior ListSubscriptions call, and that the system should return the next page of data.

class google.cloud.pubsub_v1.types.**ListSubscriptionsResponse**

Response for the ListSubscriptions method.

subscriptions

The subscriptions that match the request.

next_page_token

If not empty, indicates that there may be more subscriptions that match the request; this value should be passed in a new ListSubscriptionsRequest to get more subscriptions.

class google.cloud.pubsub_v1.types.**ListTopicSubscriptionsRequest**

Request for the ListTopicSubscriptions method.

topic

The name of the topic that subscriptions are attached to. Format is projects/{project}/topics/{topic}.

page_size

Maximum number of subscription names to return.

page_token

The value returned by the last ListTopicSubscriptionsResponse; indicates that this is a continuation of a prior ListTopicSubscriptions call, and that the system should return the next page of data.

class google.cloud.pubsub_v1.types.**ListTopicSubscriptionsResponse**

Response for the ListTopicSubscriptions method.

subscriptions

The names of the subscriptions that match the request.

next_page_token

If not empty, indicates that there may be more subscriptions that match the request; this value should be passed in a new ListTopicSubscriptionsRequest to get more subscriptions.

class google.cloud.pubsub_v1.types.**ListTopicsRequest**

Request for the ListTopics method.

project

The name of the cloud project that topics belong to. Format is projects/{project}.

page_size

Maximum number of topics to return.

page_token

The value returned by the last ListTopicsResponse; indicates that this is a continuation of a prior ListTopics call, and that the system should return the next page of data.

class google.cloud.pubsub_v1.types.**ListTopicsResponse**

Response for the ListTopics method.

topics

The resulting topics.

next_page_token

If not empty, indicates that there may be more topics that match the request; this value should be passed in a new `ListTopicsRequest`.

class `google.cloud.pubsub_v1.types.ModifyAckDeadlineRequest`

Request for the `ModifyAckDeadline` method.

subscription

The name of the subscription. Format is `projects/{project}/subscriptions/{sub}`.

ack_ids

List of acknowledgment IDs.

ack_deadline_seconds

The new ack deadline with respect to the time this request was sent to the Pub/Sub system. For example, if the value is 10, the new ack deadline will expire 10 seconds after the `ModifyAckDeadline` call was made. Specifying zero may immediately make the message available for another pull request. The minimum deadline you can specify is 0 seconds. The maximum deadline you can specify is 600 seconds (10 minutes).

class `google.cloud.pubsub_v1.types.ModifyPushConfigRequest`

Request for the `ModifyPushConfig` method.

subscription

The name of the subscription. Format is `projects/{project}/subscriptions/{sub}`.

push_config

The push configuration for future deliveries. An empty `pushConfig` indicates that the Pub/Sub system should stop pushing messages from the given subscription and allow messages to be pulled and acknowledged - effectively pausing the subscription if `Pull` is not called.

class `google.cloud.pubsub_v1.types.PublishRequest`

Request for the `Publish` method.

topic

The messages in the request will be published on this topic. Format is `projects/{project}/topics/{topic}`.

messages

The messages to publish.

class `google.cloud.pubsub_v1.types.PublishResponse`

Response for the `Publish` method.

message_ids

The server-assigned ID of each published message, in the same order as the messages in the request. IDs are guaranteed to be unique within the topic.

class `google.cloud.pubsub_v1.types.PubsubMessage`

A message data and its attributes. The message payload must not be empty; it must contain either a non-empty data field, or at least one attribute.

data

The message payload.

attributes

Optional attributes for this message.

message_id

ID of this message, assigned by the server when the message is published. Guaranteed to be unique within the topic. This value may be read by a subscriber that receives a `PubsubMessage` via a `Pull` call or a push delivery. It must not be populated by the publisher in a `Publish` call.

publish_time

The time at which the message was published, populated by the server when it receives the `Publish` call. It must not be populated by the publisher in a `Publish` call.

class `google.cloud.pubsub_v1.types.PullRequest`

Request for the `Pull` method.

subscription

The subscription from which messages should be pulled. Format is `projects/{project}/subscriptions/{sub}`.

return_immediately

If this field set to true, the system will respond immediately even if there are no messages available to return in the `Pull` response. Otherwise, the system may wait (for a bounded amount of time) until at least one message is available, rather than returning no messages. The client may cancel the request if it does not wish to wait any longer for the response.

max_messages

The maximum number of messages returned for this request. The Pub/Sub system may return fewer than the number specified.

class `google.cloud.pubsub_v1.types.PullResponse`

Response for the `Pull` method.

received_messages

Received Pub/Sub messages. The Pub/Sub system will return zero messages if there are no more available in the backlog. The Pub/Sub system may return fewer than the `maxMessages` requested even if there are more messages available in the backlog.

class `google.cloud.pubsub_v1.types.PushConfig`

Configuration for a push delivery endpoint.

push_endpoint

A URL locating the endpoint to which messages should be pushed. For example, a Webhook endpoint might use `"https://example.com/push"`.

attributes

Endpoint configuration attributes. Every endpoint has a set of API supported attributes that can be used to control different aspects of the message delivery. The currently supported attribute is `x-goog-version`, which you can use to change the format of the pushed message. This attribute indicates the version of the data expected by the endpoint. This controls the shape of the pushed message (i.e., its fields and metadata). The endpoint version is based on the version of the Pub/Sub API. If not present during the `CreateSubscription` call, it will default to the version of the API used to make such call. If not present during a `ModifyPushConfig` call, its value will not be changed. `GetSubscription` calls will always return a valid version, even if the subscription was created without this attribute. The possible values for this attribute are: `-v1beta1`: uses the push format defined in the `v1beta1` Pub/Sub API. `-v1` or `v1beta2`: uses the push format defined in the `v1` Pub/Sub API.

class `google.cloud.pubsub_v1.types.ReceivedMessage`

A message and its corresponding acknowledgment ID.

ack_id

This ID can be used to acknowledge the received message.

message

The message.

class google.cloud.pubsub_v1.types.**SeekRequest**

Request for the `Seek` method.

subscription

The subscription to affect.

time

The time to seek to. Messages retained in the subscription that were published before this time are marked as acknowledged, and messages retained in the subscription that were published after this time are marked as unacknowledged. Note that this operation affects only those messages retained in the subscription (configured by the combination of `message_retention_duration` and `retain_acked_messages`). For example, if `time` corresponds to a point before the message retention window (or to a point before the system's notion of the subscription creation time), only retained messages will be marked as unacknowledged, and already- expunged messages will not be restored.

snapshot

The snapshot to seek to. The snapshot's topic must be the same as that of the provided subscription. Format is `projects/{project}/snapshots/{snap}`.

class google.cloud.pubsub_v1.types.**Snapshot**

A snapshot resource.

name

The name of the snapshot.

topic

The name of the topic from which this snapshot is retaining messages.

expire_time

The snapshot is guaranteed to exist up until this time. A newly-created snapshot expires no later than 7 days from the time of its creation. Its exact lifetime is determined at creation by the existing backlog in the source subscription. Specifically, the lifetime of the snapshot is `7 days - (age of oldest unacked message in the subscription)`. For example, consider a subscription whose oldest unacked message is 3 days old. If a snapshot is created from this subscription, the snapshot – which will always capture this 3-day-old backlog as long as the snapshot exists – will expire in 4 days.

labels

User labels.

class google.cloud.pubsub_v1.types.**StreamingPullRequest**

Request for the `StreamingPull` streaming RPC method. This request is used to establish the initial stream as well as to stream acknowledgements and ack deadline modifications from the client to the server.

subscription

The subscription for which to initialize the new stream. This must be provided in the first request on the stream, and must not be set in subsequent requests from client to server. Format is `projects/{project}/subscriptions/{sub}`.

ack_ids

List of acknowledgement IDs for acknowledging previously received messages (received on this stream or a different stream). If an ack ID has expired, the corresponding message may be redelivered later. Acknowledging a message more than once will not result in an error. If the acknowledgement ID is malformed, the stream will be aborted with status `INVALID_ARGUMENT`.

modify_deadline_seconds

The list of new ack deadlines for the IDs listed in `modify_deadline_ack_ids`. The size of this list must be the same as the size of `modify_deadline_ack_ids`. If it differs the stream will be aborted

with `INVALID_ARGUMENT`. Each element in this list is applied to the element in the same position in `modify_deadline_ack_ids`. The new ack deadline is with respect to the time this request was sent to the Pub/Sub system. Must be ≥ 0 . For example, if the value is 10, the new ack deadline will expire 10 seconds after this request is received. If the value is 0, the message is immediately made available for another streaming or non-streaming pull request. If the value is < 0 (an error), the stream will be aborted with status `INVALID_ARGUMENT`.

modify_deadline_ack_ids

List of acknowledgement IDs whose deadline will be modified based on the corresponding element in `modify_deadline_seconds`. This field can be used to indicate that more time is needed to process a message by the subscriber, or to make the message available for redelivery if the processing was interrupted.

stream_ack_deadline_seconds

The ack deadline to use for the stream. This must be provided in the first request on the stream, but it can also be updated on subsequent requests from client to server. The minimum deadline you can specify is 10 seconds. The maximum deadline you can specify is 600 seconds (10 minutes).

class google.cloud.pubsub_v1.types.StreamingPullResponse

Response for the `StreamingPull` method. This response is used to stream messages from the server to the client.

received_messages

Received Pub/Sub messages. This will not be empty.

class google.cloud.pubsub_v1.types.Subscription

A subscription resource.

name

The name of the subscription. It must have the format `"projects/{project}/subscriptions/{subscription}"`. `{subscription}` must start with a letter, and contain only letters (`[A-Za-z]`), numbers (`[0-9]`), dashes (`-`), underscores (`_`), periods (`.`), tildes (`~`), plus (`+`) or percent signs (`%`). It must be between 3 and 255 characters in length, and it must not start with `"goog"`.

topic

The name of the topic from which this subscription is receiving messages. Format is `projects/{project}/topics/{topic}`. The value of this field will be `_deleted-topic_` if the topic has been deleted.

push_config

If push delivery is used with this subscription, this field is used to configure it. An empty `pushConfig` signifies that the subscriber will pull and ack messages using API methods.

ack_deadline_seconds

This value is the maximum time after a subscriber receives a message before the subscriber should acknowledge the message. After message delivery but before the ack deadline expires and before the message is acknowledged, it is an outstanding message and will not be delivered again during that time (on a best-effort basis). For pull subscriptions, this value is used as the initial value for the ack deadline. To override this value for a given message, call `ModifyAckDeadline` with the corresponding `ack_id` if using pull. The minimum custom deadline you can specify is 10 seconds. The maximum custom deadline you can specify is 600 seconds (10 minutes). If this parameter is 0, a default value of 10 seconds is used. For push delivery, this value is also used to set the request timeout for the call to the push endpoint. If the subscriber never acknowledges the message, the Pub/Sub system will eventually redeliver the message.

retain_acked_messages

Indicates whether to retain acknowledged messages. If true, then messages are not expunged from the subscription's backlog, even if they are acknowledged, until they fall out of the `message_retention_duration` window.

message_retention_duration

How long to retain unacknowledged messages in the subscription's backlog, from the moment a message is published. If `retain_acked_messages` is true, then this also configures the retention of acknowledged messages, and thus configures how far back in time a `Seek` can be done. Defaults to 7 days. Cannot be more than 7 days or less than 10 minutes.

labels

User labels.

class `google.cloud.pubsub_v1.types.Topic`

A topic resource.

name

The name of the topic. It must have the format `"projects/{project}/topics/{topic}"`. `{topic}` must start with a letter, and contain only letters (`[A-Za-z]`), numbers (`[0-9]`), dashes (`-`), underscores (`_`), periods (`.`), tildes (`~`), plus (`+`) or percent signs (`%`). It must be between 3 and 255 characters in length, and it must not start with `"goog"`.

labels

User labels.

class `google.cloud.pubsub_v1.types.UpdateSnapshotRequest`

Request for the `UpdateSnapshot` method.

snapshot

The updated snapshot object.

update_mask

Indicates which fields in the provided snapshot to update. Must be specified and non-empty.

class `google.cloud.pubsub_v1.types.UpdateSubscriptionRequest`

Request for the `UpdateSubscription` method.

subscription

The updated subscription object.

update_mask

Indicates which fields in the provided subscription to update. Must be specified and non-empty.

class `google.cloud.pubsub_v1.types.UpdateTopicRequest`

Request for the `UpdateTopic` method.

topic

The topic to update.

update_mask

Indicates which fields in the provided topic to update. Must be specified and non-empty.

11.1 Client

A Client for interacting with the Resource Manager API.

class `google.cloud.resource_manager.client.Client` (*credentials=None, _http=None*)

Bases: `google.cloud.client.Client`

Client to bundle configuration needed for API requests.

See <https://cloud.google.com/resource-manager/reference/rest/> for more information on this API.

Automatically get credentials:

```
>>> from google.cloud import resource_manager
>>> client = resource_manager.Client()
```

Parameters

- **credentials** (*Credentials*) – (Optional) The OAuth2 Credentials to use for this client. If not passed (and if no `_http` object is passed), falls back to the default inferred from the environment.
- **_http** (*Session*) – (Optional) HTTP object to make requests. Can be any object that defines `request()` with the same interface as `requests.Session.request()`. If not passed, an `_http` object is created that is bound to the `credentials` for the current object. This parameter should be considered private, and could change in the future.

SCOPE = ('https://www.googleapis.com/auth/cloud-platform',)

The scopes required for authenticating as a Resource Manager consumer.

fetch_project (*project_id*)

Fetch an existing project and its relevant metadata by ID.

Note: If the project does not exist, this will raise a `NotFound` error.

Parameters `project_id` (*str*) – The ID for this project.

Return type *Project*

Returns A *Project* with metadata fetched from the API.

list_projects (*filter_params=None, page_size=None*)

List the projects visible to this client.

Example:

```
>>> from google.cloud import resource_manager
>>> client = resource_manager.Client()
>>> for project in client.list_projects():
...     print(project.project_id)
```

List all projects with label 'environment' set to 'prod' (filtering by labels):

```
>>> from google.cloud import resource_manager
>>> client = resource_manager.Client()
>>> env_filter = {'labels.environment': 'prod'}
>>> for project in client.list_projects(env_filter):
...     print(project.project_id)
```

See <https://cloud.google.com/resource-manager/reference/rest/v1beta1/projects/list>

Complete filtering example:

```
>>> project_filter = { # Return projects with...
...     'name': 'My Project', # name set to 'My Project'.
...     'id': 'my-project-id', # id set to 'my-project-id'.
...     'labels.stage': 'prod', # the label 'stage' set to 'prod'
...     'labels.color': '*' # a label 'color' set to anything.
... }
>>> client.list_projects(project_filter)
```

Parameters

- **filter_params** (*dict*) – (Optional) A dictionary of filter options where each key is a property to filter on, and each value is the (case-insensitive) value to check (or the glob * to check for existence of the property). See the example above for more details.
- **page_size** (*int*) – (Optional) Maximum number of projects to return in a single page. If not passed, defaults to a value set by the API.

Return type *Iterator*

Returns Iterator of all *Project*. that the current user has access to.

new_project (*project_id, name=None, labels=None*)

Create a project bound to the current client.

Use *Project.reload()* to retrieve project metadata after creating a *Project* instance.

Parameters

- **project_id** (*str*) – The ID for this project.

- **name** (*str*) – The display name of the project.
- **labels** (*dict*) – A list of labels associated with the project.

Return type *Project*

Returns A new instance of a *Project* **without** any metadata loaded.

11.2 Projects

Utility for managing projects via the Cloud Resource Manager API.

```
class google.cloud.resource_manager.project.Project (project_id, client, name=None,  
                                                    labels=None)
```

Bases: *object*

Projects are containers for your work on Google Cloud Platform.

Note: A *Project* can also be created via *Client.new_project()*

To manage labels on a *Project*:

```
>>> from google.cloud import resource_manager
>>> client = resource_manager.Client()
>>> project = client.new_project('purple-spaceship-123')
>>> project.labels = {'color': 'purple'}
>>> project.labels['environment'] = 'production'
>>> project.update()
```

See <https://cloud.google.com/resource-manager/reference/rest/v1beta1/projects>

Parameters

- **project_id** (*str*) – The globally unique ID of the project.
- **client** (*google.cloud.resource_manager.client.Client*) – The *Client* used with this project.
- **name** (*str*) – The display name of the project.
- **labels** (*dict*) – A list of labels associated with the project.

create (*client=None*)

API call: create the project via a POST request.

See <https://cloud.google.com/resource-manager/reference/rest/v1beta1/projects/create>

Parameters **client** (*google.cloud.resource_manager.client.Client* or *NoneType*) – the client to use. If not passed, falls back to the client stored on the current project.

delete (*client=None, reload_data=False*)

API call: delete the project via a DELETE request.

See <https://cloud.google.com/resource-manager/reference/rest/v1beta1/projects/delete>

This actually changes the status (*lifecycleState*) from *ACTIVE* to *DELETE_REQUESTED*. Later (it's not specified when), the project will move into the *DELETE_IN_PROGRESS* state, which means the deleting has actually begun.

Parameters

- **client** (`google.cloud.resource_manager.client.Client` or `NoneType`) – the client to use. If not passed, falls back to the client stored on the current project.
- **reload_data** (`bool`) – Whether to reload the project with the latest state. If you want to get the updated status, you'll want this set to `True` as the `DELETE` method doesn't send back the updated project. Default: `False`.

exists (*client=None*)

API call: test the existence of a project via a GET request.

See <https://cloud.google.com/resource-manager/reference/rest/v1beta1/projects/get>

Parameters **client** (`google.cloud.resource_manager.client.Client` or `NoneType`) – the client to use. If not passed, falls back to the client stored on the current project.

Return type `bool`

Returns Boolean indicating existence of the project.

classmethod from_api_repr (*resource, client*)

Factory: construct a project given its API representation.

Parameters

- **resource** (*dict*) – project resource representation returned from the API
- **client** (`google.cloud.resource_manager.client.Client`) – The Client used with this project.

Return type `google.cloud.resource_manager.project.Project`

Returns The project created.

full_name

Fully-qualified name (ie, 'projects/purple-spaceship-123').

path

URL for the project (ie, '/projects/purple-spaceship-123').

reload (*client=None*)

API call: reload the project via a GET request.

This method will reload the newest metadata for the project. If you've created a new `Project` instance via `Client.new_project()`, this method will retrieve project metadata.

Warning: This will overwrite any local changes you've made and not saved via `update()`.

See <https://cloud.google.com/resource-manager/reference/rest/v1beta1/projects/get>

Parameters **client** (`google.cloud.resource_manager.client.Client` or `NoneType`) – the client to use. If not passed, falls back to the client stored on the current project.

set_properties_from_api_repr (*resource*)

Update specific properties from its API representation.

undelelete (*client=None, reload_data=False*)

API call: undelete the project via a POST request.

See <https://cloud.google.com/resource-manager/reference/rest/v1beta1/projects/undelelete>

This actually changes the project status (`lifecycleState`) from `DELETE_REQUESTED` to `ACTIVE`. If the project has already reached a status of `DELETE_IN_PROGRESS`, this request will fail and the project cannot be restored.

Parameters

- **client** (`google.cloud.resource_manager.client.Client` or `NoneType`) – the client to use. If not passed, falls back to the client stored on the current project.
- **reload_data** (`bool`) – Whether to reload the project with the latest state. If you want to get the updated status, you'll want this set to `True` as the `DELETE` method doesn't send back the updated project. Default: `False`.

update (`client=None`)

API call: update the project via a PUT request.

See <https://cloud.google.com/resource-manager/reference/rest/v1beta1/projects/update>

Parameters **client** (`google.cloud.resource_manager.client.Client` or `NoneType`) – the client to use. If not passed, falls back to the client stored on the current project.

The Cloud Resource Manager API provides methods that you can use to programmatically manage your projects in the Google Cloud Platform. With this API, you can do the following:

- Get a list of all projects associated with an account
- Create new projects
- Update existing projects
- Delete projects
- Undelete, or recover, projects that you don't want to delete

Note: Don't forget to look at the [Authentication](#) section below. It's slightly different from the rest of this library.

Warning: Alpha

The `projects.create()` API method is in the Alpha stage. It might be changed in backward-incompatible ways and is not recommended for production use. It is not subject to any SLA or deprecation policy. Access to this feature is currently invite-only. For an invitation, contact our sales team at <https://cloud.google.com/contact>.

Here's a quick example of the full life-cycle:

```
>>> from google.cloud import resource_manager
>>> client = resource_manager.Client()

>>> # List all projects you have access to
>>> for project in client.list_projects():
...     print(project)

>>> # Create a new project
>>> new_project = client.new_project('your-project-id-here',
...                                  name='My new project')
>>> new_project.create()

>>> # Update an existing project
```

```
>>> project = client.fetch_project('my-existing-project')
>>> print(project)
<Project: Existing Project (my-existing-project)>
>>> project.name = 'Modified name'
>>> project.update()
>>> print(project)
<Project: Modified name (my-existing-project)>

>>> # Delete a project
>>> project = client.new_project('my-existing-project')
>>> project.delete()

>>> # Undelete a project
>>> project = client.new_project('my-existing-project')
>>> project.undelete()
```

11.3 Authentication

Unlike the other APIs, the Resource Manager API is focused on managing your various projects inside Google Cloud Platform. What this means (currently, as of August 2015) is that you can't use a Service Account to work with some parts of this API (for example, creating projects).

The reason is actually pretty simple: if your API call is trying to do something like create a project, what project's Service Account can you use? Currently none.

This means that for this API you should always use the credentials provided by the [Google Cloud SDK](#), which you can get by running `gcloud auth login`.

Once you run that command, `google-cloud-python` will automatically pick up the credentials, and you can use the “automatic discovery” feature of the library.

Start by authenticating:

```
$ gcloud auth login
```

And then simply create a client:

```
>>> from google.cloud import resource_manager
>>> client = resource_manager.Client()
```


12.1 Runtime Configuration Client

Client for interacting with the Google Cloud RuntimeConfig API.

```
class google.cloud.runtimeconfig.client.Client (project=None, credentials=None,  
                                              _http=None)
```

Bases: *google.cloud.client.ClientWithProject*

Client to bundle configuration needed for API requests.

Parameters

- **project** (*str*) – (Optional) The project which the client acts on behalf of. If not passed, falls back to the default inferred from the environment.
- **credentials** (*Credentials*) – (Optional) The OAuth2 Credentials to use for this client. If not passed (and if no `_http` object is passed), falls back to the default inferred from the environment.
- **_http** (*Session*) – (Optional) HTTP object to make requests. Can be any object that defines `request()` with the same interface as `requests.Session.request()`. If not passed, an `_http` object is created that is bound to the `credentials` for the current object. This parameter should be considered private, and could change in the future.

```
SCOPE = ('https://www.googleapis.com/auth/cloudruntimeconfig',)
```

The scopes required for authenticating as a RuntimeConfig consumer.

```
config (config_name)
```

Factory constructor for config object.

Note: This will not make an HTTP request; it simply instantiates a config object owned by this client.

Parameters **config_name** (*str*) – The name of the config to be instantiated.

Return type `google.cloud.runtimeconfig.config.Config`

Returns The config object created.

12.2 Configuration

Create / interact with Google Cloud RuntimeConfig configs.

class `google.cloud.runtimeconfig.config.Config`(*client*, *name*)

Bases: `object`

A Config resource in the Cloud RuntimeConfig service.

This consists of metadata and a hierarchy of variables.

See <https://cloud.google.com/deployment-manager/runtime-configurator/reference/rest/v1beta1/projects.configs>

Parameters

- **client** (`google.cloud.runtimeconfig.client.Client`) – A client which holds credentials and project configuration for the config (which requires a project).
- **name** (`str`) – The name of the config.

client

The client bound to this config.

description

Description of the config object.

See <https://cloud.google.com/deployment-manager/runtime-configurator/reference/rest/v1beta1/projects.configs#resource-runtimeconfig>

Return type `str`, or `NoneType`

Returns the description (None until set from the server).

exists (*client=None*)

Determines whether or not this config exists.

Parameters **client** (`Client`) – (Optional) The client to use. If not passed, falls back to the `client` stored on the current config.

Return type `bool`

Returns True if the config exists in Cloud Runtime Configurator.

full_name

Fully-qualified name of this variable.

Example: `projects/my-project/configs/my-config`

Return type `str`

Returns The full name based on project and config names.

Raises `ValueError` if the config is missing a name.

get_variable (*variable_name*, *client=None*)

API call: get a variable via a GET request.

This will return None if the variable doesn't exist:

```
>>> from google.cloud import runtimeconfig
>>> client = runtimeconfig.Client()
>>> config = client.config('my-config')
>>> print(config.get_variable('variable-name'))
<Variable: my-config, variable-name>
>>> print(config.get_variable('does-not-exist'))
None
```

Parameters

- **variable_name** (*str*) – The name of the variable to retrieve.
- **client** (*Client*) – (Optional) The client to use. If not passed, falls back to the `client` stored on the current config.

Return type `google.cloud.runtimeconfig.variable.Variable` or `None`

Returns The variable object if it exists, otherwise `None`.

list_variables (*page_size=None, page_token=None, client=None*)

API call: list variables for this config.

This only lists variable names, not the values.

See <https://cloud.google.com/deployment-manager/runtime-configurator/reference/rest/v1beta1/projects.configs.variables/list>

Parameters

- **page_size** (*int*) – (Optional) Maximum number of variables to return per page.
- **page_token** (*str*) – opaque marker for the next “page” of variables. If not passed, will return the first page of variables.
- **client** (*Client*) – (Optional) The client to use. If not passed, falls back to the `client` stored on the current config.

Return type `Iterator`

Returns Iterator of `Variable` belonging to this project.

path

URL path for the config’s APIs.

Return type `str`

Returns The URL path based on project and config names.

project

Project bound to the config.

Return type `str`

Returns the project (derived from the client).

reload (*client=None*)

API call: reload the config via a GET request.

This method will reload the newest data for the config.

See <https://cloud.google.com/deployment-manager/runtime-configurator/reference/rest/v1beta1/projects.configs/get>

Parameters **client** (*google.cloud.runtimeconfig.client.Client*) – (Optional) The client to use. If not passed, falls back to the client stored on the current config.

variable (*variable_name*)

Factory constructor for variable object.

Note: This will not make an HTTP request; it simply instantiates a variable object owned by this config.

Parameters **variable_name** (*str*) – The name of the variable to be instantiated.

Return type *google.cloud.runtimeconfig.variable.Variable*

Returns The variable object created.

12.3 Variables

Create / interact with Google Cloud RuntimeConfig variables.

`google.cloud.runtimeconfig.variable.STATE_UNSPECIFIED`

The default variable state. See <https://cloud.google.com/deployment-manager/runtime-configurator/reference/rest/v1beta1/projects.configs.variables#VariableState>

`google.cloud.runtimeconfig.variable.STATE_UPDATED`

Indicates the variable was updated, while `variables.watch` was executing. See <https://cloud.google.com/deployment-manager/runtime-configurator/reference/rest/v1beta1/projects.configs.variables#VariableState>

`google.cloud.runtimeconfig.variable.STATE_DELETED`

Indicates the variable was deleted, while `variables.watch` was executing. See <https://cloud.google.com/deployment-manager/runtime-configurator/reference/rest/v1beta1/projects.configs.variables#VariableState>

class `google.cloud.runtimeconfig.variable.Variable` (*name, config*)

Bases: `object`

A variable in the Cloud RuntimeConfig service.

See <https://cloud.google.com/deployment-manager/runtime-configurator/reference/rest/v1beta1/projects.configs.variables>

Parameters

- **name** (*str*) – The name of the variable. This corresponds to the unique path of the variable in the config.
- **config** (*google.cloud.runtimeconfig.config.Config*) – The config to which this variable belongs.

client

The client bound to this variable.

exists (*client=None*)

API call: test for the existence of the variable via a GET request

See <https://cloud.google.com/deployment-manager/runtime-configurator/reference/rest/v1beta1/projects.configs.variables/get>

Parameters **client** (*Client*) – (Optional) The client to use. If not passed, falls back to the `client` stored on the variable's config.

Return type `bool`

Returns True if the variable exists in Cloud RuntimeConfig.

classmethod `from_api_repr(resource, config)`

Factory: construct a Variable given its API representation

Parameters

- **resource** (*dict*) – change set representation returned from the API.
- **config** (*google.cloud.runtimeconfig.config.Config*) – The config to which this variable belongs.

Return type *google.cloud.runtimeconfig.variable.Variable*

Returns Variable parsed from resource.

full_name

Fully-qualified name of this variable.

Example: `projects/my-project/configs/my-config/variables/my-var`

Return type *str*

Returns The full name based on config and variable names.

Raises *ValueError* if the variable is missing a name.

path

URL path for the variable's APIs.

Return type *str*

Returns The URL path based on config and variable names.

reload (*client=None*)

API call: reload the variable via a GET request.

This method will reload the newest data for the variable.

See <https://cloud.google.com/deployment-manager/runtime-configurator/reference/rest/v1beta1/projects.configs/get>

Parameters **client** (*google.cloud.runtimeconfig.client.Client*) – (Optional) The client to use. If not passed, falls back to the client stored on the current config.

state

Retrieve the state of the variable.

See <https://cloud.google.com/deployment-manager/runtime-configurator/reference/rest/v1beta1/projects.configs.variables#VariableState>

Return type *str*

Returns If set, one of “UPDATED”, “DELETED”, or “VARIABLE_STATE_UNSPECIFIED”, else None.

update_time

Retrieve the timestamp at which the variable was updated.

See <https://cloud.google.com/deployment-manager/runtime-configurator/reference/rest/v1beta1/projects.configs.variables>

Return type *datetime.datetime* or *NoneType*

Returns Datetime object parsed from RFC3339 valid timestamp, or None if the property is not set locally.

value

Value of the variable, as bytes.

See <https://cloud.google.com/deployment-manager/runtime-configurator/reference/rest/v1beta1/projects.configs.variables>

Return type bytes or `NoneType`

Returns The value of the variable or `None` if the property is not set locally.

12.4 Modules

Google Cloud Runtime Configurator API package.

class `google.cloud.runtimeconfig.Client` (*project=None, credentials=None, _http=None*)

Bases: `google.cloud.client.ClientWithProject`

Client to bundle configuration needed for API requests.

Parameters

- **project** (*str*) – (Optional) The project which the client acts on behalf of. If not passed, falls back to the default inferred from the environment.
- **credentials** (*Credentials*) – (Optional) The OAuth2 Credentials to use for this client. If not passed (and if no `_http` object is passed), falls back to the default inferred from the environment.
- **_http** (*Session*) – (Optional) HTTP object to make requests. Can be any object that defines `request()` with the same interface as `requests.Session.request()`. If not passed, an `_http` object is created that is bound to the `credentials` for the current object. This parameter should be considered private, and could change in the future.

config (*config_name*)

Factory constructor for config object.

Note: This will not make an HTTP request; it simply instantiates a config object owned by this client.

Parameters **config_name** (*str*) – The name of the config to be instantiated.

Return type `google.cloud.runtimeconfig.config.Config`

Returns The config object created.

13.1 Client

To use the API, the *Client* class defines a high-level interface which handles authorization and creating other objects:

```
from google.cloud.spanner.client import Client
client = Client()
```

13.1.1 Long-lived Defaults

When creating a *Client*, the `user_agent` and `timeout_seconds` arguments have sensible defaults (`DEFAULT_USER_AGENT` and `DEFAULT_TIMEOUT_SECONDS`). However, you may over-ride them and these will be used throughout all API requests made with the `client` you create.

13.1.2 Configuration

- For an overview of authentication in `google.cloud-python`, see [Authentication](#).
- In addition to any authentication configuration, you can also set the `G_CLOUD_PROJECT` environment variable for the Google Cloud Console project you'd like to interact with. If your code is running in Google App Engine or Google Compute Engine the project will be detected automatically. (Setting this environment variable is not required, you may instead pass the `project` explicitly when constructing a *Client*).
- After configuring your environment, create a *Client*

```
>>> from google.cloud import spanner
>>> client = spanner.Client()
```

or pass in credentials and project explicitly

```
>>> from google.cloud import spanner
>>> client = spanner.Client(project='my-project', credentials=creds)
```

Tip: Be sure to use the **Project ID**, not the **Project Number**.

13.1.3 Next Step

After a *Client*, the next highest-level object is an *Instance*. You'll need one before you can interact with databases.

Next, learn about the *Instance Admin API*.

13.2 Instance Admin API

After creating a *Client*, you can interact with individual instances for a project.

13.2.1 Instance Configurations

Each instance within a project maps to a named “instance configuration”, specifying the location and other parameters for a set of instances. These configurations are defined by the server, and cannot be changed.

To list of all instance configurations available to your project, use the `list_instance_configs()` method of the client:

```
configs, token = client.list_instance_configs()
```

To fetch a single instance configuration, use the `get_instance_configuration()` method of the client:

```
config = client.get_instance_configuration('config-name')
```

13.2.2 List Instances

If you want a comprehensive list of all existing instances, use the `list_instances()` method of the client:

```
instances, token = client.list_instances()
```

13.2.3 Instance Factory

To create a *Instance* object:

```
config = configs[0]
instance = client.instance(instance_id,
                           configuration_name=config.name,
                           node_count=10,
                           display_name='My Instance')
```

- `configuration_name` is the name of the instance configuration to which the instance will be bound. It must be one of the names configured for your project, discoverable via `google.cloud.spanner.client.Client.list_instance_configs()`.
- `node_count` is a positive integral count of the number of nodes used by the instance. More nodes allows for higher performance, but at a higher billing cost.

- `display_name` is optional. When not provided, `display_name` defaults to the `instance_id` value.

You can also use `Client.instance()` to create a local wrapper for an instance that has already been created:

```
instance = client.instance(existing_instance_id)
instance.reload()
```

13.2.4 Create a new Instance

After creating the instance object, use its `create()` method to trigger its creation on the server:

```
instance.display_name = 'My very own instance'
operation = instance.create()
```

Note: Creating an instance triggers a “long-running operation” and returns an `google.cloud.spanner.instance.Operation` object. See [Check on Current Instance Operation](#) for polling to find out if the operation is completed.

13.2.5 Refresh metadata for an existing Instance

After creating the instance object, reload its server-side configuration using its `reload()` method:

```
instance.reload()
```

This will load `display_name`, `config_name`, and `node_count` for the existing instance object from the back-end.

13.2.6 Update an existing Instance

After creating the instance object, you can update its metadata via its `update()` method:

```
client.display_name = 'New display_name'
operation = instance.update()
```

Note: Update an instance triggers a “long-running operation” and returns a `google.cloud.spanner.instance.Operation` object. See [Check on Current Instance Operation](#) for polling to find out if the operation is completed.

13.2.7 Delete an existing Instance

Delete an instance using its `delete()` method:

```
instance.delete()
```

13.2.8 Check on Current Instance Operation

The `create()` and `update()` methods of instance object trigger long-running operations on the server, and return instances of the `Operation` class.

You can check if a long-running operation has finished by using its `finished()` method:

```
>>> operation = instance.create()
>>> operation.finished()
True
```

Note: Once an `Operation` object has returned `True` from its `finished()` method, the object should not be re-used. Subsequent calls to `finished()` will result in an `exc'ValueError'` being raised.

13.2.9 Next Step

Now we go down the hierarchy from *Instance* to a *Database*.

Next, learn about the *Database Admin API*.

13.3 Database Admin API

After creating a *Instance*, you can interact with individual databases for that instance.

13.3.1 List Databases

To list of all existing databases for an instance, use its `list_databases()` method:

```
databases, token = instance.list_databases()
```

13.3.2 Database Factory

To create a *Database* object:

```
database = instance.database(database_id, ddl_statements)
```

- `ddl_statements` is a string containing DDL for the new database.

You can also use `Instance.database()` to create a local wrapper for a database that has already been created:

```
database = instance.database(existing_database_id)
```

13.3.3 Create a new Database

After creating the database object, use its `create()` method to trigger its creation on the server:

```
operation = database.create()
```

Note: Creating an instance triggers a “long-running operation” and returns an `google.cloud.spanner.database.Operation` object. See [Check on Current Database Operation](#) for polling to find out if the operation is completed.

13.3.4 Update an existing Database

After creating the database object, you can apply additional DDL statements via its `update_ddl()` method:

```
operation = instance.update_ddl(ddl_statements, operation_id)
```

- `ddl_statements` is a string containing DDL to be applied to the database.
- `operation_id` is a string ID for the long-running operation.

Note: Update an instance triggers a “long-running operation” and returns a `google.cloud.spanner.database.Operation` object. See [Check on Current Database Operation](#) for polling to find out if the operation is completed.

13.3.5 Drop a Database

Drop a database using its `drop()` method:

```
database.drop()
```

13.3.6 Check on Current Database Operation

The `create()` and `update()` methods of instance object trigger long-running operations on the server, and return instances of the `Operation` class.

You can check if a long-running operation has finished by using its `finished()` method:

```
>>> operation = instance.create()
>>> operation.finished()
True
```

Note: Once an `Operation` object has returned `True` from its `finished()` method, the object should not be re-used. Subsequent calls to `finished()` will result in an `:exc'ValueError'` being raised.

13.4 Non-Admin Database Usage

13.4.1 Use a Snapshot to Read / Query the Database

A snapshot represents a read-only point-in-time view of the database.

Calling `snapshot()` with no arguments creates a snapshot with strong concurrency:

```
with database.snapshot() as snapshot:
    do_something_with(snapshot)
```

See [Snapshot](#) for the other options which can be passed.

Note: `snapshot()` returns an object intended to be used as a Python context manager (i.e., as the target of a `with` statement). Use the instance, and any result sets returned by its `read` or `execute_sql` methods, only inside the block created by the `with` statement.

See [Read-only Transactions via Snapshots](#) for more complete examples of snapshot usage.

13.4.2 Use a Batch to Modify Rows in the Database

A batch represents a bundled set of insert/upsert/update/delete operations on the rows of tables in the database.

```
with database.batch() as batch:
    batch.insert_or_update(table, columns, rows)
    batch.delete(table, keyset_to_delete)
```

Note: `batch()` returns an object intended to be used as a Python context manager (i.e., as the target of a `with` statement). It applies any changes made inside the block of its `with` statement when exiting the block, unless an exception is raised within the block. Use the batch only inside the block created by the `with` statement.

See [Batching Modifications](#) for more complete examples of batch usage.

13.4.3 Use a Transaction to Query / Modify Rows in the Database

A transaction represents the union of a “strong” snapshot and a batch: it allows `read` and `execute_sql` operations, and accumulates insert/upsert/update/delete operations.

Because other applications may be performing concurrent updates which would invalidate the reads / queries, the work done by a transaction needs to be bundled as a retryable “unit of work” function, which takes the transaction as a required argument:

```
def unit_of_work(transaction):
    result = transaction.execute_sql(QUERY)

    for emp_id, hours, pay in _compute_pay(result):
        transaction.insert_or_update(
            table='monthly_hours',
            columns=['employee_id', 'month', 'hours', 'pay'],
            values=[emp_id, month_start, hours, pay])

database.run_in_transaction(unit_of_work)
```

Note: `run_in_transaction()` commits the transaction automatically if the “unit of work” function returns without raising an exception.

Note: `run_in_transaction()` retries the “unit of work” function if the read / query operations or the commit are aborted due to concurrent updates

See [Read-write Transactions](#) for more complete examples of transaction usage.

13.4.4 Configuring a session pool for a database

Under the covers, the `snapshot`, `batch`, and `run_in_transaction` methods use a pool of `Session` objects to manage their communication with the back-end. You can configure one of the pools manually to control the number of sessions, timeouts, etc., and then passing it to the `Database` constructor:

```
from google.cloud.spanner import Client
from google.cloud.spanner import FixedSizePool
client = Client()
instance = client.instance(INSTANCE_NAME)
pool = FixedSizePool(size=10, default_timeout=5)
database = instance.database(DATABASE_NAME, pool=pool)
```

Note that creating a database with a pool may presume that its database already exists, as it may need to pre-create sessions (rather than creating them on demand, as the default implementation does).

You can supply your own pool implementation, which must satisfy the contract laid out in `AbstractSessionPool`:

```
from google.cloud.pool import AbstractSessionPool

class MyCustomPool(AbstractSessionPool):

    def __init__(self, database, custom_param):
        super(MyCustomPool, self).__init__(database)
        self.custom_param = custom_param

    def get(self, read_only=False):
        ...

    def put(self, session, discard_if_full=True):
        ...

database = instance.database(DATABASE_NAME, pool=pool)
pool = MyCustomPool(database, custom_param=42)
```

See [Advanced Session Pool Topics](#) for more advanced coverage of session pools.

13.5 Batching Modifications

A `Batch` represents a set of data modification operations to be performed on tables in a dataset. Use of a `Batch` does not require creating an explicit `Snapshot` or `Transaction`. Until `commit()` is called on a `Batch`, no changes are propagated to the back-end.

13.5.1 Starting a Batch

```
batch = session.batch()
```

13.5.2 Inserting records using a Batch

`Batch.insert()` adds one or more new records to a table. Fails if any of the records already exists.

```
batch.insert(
    'citizens', columns=['email', 'first_name', 'last_name', 'age'],
    values=[
        ['phred@example.com', 'Phred', 'Phlyntstone', 32],
        ['bharney@example.com', 'Bharney', 'Rhubble', 31],
    ])
```

Note: Ensure that data being sent for `STRING` columns uses a text string (`str` in Python 3; `unicode` in Python 2). Additionally, if you are writing data intended for a `BYTES` column, you must `base64` encode it.

13.5.3 Update records using a Batch

`Batch.update()` updates one or more existing records in a table. Fails if any of the records does not already exist.

```
batch.update(
    'citizens', columns=['email', 'age'],
    values=[
        ['phred@example.com', 33],
        ['bharney@example.com', 32],
    ])
```

Note: Ensure that data being sent for `STRING` columns uses a text string (`str` in Python 3; `unicode` in Python 2). Additionally, if you are writing data intended for a `BYTES` column, you must `base64` encode it.

13.5.4 Insert or update records using a Batch

`Batch.insert_or_update()` inserts *or* updates one or more records in a table. Existing rows have values for the supplied columns overwritten; other column values are preserved.

```
batch.insert_or_update(
    'citizens', columns=['email', 'first_name', 'last_name', 'age'],
    values=[
        ['phred@example.com', 'Phred', 'Phlyntstone', 31],
        ['wylma@example.com', 'Wylma', 'Phlyntstone', 29],
    ])
```

Note: Ensure that data being sent for `STRING` columns uses a text string (`str` in Python 3; `unicode` in Python 2). Additionally, if you are writing data intended for a `BYTES` column, you must `base64` encode it.

13.5.5 Replace records using a Batch

`Batch.replace()` inserts *or* updates one or more records in a table. Existing rows have values for the supplied columns overwritten; other column values are set to null.

```
batch.replace(
    'citizens', columns=['email', 'first_name', 'last_name', 'age'],
    values=[
        ['bharney@example.com', 'Bharney', 'Rhubble', 30],
        ['bhettye@example.com', 'Bhettye', 'Rhubble', 30],
    ])
```

Note: Ensure that data being sent for `STRING` columns uses a text string (`str` in Python 3; `unicode` in Python 2). Additionally, if you are writing data intended for a `BYTES` column, you must base64 encode it.

13.5.6 Delete records using a Batch

`Batch.delete()` removes one or more records from a table. Non-existent rows do not cause errors.

```
from google.cloud.spanner.keyset import KeySet

to_delete = KeySet(keys=[
    ('bharney@example.com',),
    ('nonesuch@example.com',)
])

batch.delete('citizens', to_delete)
```

13.5.7 Commit changes for a Batch

After describing the modifications to be made to table data via the `Batch.insert()`, `Batch.update()`, `Batch.insert_or_update()`, `Batch.replace()`, and `Batch.delete()` methods above, send them to the back-end by calling `Batch.commit()`, which makes the Commit API call.

```
batch.commit()
```

13.5.8 Use a Batch as a Context Manager

Rather than calling `Batch.commit()` manually, you can use the `Batch` instance as a context manager, and have it called automatically if the `with` block exits without raising an exception.

```
from google.cloud.spanner.keyset import KeySet

to_delete = KeySet(keys=[
    ('bharney@example.com',),
    ('nonesuch@example.com',)
])

with session.batch() as batch:
```

```
batch.insert(
    'citizens', columns=['email', 'first_name', 'last_name', 'age'],
    values=[
        ['phred@example.com', 'Phred', 'Phlyntstone', 32],
        ['bharney@example.com', 'Bharney', 'Rhubble', 31],
    ])

batch.update(
    'citizens', columns=['email', 'age'],
    values=[
        ['phred@example.com', 33],
        ['bharney@example.com', 32],
    ])

...

batch.delete('citizens', to_delete)
```

13.5.9 Next Step

Next, learn about *Read-only Transactions via Snapshots*.

13.6 Read-only Transactions via Snapshots

A *Snapshot* represents a read-only transaction: when multiple read operations are performed via a Snapshot, the results are consistent as of a particular point in time.

13.6.1 Beginning a Snapshot

To begin using a snapshot using the default “bound” (which is “strong”), meaning all reads are performed at a timestamp where all previously-committed transactions are visible:

```
snapshot = session.snapshot()
```

You can also specify a weaker bound, which can either be to perform all reads as of a given timestamp:

```
import datetime
from pytz import UTC
TIMESTAMP = datetime.utcnow().replace(tzinfo=UTC)
snapshot = session.snapshot(read_timestamp=TIMESTAMP)
```

or as of a given duration in the past:

```
import datetime
DURATION = datetime.timedelta(seconds=5)
snapshot = session.snapshot(exact staleness=DURATION)
```

13.6.2 Read Table Data

Read data for selected rows from a table in the session’s database. Calls the `Read API`, which returns all rows specified in `key_set`, or else fails if the result set is too large,


```

with database.snapshot() as snapshot:
    result = snapshot.read(
        table='table-name', columns=['first_name', 'last_name', 'age'],
        key_set=['phred@example.com', 'bharney@example.com'])

    for row in result.rows:
        print(row)

```

Note: The result set returned by `execute_sql()` *must not* be iterated after the snapshot's session has been returned to the database's session pool. Therefore, unless your application creates sessions manually, perform all iteration within the context of the `with database.snapshot()` block.

Note: If streaming a chunk raises an exception, the application can retry the `read`, passing the `resume_token` from `StreamingResultSet` which raised the error. E.g.:

```

result = snapshot.read(table, columns, keys)
while True:
    try:
        for row in result.rows:
            print row
    except Exception:
        result = snapshot.read(
            table, columns, keys, resume_token=result.resume_token)
        continue
    else:
        break

```

13.6.3 Execute a SQL Select Statement

Read data from a query against tables in the session's database. Calls the `ExecuteSql` API, which returns all rows matching the query, or else fails if the result set is too large,

```

with database.snapshot() as snapshot:
    QUERY = (
        'SELECT e.first_name, e.last_name, p.telephone '
        'FROM employees as e, phones as p '
        'WHERE p.employee_id == e.employee_id')
    result = snapshot.execute_sql(QUERY)

    for row in result.rows:
        print(row)

```

Note: The result set returned by `execute_sql()` *must not* be iterated after the snapshot's session has been returned to the database's session pool. Therefore, unless your application creates sessions manually, perform all iteration within the context of the `with database.snapshot()` block.

Note: If streaming a chunk raises an exception, the application can retry the query, passing the `resume_token` from `StreamingResultSet` which raised the error. E.g.:

```
result = snapshot.execute_sql(QUERY)
while True:
    try:
        for row in result.rows:
            print row
    except Exception:
        result = snapshot.execute_sql(
            QUERY, resume_token=result.resume_token)
        continue
    else:
        break
```

13.6.4 Next Step

Next, learn about *Read-write Transactions*.

13.7 Read-write Transactions

A *Transaction* represents a transaction: when the transaction commits, it will send any accumulated mutations to the server.

13.7.1 Begin a Transaction

To begin using a transaction:

```
transaction = session.transaction()
```

13.7.2 Read Table Data

Read data for selected rows from a table in the session's database. Calls the `Read` API, which returns all rows specified in `key_set`, or else fails if the result set is too large,

```
result = transaction.read(
    table='table-name', columns=['first_name', 'last_name', 'age'],
    key_set=['phred@example.com', 'bharney@example.com'])

for row in result.rows:
    print(row)
```

Note: If streaming a chunk fails due to a “resumable” error, `Session.read()` retries the `StreamingRead` API request, passing the `resume_token` from the last partial result streamed.

13.7.3 Execute a SQL Select Statement

Read data from a query against tables in the session's database. Calls the `ExecuteSql` API, which returns all rows matching the query, or else fails if the result set is too large,

```

QUERY = (
    'SELECT e.first_name, e.last_name, p.telephone '
    'FROM employees as e, phones as p '
    'WHERE p.employee_id == e.employee_id')
result = transaction.execute_sql(QUERY)

for row in result.rows:
    print(row)

```

13.7.4 Insert records using a Transaction

`Transaction.insert()` adds one or more new records to a table. Fails if any of the records already exists.

```

transaction.insert(
    'citizens', columns=['email', 'first_name', 'last_name', 'age'],
    values=[
        ['phred@example.com', 'Phred', 'Phlyntstone', 32],
        ['bharney@example.com', 'Bharney', 'Rhubble', 31],
    ])

```

Note: Ensure that data being sent for `STRING` columns uses a text string (`str` in Python 3; `unicode` in Python 2). Additionally, if you are writing data intended for a `BYTES` column, you must `base64` encode it.

13.7.5 Update records using a Transaction

`Transaction.update()` updates one or more existing records in a table. Fails if any of the records does not already exist.

```

transaction.update(
    'citizens', columns=['email', 'age'],
    values=[
        ['phred@example.com', 33],
        ['bharney@example.com', 32],
    ])

```

Note: Ensure that data being sent for `STRING` columns uses a text string (`str` in Python 3; `unicode` in Python 2). Additionally, if you are writing data intended for a `BYTES` column, you must `base64` encode it.

13.7.6 Insert or update records using a Transaction

`Transaction.insert_or_update()` inserts *or* updates one or more records in a table. Existing rows have values for the supplied columns overwritten; other column values are preserved.

```

transaction.insert_or_update(
    'citizens', columns=['email', 'first_name', 'last_name', 'age'],
    values=[
        ['phred@example.com', 'Phred', 'Phlyntstone', 31],
    ])

```

```
[ 'wylma@example.com', 'Wylma', 'Phlyntstone', 29],  
])
```

Note: Ensure that data being sent for STRING columns uses a text string (`str` in Python 3; `unicode` in Python 2). Additionally, if you are writing data intended for a BYTES column, you must `base64` encode it.

13.7.7 Replace records using a Transaction

`Transaction.replace()` inserts *or* updates one or more records in a table. Existing rows have values for the supplied columns overwritten; other column values are set to null.

```
transaction.replace(  
    'citizens', columns=['email', 'first_name', 'last_name', 'age'],  
    values=[  
        ['bharney@example.com', 'Bharney', 'Rhubble', 30],  
        ['bhettie@example.com', 'Bhettie', 'Rhubble', 30],  
    ]  
)
```

Note: Ensure that data being sent for STRING columns uses a text string (`str` in Python 3; `unicode` in Python 2). Additionally, if you are writing data intended for a BYTES column, you must `base64` encode it.

13.7.8 Delete records using a Transaction

`Transaction.delete()` removes one or more records from a table. Non-existent rows do not cause errors.

```
transaction.delete(  
    'citizens', keyset=['bharney@example.com', 'nonesuch@example.com'])
```

13.7.9 Commit changes for a Transaction

After describing the modifications to be made to table data via the `Transaction.insert()`, `Transaction.update()`, `Transaction.insert_or_update()`, `Transaction.replace()`, and `Transaction.delete()` methods above, send them to the back-end by calling `Transaction.commit()`, which makes the Commit API call.

```
transaction.commit()
```

13.7.10 Roll back changes for a Transaction

After describing the modifications to be made to table data via the `Transaction.insert()`, `Transaction.update()`, `Transaction.insert_or_update()`, `Transaction.replace()`, and `Transaction.delete()` methods above, cancel the transaction on the the back-end by calling `Transaction.rollback()`, which makes the Rollback API call.

```
transaction.rollback()
```

13.7.11 Use a Transaction as a Context Manager

Rather than calling `Transaction.commit()` or `Transaction.rollback()` manually, you can use the `Transaction` instance as a context manager: in that case, the transaction's `commit()` method will be called automatically if the `with` block exits without raising an exception.

If an exception is raised inside the `with` block, the transaction's `rollback()` method will be called instead.

```
with session.transaction() as transaction:

    transaction.insert(
        'citizens', columns=['email', 'first_name', 'last_name', 'age'],
        values=[
            ['phred@example.com', 'Phred', 'Phlyntstone', 32],
            ['bharney@example.com', 'Bharney', 'Rhubble', 31],
        ])

    transaction.update(
        'citizens', columns=['email', 'age'],
        values=[
            ['phred@example.com', 33],
            ['bharney@example.com', 32],
        ])

    ...

    transaction.delete('citizens',
        keyset['bharney@example.com', 'nonesuch@example.com'])
```

13.8 Advanced Session Pool Topics

13.8.1 Custom Session Pool Implementations

You can supply your own pool implementation, which must satisfy the contract laid out in `AbstractSessionPool`:

```
from google.cloud.spanner.pool import AbstractSessionPool

class MyCustomPool(AbstractSessionPool):

    def __init__(self, custom_param):
        super(MyCustomPool, self).__init__()
        self.custom_param = custom_param

    def bind(self, database):
        ...

    def get(self, read_only=False):
        ...

    def put(self, session, discard_if_full=True):
        ...

pool = MyCustomPool(custom_param=42)
database = instance.database(DATABASE_NAME, pool=pool)
```

13.8.2 Lowering latency for read / query operations

Some applications may need to minimize latency for read operations, including particularly the overhead of making an API request to create or refresh a session. *PingingPool* is designed for such applications, which need to configure a background thread to do the work of keeping the sessions fresh.

Create an instance of *PingingPool*:

```
from google.cloud.spanner import Client
from google.cloud.spanner.pool import PingingPool

client = Client()
instance = client.instance(INSTANCE_NAME)
pool = PingingPool(size=10, default_timeout=5, ping_interval=300)
database = instance.database(DATABASE_NAME, pool=pool)
```

Set up a background thread to ping the pool's session, keeping them from becoming stale:

```
import threading

background = threading.Thread(target=pool.ping, name='ping-pool')
background.daemon = True
background.start()
```

13.8.3 Lowering latency for mixed read-write operations

Some applications may need to minimize latency for read write operations, including particularly the overhead of making an API request to create or refresh a session or to begin a session's transaction. *TransactionPingingPool* is designed for such applications, which need to configure a background thread to do the work of keeping the sessions fresh and starting their transactions after use.

Create an instance of *TransactionPingingPool*:

```
from google.cloud.spanner import Client
from google.cloud.spanner.pool import TransactionPingingPool

client = Client()
instance = client.instance(INSTANCE_NAME)
pool = TransactionPingingPool(size=10, default_timeout=5, ping_interval=300)
database = instance.database(DATABASE_NAME, pool=pool)
```

Set up a background thread to ping the pool's session, keeping them from becoming stale, and ensuring that each session has a new transaction started before it is used:

```
import threading

background = threading.Thread(target=pool.ping, name='ping-pool')
background.daemon = True
background.start()
```

13.9 Spanner Client

Parent client for calling the Cloud Spanner API.

This is the base from which all interactions with the API occur.

In the hierarchy of API concepts

- a *Client* owns an *Instance*
- a *Instance* owns a *Database*

class google.cloud.spanner.client.**Client** (*project=None*, *credentials=None*,
user_agent='gcloud-python/0.27.1')

Bases: *google.cloud.client.ClientWithProject*

Client for interacting with Cloud Spanner API.

Note: Since the Cloud Spanner API requires the gRPC transport, no `_http` argument is accepted by this class.

Parameters

- **project** (*str* or *unicode*) – (Optional) The ID of the project which owns the instances, tables and data. If not provided, will attempt to determine from the environment.
- **credentials** (*OAuth2Credentials* or *NoneType*) – (Optional) The OAuth2 Credentials to use for this client. If not provided, defaults to the Google Application Default Credentials.
- **user_agent** (*str*) – (Optional) The user agent to be used with API request. Defaults to `DEFAULT_USER_AGENT`.

Raises *ValueError* if both `read_only` and `admin` are `True`

SCOPE = ('https://www.googleapis.com/auth/spanner.admin',)

The scopes required for Google Cloud Spanner.

copy()

Make a copy of this client.

Copies the local data stored as simple types but does not copy the current state of any open connections with the Cloud Bigtable API.

Return type *Client*

Returns A copy of the current client.

credentials

Getter for client's credentials.

Return type *OAuth2Credentials*

Returns The credentials stored on the client.

database_admin_api

Helper for session-related API calls.

instance (*instance_id*, *configuration_name=None*, *display_name=None*, *node_count=1*)

Factory to create a instance associated with this client.

Parameters

- **instance_id** (*str*) – The ID of the instance.
- **configuration_name** (*string*) – (Optional) Name of the instance configuration used to set up the instance's cluster, in the form: `projects/<project>/instanceConfigs/<config>`. **Required** for instances which do not yet exist.

- **display_name** (*str*) – (Optional) The display name for the instance in the Cloud Console UI. (Must be between 4 and 30 characters.) If this value is not set in the constructor, will fall back to the instance ID.
- **node_count** (*int*) – (Optional) The number of nodes in the instance's cluster; used to set up the instance's cluster.

Return type *Instance*

Returns an instance owned by this client.

instance_admin_api

Helper for session-related API calls.

list_instance_configs (*page_size=None, page_token=None*)

List available instance configurations for the client's project.

See [RPC docs](#).

Parameters

- **page_size** (*int*) – (Optional) Maximum number of results to return.
- **page_token** (*str*) – (Optional) Token for fetching next page of results.

Return type *Iterator*

Returns Iterator of *InstanceConfig* resources within the client's project.

list_instances (*filter_=", page_size=None, page_token=None*)

List instances for the client's project.

See <https://cloud.google.com/spanner/reference/rpc/google.spanner.admin.database.v1#google.spanner.admin.database.v1.InstanceAdmin.ListInstances>

Parameters

- **filter** (*string*) – (Optional) Filter to select instances listed. See the *ListInstancesRequest* docs above for examples.
- **page_size** (*int*) – (Optional) Maximum number of results to return.
- **page_token** (*str*) – (Optional) Token for fetching next page of results.

Return type *Iterator*

Returns Iterator of *Instance* resources within the client's project.

project_name

Project name to be used with Spanner APIs.

Note: This property will not change if *project* does not, but the return value is not cached.

The project name is of the form

"projects/{project}"

Return type *str*

Returns The project name to be used with the Cloud Spanner Admin API RPC service.

class `google.cloud.spanner.client.InstanceConfig` (*name, display_name*)

Bases: *object*

Named configurations for Spanner instances.

Parameters

- **name** (*str*) – ID of the instance configuration
- **display_name** (*str*) – Name of the instance configuration

classmethod **from_pb** (*config_pb*)

Construct an instance from the equivalent protobuf.

Parameters **config_pb** (*InstanceConfig*) – the protobuf to parse

Return type *InstanceConfig*

Returns an instance of this class

13.10 Instance API

User friendly container for Cloud Spanner Instance.

```
class google.cloud.spanner.instance.Instance (instance_id, client, configuration_name=None, node_count=1, display_name=None)
```

Bases: *object*

Representation of a Cloud Spanner Instance.

We can use a *Instance* to:

- *reload()* itself
- *create()* itself
- *update()* itself
- *delete()* itself

Parameters

- **instance_id** (*str*) – The ID of the instance.
- **client** (*Client*) – The client that owns the instance. Provides authorization and a project ID.
- **configuration_name** (*str*) – Name of the instance configuration defining how the instance will be created. Required for instances which do not yet exist.
- **node_count** (*int*) – (Optional) Number of nodes allocated to the instance.
- **display_name** (*str*) – (Optional) The display name for the instance in the Cloud Console UI. (Must be between 4 and 30 characters.) If this value is not set in the constructor, will fall back to the instance ID.

copy ()

Make a copy of this instance.

Copies the local data stored as simple types and copies the client attached to this instance.

Return type *Instance*

Returns A copy of the current instance.

create()

Create this instance.

See <https://cloud.google.com/spanner/reference/rpc/google.spanner.admin.instance.v1#google.spanner.admin.instance.v1.InstanceAdmin.CreateInstance>

Note: Uses the project and instance_id on the current *Instance* in addition to the display_name. To change them before creating, reset the values via

```
instance.display_name = 'New display name'
instance.instance_id = 'i-changed-my-mind'
```

before calling *create()*.

Return type `google.api.core.operation.Operation`

Returns an operation instance

Raises

- **Conflict** – if the instance already exists
- **GaxError** – for errors other than `ALREADY_EXISTS` returned from the call

database(database_id, ddl_statements=(), pool=None)

Factory to create a database within this instance.

Parameters

- **database_id** (*str*) – The ID of the instance.
- **ddl_statements** (*list of string*) – (Optional) DDL statements, excluding the ‘CREATE DATABASE’ statement.
- **pool** (concrete subclass of *AbstractSessionPool*.) – (Optional) session pool to be used by database.

Return type *Database*

Returns a database owned by this instance.

delete()

Mark an instance and all of its databases for permanent deletion.

See <https://cloud.google.com/spanner/reference/rpc/google.spanner.admin.instance.v1#google.spanner.admin.instance.v1.InstanceAdmin.DeleteInstance>

Immediately upon completion of the request:

- Billing will cease for all of the instance’s reserved resources.

Soon afterward:

- The instance and all databases within the instance will be deleted. All data in the databases will be permanently deleted.

exists()

Test whether this instance exists.

See <https://cloud.google.com/spanner/reference/rpc/google.spanner.admin.instance.v1#google.spanner.admin.instance.v1.InstanceAdmin.GetInstanceConfig>

Return type `bool`

Returns True if the instance exists, else false

Raises **GaxError** – for errors other than NOT_FOUND returned from the call

classmethod **from_pb**(*instance_pb*, *client*)

Creates an instance from a protobuf.

Parameters

- **instance_pb** (`google.spanner.v2.spanner_instance_admin_pb2.Instance`) – A instance protobuf object.
- **client** (*Client*) – The client that owns the instance.

Return type *Instance*

Returns The instance parsed from the protobuf response.

Raises **ValueError** – if the instance name does not match `projects/{project}/instances/{instance_id}` or if the parsed project ID does not match the project ID on the client.

list_databases(*page_size=None*, *page_token=None*)

List databases for the instance.

See <https://cloud.google.com/spanner/reference/rpc/google.spanner.admin.database.v1#google.spanner.admin.database.v1.DatabaseAdmin.ListDatabases>

Parameters

- **page_size** (*int*) – (Optional) Maximum number of results to return.
- **page_token** (*str*) – (Optional) Token for fetching next page of results.

Return type *Iterator*

Returns Iterator of *Database* resources within the current instance.

name

Instance name used in requests.

Note: This property will not change if `instance_id` does not, but the return value is not cached.

The instance name is of the form

`"projects/{project}/instances/{instance_id}"`

Return type *str*

Returns The instance name.

reload()

Reload the metadata for this instance.

See <https://cloud.google.com/spanner/reference/rpc/google.spanner.admin.instance.v1#google.spanner.admin.instance.v1.InstanceAdmin.GetInstanceConfig>

Raises

- **NotFound** – if the instance does not exist
- **GaxError** – for other errors returned from the call

update()

Update this instance.

See <https://cloud.google.com/spanner/reference/rpc/google.spanner.admin.instance.v1#google.spanner.admin.instance.v1.InstanceAdmin.UpdateInstance>

Note: Updates the `display_name` and `node_count`. To change those values before updating, set them via

```
instance.display_name = 'New display name'
instance.node_count = 5
```

```
before calling :meth:`update`.
```

Return type `google.api.core.operation.Operation`

Returns an operation instance

Raises

- **NotFound** – if the instance does not exist
- **GaxError** – for other errors returned from the call

13.11 Database API

User friendly container for Cloud Spanner Database.

class `google.cloud.spanner.database.BatchCheckout` (*database*)

Bases: `object`

Context manager for using a batch from a database.

Inside the context manager, checks out a session from the database, creates a batch from it, making the batch available.

Caller must *not* use the batch to perform API requests outside the scope of the context manager.

Parameters `database` (`Database`) – database to use

class `google.cloud.spanner.database.Database` (*database_id*, *instance*, *ddl_statements=()*,
pool=None)

Bases: `object`

Representation of a Cloud Spanner Database.

We can use a `Database` to:

- `create()` the database
- `reload()` the database
- `update()` the database
- `drop()` the database

Parameters

- **database_id** (*str*) – The ID of the database.

- **instance** (*Instance*) – The instance that owns the database.
- **ddl_statements** (*list of string*) – (Optional) DDL statements, excluding the CREATE DATABASE statement.
- **pool** (concrete subclass of *AbstractSessionPool*.) – (Optional) session pool to be used by database. If not passed, the database will construct an instance of *BurstyPool*.

batch()

Return an object which wraps a batch.

The wrapper *must* be used as a context manager, with the batch as the value returned by the wrapper.

Return type *BatchCheckout*

Returns new wrapper

create()

Create this database within its instance

Includes any configured schema assigned to *ddl_statements*.

See <https://cloud.google.com/spanner/reference/rpc/google.spanner.admin.database.v1#google.spanner.admin.database.v1.DatabaseAdmin.CreateDatabase>

Return type *Operation*

Returns a future used to poll the status of the create request

Raises

- **Conflict** – if the database already exists
- **NotFound** – if the instance owning the database does not exist
- **GaxError** – for errors other than ALREADY_EXISTS returned from the call

ddl_statements

DDL Statements used to define database schema.

See cloud.google.com/spanner/docs/data-definition-language

Return type sequence of string

Returns the statements

drop()

Drop this database.

See <https://cloud.google.com/spanner/reference/rpc/google.spanner.admin.database.v1#google.spanner.admin.database.v1.DatabaseAdmin.DropDatabase>

exists()

Test whether this database exists.

See <https://cloud.google.com/spanner/reference/rpc/google.spanner.admin.database.v1#google.spanner.admin.database.v1.DatabaseAdmin.GetDatabaseDDL>

Return type *bool*

Returns True if the database exists, else false.

Raises **GaxError** – for errors other than NOT_FOUND returned from the call

classmethod from_pb(database_pb, instance, pool=None)

Creates an instance of this class from a protobuf.

Parameters

- **database_pb** (`google.spanner.v2.spanner_instance_admin_pb2.Instance`) – A instance protobuf object.
- **instance** (*Instance*) – The instance that owns the database.
- **pool** (concrete subclass of *AbstractSessionPool*.) – (Optional) session pool to be used by database.

Return type *Database***Returns** The database parsed from the protobuf response.**Raises** **ValueError** – if the instance name does not match the expected format or if the parsed project ID does not match the project ID on the instance’s client, or if the parsed instance ID does not match the instance’s ID.**name**

Database name used in requests.

Note: This property will not change if `database_id` does not, but the return value is not cached.

The database name is of the form

```
"projects/.../instances/.../databases/{database_id}"
```

Return type *str***Returns** The database name.**reload()**

Reload this database.

Refresh any configured schema into *ddl_statements*.See <https://cloud.google.com/spanner/reference/rpc/google.spanner.admin.database.v1#google.spanner.admin.database.v1.DatabaseAdmin.GetDatabaseDDL>**Raises**

- **NotFound** – if the database does not exist
- **GaxError** – for errors other than `NOT_FOUND` returned from the call

run_in_transaction(func, *args, **kw)

Perform a unit of work in a transaction, retrying on abort.

Parameters

- **func** (*callable*) – takes a required positional argument, the transaction, and additional positional / keyword arguments as supplied by the caller.
- **args** (*tuple*) – additional positional arguments to be passed to *func*.
- **kw** (*dict*) – optional keyword arguments to be passed to *func*. If passed, “*timeout_secs*” will be removed and used to override the default timeout.

Return type *datetime.datetime***Returns** timestamp of committed transaction**session()**

Factory to create a session for this database.

Return type *Session*

Returns a session bound to this database.

snapshot (***kw*)

Return an object which wraps a snapshot.

The wrapper *must* be used as a context manager, with the snapshot as the value returned by the wrapper.

See <https://cloud.google.com/spanner/reference/rpc/google.spanner.v1#google.spanner.v1.TransactionOptions.ReadOnly>

Parameters **kw** (*dict*) – Passed through to *Snapshot* constructor.

Return type *SnapshotCheckout*

Returns new wrapper

spanner_api

Helper for session-related API calls.

update_ddl (*ddl_statements*)

Update DDL for this database.

Apply any configured schema from *ddl_statements*.

See <https://cloud.google.com/spanner/reference/rpc/google.spanner.admin.database.v1#google.spanner.admin.database.v1.DatabaseAdmin.UpdateDatabase>

Return type *google.api.core.operation.Operation*

Returns an operation instance

Raises

- **NotFound** – if the database does not exist
- **GaxError** – for errors other than NOT_FOUND returned from the call

class *google.cloud.spanner.database.SnapshotCheckout* (*database*, ***kw*)

Bases: *object*

Context manager for using a snapshot from a database.

Inside the context manager, checks out a session from the database, creates a snapshot from it, making the snapshot available.

Caller must *not* use the snapshot to perform API requests outside the scope of the context manager.

Parameters

- **database** (*Database*) – database to use
- **kw** (*dict*) – Passed through to *Snapshot* constructor.

13.12 Session API

Wrapper for Cloud Spanner Session objects.

google.cloud.spanner.session.DEFAULT_RETRY_TIMEOUT_SECS = 30

Default timeout used by *Session.run_in_transaction()*.

class google.cloud.spanner.session.**Session** (*database*)

Bases: `object`

Representation of a Cloud Spanner Session.

We can use a *Session* to:

- `create()` the session
- Use `exists()` to check for the existence of the session
- `drop()` the session

Parameters `database` (*Database*) – The database to which the session is bound.

batch ()

Factory to create a batch for this session.

Return type *Batch*

Returns a batch bound to this session

Raises **ValueError** – if the session has not yet been created.

create ()

Create this session, bound to its database.

See <https://cloud.google.com/spanner/reference/rpc/google.spanner.v1#google.spanner.v1.Spanner.CreateSession>

Raises **ValueError** if `session_id` is already set.

delete ()

Delete this session.

See <https://cloud.google.com/spanner/reference/rpc/google.spanner.v1#google.spanner.v1.Spanner.GetSession>

Raises

- **ValueError** – if `session_id` is not already set.
- **NotFound** – if the session does not exist
- **GaxError** – for errors other than `NOT_FOUND` returned from the call

execute_sql (*sql*, *params=None*, *param_types=None*, *query_mode=None*, *resume_token=""*)

Perform an `ExecuteStreamingSql` API request.

Parameters

- **sql** (*str*) – SQL query statement
- **params** (*dict*, {*str* -> *column value*}) – values for parameter replacement. Keys must match the names used in `sql`.
- **param_types** (*dict*, {*str* -> `google.spanner.v1.type_pb2.TypeCode`}) – (Optional) explicit types for one or more param values; overrides default type detection on the back-end.
- **query_mode** (`google.spanner.v1.spanner_pb2.ExecuteSqlRequest.QueryMode`) – Mode governing return of results / query plan. See <https://cloud.google.com/spanner/reference/rpc/google.spanner.v1#google.spanner.v1.ExecuteSqlRequest.QueryMode>
- **resume_token** (*bytes*) – token for resuming previously-interrupted query

Return type *StreamedResultSet*

Returns a result set instance which can be used to consume rows.

exists()

Test for the existence of this session.

See <https://cloud.google.com/spanner/reference/rpc/google.spanner.v1#google.spanner.v1.Spanner.GetSession>

Return type *bool*

Returns True if the session exists on the back-end, else False.

Raises **GaxError** – for errors other than NOT_FOUND returned from the call

name

Session name used in requests.

Note: This property will not change if `session_id` does not, but the return value is not cached.

The session name is of the form

"projects/.../instances/.../databases/.../sessions/{session_id}"

Return type *str*

Returns The session name.

Raises **ValueError** – if session is not yet created

read(table, columns, keyset, index="", limit=0, resume_token="")

Perform a `StreamingRead` API request for rows in a table.

Parameters

- **table** (*str*) – name of the table from which to fetch data
- **columns** (*list of str*) – names of columns to be retrieved
- **keyset** (*KeySet*) – keys / ranges identifying rows to be retrieved
- **index** (*str*) – (Optional) name of index to use, rather than the table's primary key
- **limit** (*int*) – (Optional) maximum number of rows to return
- **resume_token** (*bytes*) – token for resuming previously-interrupted read

Return type *StreamedResultSet*

Returns a result set instance which can be used to consume rows.

run_in_transaction(func, *args, **kw)

Perform a unit of work in a transaction, retrying on abort.

Parameters

- **func** (*callable*) – takes a required positional argument, the transaction, and additional positional / keyword arguments as supplied by the caller.
- **args** (*tuple*) – additional positional arguments to be passed to `func`.
- **kw** (*dict*) – optional keyword arguments to be passed to `func`. If passed, "timeout_secs" will be removed and used to override the default timeout.

Return type Any

Returns The return value of `func`.

Raises **Exception** – reraises any non-ABORT exceptions raised by `func`.

session_id

Read-only ID, set by the back-end during `create()`.

snapshot (***kw*)

Create a snapshot to perform a set of reads with shared staleness.

See <https://cloud.google.com/spanner/reference/rpc/google.spanner.v1#google.spanner.v1.TransactionOptions.ReadOnly>

Parameters **kw** (*dict*) – Passed through to `Snapshot` ctor.

Return type `Snapshot`

Returns a snapshot bound to this session

Raises **ValueError** – if the session has not yet been created.

transaction ()

Create a transaction to perform a set of reads with shared staleness.

Return type `Transaction`

Returns a transaction bound to this session

Raises **ValueError** – if the session has not yet been created.

13.13 Session Pools API

Pools managing shared Session objects.

class `google.cloud.spanner.pool.AbstractSessionPool`

Bases: `object`

Specifies required API for concrete session pool implementations.

bind (*database*)

Associate the pool with a database.

Parameters **database** (*Database*) – database used by the pool: used to create sessions when needed.

Concrete implementations of this method may pre-fill the pool using the database.

Raises **NotImplementedError** – abstract method

clear ()

Delete all sessions in the pool.

Concrete implementations of this method are allowed to raise an error to signal that the pool is full, or to block until it is not full.

Raises **NotImplementedError** – abstract method

get ()

Check a session out from the pool.

Concrete implementations of this method are allowed to raise an error to signal that the pool is exhausted, or to block until a session is available.

Raises `NotImplementedError` – abstract method

put (*session*)

Return a session to the pool.

Parameters *session* (*Session*) – the session being returned.

Concrete implementations of this method are allowed to raise an error to signal that the pool is full, or to block until it is not full.

Raises `NotImplementedError` – abstract method

session (***kwargs*)

Check out a session from the pool.

Parameters *kwargs* (*dict*) – (optional) keyword arguments, passed through to the returned checkout.

Return type *SessionCheckout*

Returns a checkout instance, to be used as a context manager for accessing the session and returning it to the pool.

class `google.cloud.spanner.pool.BurstyPool` (*target_size=10*)

Bases: `google.cloud.spanner.pool.AbstractSessionPool`

Concrete session pool implementation:

- “Pings” existing sessions via `session.exists()` before returning them.
- Creates a new session, rather than blocking, when `get()` is called on an empty pool.
- Discards the returned session, rather than blocking, when `put()` is called on a full pool.

Parameters *target_size* (*int*) – max pool size

bind (*database*)

Associate the pool with a database.

Parameters *database* (*Database*) – database used by the pool: used to create sessions when needed.

clear ()

Delete all sessions in the pool.

get ()

Check a session out from the pool.

Return type *Session*

Returns an existing session from the pool, or a newly-created session.

put (*session*)

Return a session to the pool.

Never blocks: if the pool is full, the returned session is discarded.

Parameters *session* (*Session*) – the session being returned.

class `google.cloud.spanner.pool.FixedSizePool` (*size=10, default_timeout=10*)

Bases: `google.cloud.spanner.pool.AbstractSessionPool`

Concrete session pool implementation:

- Pre-allocates / creates a fixed number of sessions.

- “Pings” existing sessions via `session.exists()` before returning them, and replaces expired sessions.
- Blocks, with a timeout, when `get()` is called on an empty pool. Raises after timing out.
- Raises when `put()` is called on a full pool. That error is never expected in normal practice, as users should be calling `get()` followed by `put()` whenever in need of a session.

Parameters

- **size** (*int*) – fixed pool size
- **default_timeout** (*int*) – default timeout, in seconds, to wait for a returned session.

bind (*database*)

Associate the pool with a database.

Parameters **database** (*Database*) – database used by the pool: used to create sessions when needed.

clear ()

Delete all sessions in the pool.

get (*timeout=None*)

Check a session out from the pool.

Parameters **timeout** (*int*) – seconds to block waiting for an available session

Return type *Session*

Returns an existing session from the pool, or a newly-created session.

Raises `six.moves.queue.Empty` if the queue is empty.

put (*session*)

Return a session to the pool.

Never blocks: if the pool is full, raises.

Parameters **session** (*Session*) – the session being returned.

Raises `six.moves.queue.Full` if the queue is full.

class `google.cloud.spanner.pool.PingingPool` (*size=10*, *default_timeout=10*,
ping_interval=3000)

Bases: `google.cloud.spanner.pool.AbstractSessionPool`

Concrete session pool implementation:

- Pre-allocates / creates a fixed number of sessions.
- Sessions are used in “round-robin” order (LRU first).
- “Pings” existing sessions in the background after a specified interval via an API call (`session.exists()`).
- Blocks, with a timeout, when `get()` is called on an empty pool. Raises after timing out.
- Raises when `put()` is called on a full pool. That error is never expected in normal practice, as users should be calling `get()` followed by `put()` whenever in need of a session.

The application is responsible for calling `ping()` at appropriate times, e.g. from a background thread.

Parameters

- **size** (*int*) – fixed pool size
- **default_timeout** (*int*) – default timeout, in seconds, to wait for a returned session.

- **ping_interval** (*int*) – interval at which to ping sessions.

bind (*database*)

Associate the pool with a database.

Parameters **database** (*Database*) – database used by the pool: used to create sessions when needed.

clear ()

Delete all sessions in the pool.

get (*timeout=None*)

Check a session out from the pool.

Parameters **timeout** (*int*) – seconds to block waiting for an available session

Return type *Session*

Returns an existing session from the pool, or a newly-created session.

Raises `six.moves.queue.Empty` if the queue is empty.

ping ()

Refresh maybe-expired sessions in the pool.

This method is designed to be called from a background thread, or during the “idle” phase of an event loop.

put (*session*)

Return a session to the pool.

Never blocks: if the pool is full, raises.

Parameters **session** (*Session*) – the session being returned.

Raises `six.moves.queue.Full` if the queue is full.

class `google.cloud.spanner.pool.SessionCheckout` (*pool, **kwargs*)

Bases: `object`

Context manager: hold session checked out from a pool.

Parameters

- **pool** (concrete subclass of `AbstractSessionPool`) – Pool from which to check out a session.
- **kwargs** (*dict*) – extra keyword arguments to be passed to `pool.get()`.

class `google.cloud.spanner.pool.TransactionPingingPool` (*size=10, fault_timeout=10, ping_interval=3000*) *de-*

Bases: `google.cloud.spanner.pool.PingingPool`

Concrete session pool implementation:

In addition to the features of `PingingPool`, this class creates and begins a transaction for each of its sessions at startup.

When a session is returned to the pool, if its transaction has been committed or rolled back, the pool creates a new transaction for the session and pushes the transaction onto a separate queue of “transactions to begin.” The application is responsible for flushing this queue as appropriate via the pool’s `begin_pending_transactions()` method.

Parameters

- **size** (*int*) – fixed pool size

- **default_timeout** (*int*) – default timeout, in seconds, to wait for a returned session.
- **ping_interval** (*int*) – interval at which to ping sessions.

begin_pending_transactions ()

Begin all transactions for sessions added to the pool.

bind (*database*)

Associate the pool with a database.

Parameters **database** (*Database*) – database used by the pool: used to create sessions when needed.

put (*session*)

Return a session to the pool.

Never blocks: if the pool is full, raises.

Parameters **session** (*Session*) – the session being returned.

Raises `six.moves.queue.Full` if the queue is full.

13.14 Keyset API

Wrap representation of Spanner keys / ranges.

class google.cloud.spanner.keyset.**KeyRange** (*start_open=None, start_closed=None, end_open=None, end_closed=None*)

Bases: `object`

Identify range of table rows via start / end points.

Parameters

- **start_open** (*list of scalars*) – keys identifying start of range (this key excluded)
- **start_closed** (*list of scalars*) – keys identifying start of range (this key included)
- **end_open** (*list of scalars*) – keys identifying end of range (this key excluded)
- **end_closed** (*list of scalars*) – keys identifying end of range (this key included)

to_pb ()

Construct a KeyRange protobuf.

Return type `KeyRange`

Returns protobuf corresponding to this instance.

class google.cloud.spanner.keyset.**KeySet** (*keys=(), ranges=(), all=False*)

Bases: `object`

Identify table rows via keys / ranges.

Parameters

- **keys** (*list of list of scalars*) – keys identifying individual rows within a table.
- **ranges** (*list of KeyRange*) – ranges identifying rows within a table.
- **all** (*boolean*) – if True, identify all rows within a table

to_pb ()

Construct a KeySet protobuf.

Return type `KeySet`

Returns `protobuf` corresponding to this instance.

13.15 Snapshot API

Model a set of read-only queries to a database as a snapshot.

```
class google.cloud.spanner.snapshot.Snapshot (session, read_timestamp=None,
                                              min_read_timestamp=None,
                                              max_staleness=None, ex-
                                              act_staleness=None, multi_use=False)
```

Bases: `google.cloud.spanner.snapshot._SnapshotBase`

Allow a set of reads / SQL statements with shared staleness.

See <https://cloud.google.com/spanner/reference/rpc/google.spanner.v1#google.spanner.v1.TransactionOptions.ReadOnly>

If no options are passed, reads will use the `strong` model, reading at a timestamp where all previously committed transactions are visible.

Parameters

- **session** (`Session`) – the session used to perform the commit.
- **read_timestamp** (`datetime.datetime`) – Execute all reads at the given timestamp.
- **min_read_timestamp** (`datetime.datetime`) – Execute all reads at a timestamp \geq `min_read_timestamp`.
- **max_staleness** (`datetime.timedelta`) – Read data at a timestamp \geq NOW - `max_staleness` seconds.
- **exact_staleness** (`datetime.timedelta`) – Execute all reads at a timestamp that is `exact_staleness` old.
- **multi_use** (`bool`) – If true, multiple `read()` / `execute_sql()` calls can be performed with the snapshot in the context of a read-only transaction, used to ensure isolation / consistency. Incompatible with `max_staleness` and `min_read_timestamp`.

begin()

Begin a read-only transaction on the database.

Return type `bytes`

Returns the ID for the newly-begun transaction.

Raises `ValueError` – if the transaction is already begun, committed, or rolled back.

13.16 Batch API

Context manager for Cloud Spanner batched writes.

```
class google.cloud.spanner.batch.Batch (session)
    Bases: google.cloud.spanner.batch._BatchBase
    Accumulate mutations for transmission during commit().
```

commit()
Commit mutations to the database.

Return type `datetime`

Returns timestamp of the committed changes.

committed = None
Timestamp at which the batch was successfully committed.

13.17 Transaction API

Spanner read-write transaction support.

class `google.cloud.spanner.transaction.Transaction` (*session*)
Bases: `google.cloud.spanner.snapshot._SnapshotBase`, `google.cloud.spanner.batch._BatchBase`

Implement read-write transaction semantics for a session.

Parameters *session* (*Session*) – the session used to perform the commit

Raises `ValueError` – if session has an existing transaction

begin()
Begin a transaction on the database.

Return type `bytes`

Returns the ID for the newly-begun transaction.

Raises `ValueError` – if the transaction is already begun, committed, or rolled back.

commit()
Commit mutations to the database.

Return type `datetime`

Returns timestamp of the committed changes.

Raises `ValueError` – if there are no mutations to commit.

committed = None
Timestamp at which the transaction was successfully committed.

rollback()
Roll back a transaction on the database.

13.18 StreamedResultSet API

Wrapper for streaming results.

class `google.cloud.spanner.streamed.StreamedResultSet` (*response_iterator*,
source=None)
Bases: `object`

Process a sequence of partial result sets into a single set of row data.

Parameters

- **response_iterator** – Iterator yielding `google.cloud.proto.spanner.v1.result_set_pb2.PartialResultSet` instances.
- **source** (*Snapshot*) – Snapshot from which the result set was fetched.

consume_all()

Consume the streamed responses until there are no more.

consume_next()

Consume the next partial result set from the stream.

Parse the result set into new/existing rows in `_rows`

fields

Field descriptors for result set columns.

Return type list of `Field`

Returns list of fields describing column names / types.

metadata

Result set metadata

Return type `ResultSetMetadata`

Returns structure describing the results

one()

Return exactly one result, or raise an exception.

Raises `NotFound`: If there are no results.

Raises `ValueError`: If there are multiple results.

Raises `RuntimeError`: If consumption has already occurred, in whole or in part.

one_or_none()

Return exactly one result, or `None` if there are no results.

Raises `ValueError`: If there are multiple results.

Raises `RuntimeError`: If consumption has already occurred, in whole or in part.

resume_token

Token for resuming interrupted read / query.

Return type `bytes`

Returns token from last chunk of results.

rows

Fully-processed rows.

Return type list of row-data lists.

Returns list of completed row data, from processed PRS responses.

stats

Result set statistics

Return type `ResultSetStats`

Returns structure describing status about the response

exception `google.cloud.spanner.streamed.Unmergeable` (*lhs, rhs, type_*)

Bases: `exceptions.ValueError`

Unable to merge two values.

Parameters

- **lhs** (`google.protobuf.struct_pb2.Value`) – pending value to be merged
- **rhs** (`google.protobuf.struct_pb2.Value`) – remaining value to be merged
- **type** (`google.cloud.proto.spanner.v1.type_pb2.Type`) – field type of values being merged

API requests are sent to the [Cloud Spanner](#) API via RPC over HTTP/2. In order to support this, we'll rely on [gRPC](#).

Get started by learning about the [Client](#) on the [Client](#) page.

In the hierarchy of API concepts

- a [Client](#) owns an [Instance](#)
- an [Instance](#) owns a [Database](#)

The [Google Speech](#) API enables developers to convert audio to text. The API recognizes over 80 languages and variants, to support your global user base.

14.1 Authentication and Configuration

SpeechClient objects provide a means to configure your application. Each instance holds an authenticated connection to the Cloud Speech Service.

For an overview of authentication in `google-cloud-python`, see [Authentication](#).

Assuming your environment is set up as described in that document, create an instance of *SpeechClient*.

```
>>> from google.cloud import speech
>>> client = speech.SpeechClient()
```

14.2 Asynchronous Recognition

The *long_running_recognize()* method sends audio data to the Speech API and initiates a Long Running Operation.

Using this operation, you can periodically poll for recognition results. Use asynchronous requests for audio data of any duration up to 80 minutes.

See: [Speech Asynchronous Recognize](#)

```
>>> import time
>>> from google.cloud import speech
>>> client = speech.SpeechClient()
>>> operation = client.long_running_recognize(
...     audio=speech.types.RecognitionAudio(
...         uri='gs://my-bucket/recording.flac',
```

```

...     ),
...     config=speech.types.RecognitionConfig(
...         encoding='LINEAR16',
...         language_code='en-US',
...         sample_rate_hertz=44100,
...     ),
... )
>>> retry_count = 100
>>> while retry_count > 0 and not operation.complete:
...     retry_count -= 1
...     time.sleep(10)
...     operation.poll() # API call
>>> operation.complete
True
>>> for result in operation.results:
...     for alternative in result.alternatives:
...         print('=' * 20)
...         print(alternative.transcript)
...         print(alternative.confidence)
=====
'how old is the Brooklyn Bridge'
0.98267895

```

14.3 Synchronous Recognition

The `recognize()` method converts speech data to text and returns alternative text transcriptions.

This example uses `language_code='en-GB'` to better recognize a dialect from Great Britain.

```

>>> from google.cloud import speech
>>> client = speech.SpeechClient()
>>> results = client.recognize(
...     audio=speech.types.RecognitionAudio(
...         uri='gs://my-bucket/recording.flac',
...     ),
...     config=speech.types.RecognitionConfig(
...         encoding='LINEAR16',
...         language_code='en-US',
...         sample_rate_hertz=44100,
...     ),
... )
>>> for result in results:
...     for alternative in result.alternatives:
...         print('=' * 20)
...         print('transcript: ' + alternative.transcript)
...         print('confidence: ' + str(alternative.confidence))
=====
transcript: Hello, this is a test
confidence: 0.81
=====
transcript: Hello, this is one test
confidence: 0

```

Example of using the profanity filter.

```
>>> from google.cloud import speech
>>> client = speech.SpeechClient()
>>> results = client.recognize(
...     audio=speech.types.RecognitionAudio(
...         uri='gs://my-bucket/recording.flac',
...     ),
...     config=speech.types.RecognitionConfig(
...         encoding='LINEAR16',
...         language_code='en-US',
...         profanity_filter=True,
...         sample_rate_hertz=44100,
...     ),
... )
>>> for result in results:
...     for alternative in result.alternatives:
...         print('=' * 20)
...         print('transcript: ' + alternative.transcript)
...         print('confidence: ' + str(alternative.confidence))
=====
transcript: Hello, this is a f***** test
confidence: 0.81
```

Using speech context hints to get better results. This can be used to improve the accuracy for specific words and phrases. This can also be used to add new words to the vocabulary of the recognizer.

```
>>> from google.cloud import speech
>>> from google.cloud import speech
>>> client = speech.SpeechClient()
>>> results = client.recognize(
...     audio=speech.types.RecognitionAudio(
...         uri='gs://my-bucket/recording.flac',
...     ),
...     config=speech.types.RecognitionConfig(
...         encoding='LINEAR16',
...         language_code='en-US',
...         sample_rate_hertz=44100,
...         speech_contexts=[speech.types.SpeechContext(
...             phrases=['hi', 'good afternoon'],
...         )],
...     ),
... )
>>> for result in results:
...     for alternative in result.alternatives:
...         print('=' * 20)
...         print('transcript: ' + alternative.transcript)
...         print('confidence: ' + str(alternative.confidence))
=====
transcript: Hello, this is a test
confidence: 0.81
```

14.4 Streaming Recognition

The `streaming_recognize()` method converts speech data to possible text alternatives on the fly.

Note: Streaming recognition requests are limited to 1 minute of audio.

See: <https://cloud.google.com/speech/limits#content>

```
>>> import io
>>> from google.cloud import speech
>>> client = speech.SpeechClient()
>>> config = speech.types.RecognitionConfig(
...     encoding='LINEAR16',
...     language_code='en-US',
...     sample_rate_hertz=44100,
... )
>>> with io.open('./hello.wav', 'rb') as stream:
...     requests = [speech.types.StreamingRecognizeRequest(
...         audio_content=stream.read(),
...     )]
>>> results = sample.streaming_recognize(
...     config=speech.types.StreamingRecognitionConfig(config=config),
...     requests,
... )
>>> for result in results:
...     for alternative in result.alternatives:
...         print('=' * 20)
...         print('transcript: ' + alternative.transcript)
...         print('confidence: ' + str(alternative.confidence))
=====
transcript: hello thank you for using Google Cloud platform
confidence: 0.927983105183
```

By default the API will perform continuous recognition (continuing to process audio even if the speaker in the audio pauses speaking) until the client closes the output stream or until the maximum time limit has been reached.

If you only want to recognize a single utterance you can set `single_utterance` to `True` and only one result will be returned.

See: [Single Utterance](#)

```
>>> import io
>>> from google.cloud import speech
>>> client = speech.SpeechClient()
>>> config = speech.types.RecognitionConfig(
...     encoding='LINEAR16',
...     language_code='en-US',
...     sample_rate_hertz=44100,
... )
>>> with io.open('./hello-pause-goodbye.wav', 'rb') as stream:
...     requests = [speech.types.StreamingRecognizeRequest(
...         audio_content=stream.read(),
...     )]
>>> results = sample.streaming_recognize(
...     config=speech.types.StreamingRecognitionConfig(
...         config=config,
...         single_utterance=False,
...     ),
...     requests,
... )
>>> for result in results:
...     for alternative in result.alternatives:
...         print('=' * 20)
...         print('transcript: ' + alternative.transcript)
```

```

...     print('confidence: ' + str(alternative.confidence))
...     for result in results:
...         for alternative in result.alternatives:
...             print('=' * 20)
...             print('transcript: ' + alternative.transcript)
...             print('confidence: ' + str(alternative.confidence))
=====
transcript: testing a pause
confidence: 0.933770477772

```

If `interim_results` is set to `True`, interim results (tentative hypotheses) may be returned as they become available.

```

>>> import io
>>> from google.cloud import speech
>>> client = speech.SpeechClient()
>>> config = speech.types.RecognitionConfig(
...     encoding='LINEAR16',
...     language_code='en-US',
...     sample_rate_hertz=44100,
... )
>>> with io.open('./hello.wav', 'rb') as stream:
...     requests = [speech.types.StreamingRecognizeRequest(
...         audio_content=stream.read(),
...     )]
>>> config = speech.types.StreamingRecognitionConfig(config=config)
>>> responses = client.streaming_recognize(config, requests)
>>> for response in responses:
...     for result in response:
...         for alternative in result.alternatives:
...             print('=' * 20)
...             print('transcript: ' + alternative.transcript)
...             print('confidence: ' + str(alternative.confidence))
...             print('is_final:' + str(result.is_final))
=====
'he'
None
False
=====
'hell'
None
False
=====
'hello'
0.973458576
True

```

14.5 API Reference

14.5.1 Speech Client API

```
class google.cloud.speech_v1.SpeechClient (service_path='speech.googleapis.com',
                                           port=443, channel=None, credentials=None,
                                           ssl_credentials=None, scopes=None,
                                           client_config=None, app_name=None,
                                           app_version="", lib_name=None,
                                           lib_version="", metrics_headers=())
```

Service that implements Google Cloud Speech API.

Constructor.

Parameters

- **service_path** (*string*) – The domain name of the API remote host.
- **port** (*int*) – The port on which to connect to the remote host.
- **channel** (*grpc.Channel*) – A `Channel` instance through which to make calls.
- **credentials** (*object*) – The authorization credentials to attach to requests. These credentials identify this application to the service.
- **ssl_credentials** (*grpc.ChannelCredentials*) – A `ChannelCredentials` instance for use with an SSL-enabled channel.
- **scopes** (*list[string]*) – A list of OAuth2 scopes to attach to requests.
- **client_config** (*dict*) – A dictionary for call options for each method. See `google.gax.construct_settings()` for the structure of this data. Falls back to the default config if not specified or the specified config is missing data points.
- **app_name** (*string*) – The name of the application calling the service. Recommended for analytics purposes.
- **app_version** (*string*) – The version of the application calling the service. Recommended for analytics purposes.
- **lib_name** (*string*) – The API library software used for calling the service. (Unless you are writing an API client itself, leave this as default.)
- **lib_version** (*string*) – The API library software version used for calling the service. (Unless you are writing an API client itself, leave this as default.)
- **metrics_headers** (*dict*) – A dictionary of values for tracking client library metrics. Ultimately serializes to a string (e.g. 'foo/1.2.3 bar/3.14.1'). This argument should be considered private.

Returns A `SpeechClient` object.

```
enums = <module 'google.cloud.gapic.speech.v1.enums' from '/home/docs/checkouts/readth
```

```
long_running_recognize (config, audio, options=None)
```

Performs asynchronous speech recognition: receive results via the `google.longrunning.Operations` interface. Returns either an `Operation.error` or an `Operation.response` which contains a `LongRunningRecognizeResponse` message.

Example

```

>>> from google.cloud.gapic.speech.v1 import speech_client
>>> from google.cloud.gapic.speech.v1 import enums
>>> from google.cloud.proto.speech.v1 import cloud_speech_pb2
>>> client = speech_client.SpeechClient()
>>> encoding = enums.RecognitionConfig.AudioEncoding.FLAC
>>> sample_rate_hertz = 44100
>>> language_code = 'en-US'
>>> config = cloud_speech_pb2.RecognitionConfig(encoding=encoding, sample_
↳ rate_hertz=sample_rate_hertz, language_code=language_code)
>>> uri = 'gs://bucket_name/file_name.flac'
>>> audio = cloud_speech_pb2.RecognitionAudio(uri=uri)
>>> response = client.long_running_recognize(config, audio)
>>>
>>> def callback(operation_future):
>>>     # Handle result.
>>>     result = operation_future.result()
>>>
>>> response.add_done_callback(callback)
>>>
>>> # Handle metadata.
>>> metadata = response.metadata()

```

Parameters

- **config** (google.cloud.proto.speech.v1.cloud_speech_pb2.RecognitionConfig) – *Required* Provides information to the recognizer that specifies how to process the request.
- **audio** (google.cloud.proto.speech.v1.cloud_speech_pb2.RecognitionAudio) – *Required* The audio data to be recognized.
- **options** (google.gax.CallOptions) – Overrides the default settings for this call, e.g. timeout, retries etc.

Returns A google.gax._OperationFuture instance.

Raises

- google.gax.errors.GaxError if the RPC is aborted.
- ValueError if the parameters are invalid.

recognize (config, audio, options=None)

Performs synchronous speech recognition: receive results after all audio has been sent and processed.

Example

```

>>> from google.cloud.gapic.speech.v1 import speech_client
>>> from google.cloud.gapic.speech.v1 import enums
>>> from google.cloud.proto.speech.v1 import cloud_speech_pb2
>>> client = speech_client.SpeechClient()
>>> encoding = enums.RecognitionConfig.AudioEncoding.FLAC
>>> sample_rate_hertz = 44100
>>> language_code = 'en-US'
>>> config = cloud_speech_pb2.RecognitionConfig(encoding=encoding, sample_
↳ rate_hertz=sample_rate_hertz, language_code=language_code)

```

```
>>> uri = 'gs://bucket_name/file_name.flac'
>>> audio = cloud_speech_pb2.RecognitionAudio(uri=uri)
>>> response = client.recognize(config, audio)
```

Parameters

- **config** (`google.cloud.proto.speech.v1.cloud_speech_pb2.RecognitionConfig`) – *Required* Provides information to the recognizer that specifies how to process the request.
- **audio** (`google.cloud.proto.speech.v1.cloud_speech_pb2.RecognitionAudio`) – *Required* The audio data to be recognized.
- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A `google.cloud.proto.speech.v1.cloud_speech_pb2.RecognizeResponse` instance.

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

streaming_recognize (*config, requests, options=None*)

Perform bi-directional speech recognition.

This method allows you to receive results while sending audio; it is only available via. gRPC (not REST).

Warning: This method is EXPERIMENTAL. Its interface might change in the future.

Example

```
>>> from google.cloud.speech_v1 import enums
>>> from google.cloud.speech_v1 import SpeechClient
>>> from google.cloud.speech_v1 import types
>>> client = SpeechClient()
>>> config = types.StreamingRecognitionConfig(
...     config=types.RecognitionConfig(
...         encoding=enums.RecognitionConfig.AudioEncoding.FLAC,
...     ),
... )
>>> request = types.StreamingRecognizeRequest(audio_content=b'...')
>>> requests = [request]
>>> for element in client.streaming_recognize(config, requests):
...     # process element
...     pass
```

Parameters

- **config** (`StreamingRecognitionConfig`) – The configuration to use for the stream.
- **requests** (`Iterable[StreamingRecognizeRequest]`) – The input objects.

- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns `Iterable[StreamingRecognizeResponse]`

Raises

- `google.gax.errors.GaxError` if the RPC is aborted.
- `ValueError` if the parameters are invalid.

14.5.2 Speech Client Types

class `google.cloud.speech_v1.types.LongRunningRecognizeMetadata`

Describes the progress of a long-running `LongRunningRecognize` call. It is included in the `metadata` field of the `Operation` returned by the `GetOperation` call of the `google::longrunning::Operations` service.

progress_percent

Approximate percentage of audio processed thus far. Guaranteed to be 100 when the audio is fully processed and the results are available.

start_time

Time when the request was received.

last_update_time

Time of the most recent processing update.

class `google.cloud.speech_v1.types.LongRunningRecognizeRequest`

The top-level message sent by the client for the `LongRunningRecognize` method.

config

Required Provides information to the recognizer that specifies how to process the request.

audio

Required The audio data to be recognized.

class `google.cloud.speech_v1.types.LongRunningRecognizeResponse`

The only message returned to the client by the `LongRunningRecognize` method. It contains the result as zero or more sequential `SpeechRecognitionResult` messages. It is included in the `result.response` field of the `Operation` returned by the `GetOperation` call of the `google::longrunning::Operations` service.

results

Output-only Sequential list of transcription results corresponding to sequential portions of audio.

class `google.cloud.speech_v1.types.RecognitionAudio`

Contains audio data in the encoding specified in the `RecognitionConfig`. Either `content` or `uri` must be supplied. Supplying both or neither returns `[google.rpc.Code.INVALID_ARGUMENT][google.rpc.Code.INVALID_ARGUMENT]`. See [audio limits](#).

audio_source

The audio source, which is either inline content or a Google Cloud Storage uri.

content

The audio data bytes encoded as specified in `RecognitionConfig`. Note: as with all bytes fields, protobufs use a pure binary representation, whereas JSON representations use base64.

uri

URI that points to a file that contains audio data bytes as specified in `RecognitionConfig`.

Currently, only Google Cloud Storage URIs are supported, which must be specified in the following format: `gs://bucket_name/object_name` (other URI formats return `[google.rpc.Code.INVALID_ARGUMENT][google.rpc.Code.INVALID_ARGUMENT]`). For more information, see [Request URIs](#).

class `google.cloud.speech_v1.types.RecognitionConfig`

Provides information to the recognizer that specifies how to process the request.

encoding

Required Encoding of audio data sent in all `RecognitionAudio` messages.

sample_rate_hertz

Required Sample rate in Hertz of the audio data sent in all `RecognitionAudio` messages. Valid values are: 8000-48000. 16000 is optimal. For best results, set the sampling rate of the audio source to 16000 Hz. If that's not possible, use the native sample rate of the audio source (instead of re- sampling).

language_code

Required The language of the supplied audio as a [BCP-47](#) language tag. Example: "en-US". See [Language Support](#) for a list of the currently supported language codes.

max_alternatives

Optional Maximum number of recognition hypotheses to be returned. Specifically, the maximum number of `SpeechRecognitionAlternative` messages within each `SpeechRecognitionResult`. The server may return fewer than `max_alternatives`. Valid values are 0-30. A value of 0 or 1 will return a maximum of one. If omitted, will return a maximum of one.

profanity_filter

Optional If set to `true`, the server will attempt to filter out profanities, replacing all but the initial character in each filtered word with asterisks, e.g. "f***". If set to `false` or omitted, profanities won't be filtered out.

speech_contexts

Optional A means to provide context to assist the speech recognition.

enable_word_time_offsets

Optional If `true`, the top result includes a list of words and the start and end time offsets (timestamps) for those words. If `false`, no word-level time offset information is returned. The default is `false`.

class `google.cloud.speech_v1.types.RecognizeRequest`

The top-level message sent by the client for the `Recognize` method.

config

Required Provides information to the recognizer that specifies how to process the request.

audio

Required The audio data to be recognized.

class `google.cloud.speech_v1.types.RecognizeResponse`

The only message returned to the client by the `Recognize` method. It contains the result as zero or more sequential `SpeechRecognitionResult` messages.

results

Output-only Sequential list of transcription results corresponding to sequential portions of audio.

class `google.cloud.speech_v1.types.SpeechContext`

Provides "hints" to the speech recognizer to favor specific words and phrases in the results.

phrases

Optional A list of strings containing words and phrases "hints" so that the speech recognition is more likely to recognize them. This can be used to improve the accuracy for specific words and phrases, for

example, if specific commands are typically spoken by the user. This can also be used to add additional words to the vocabulary of the recognizer. See [usage limits](#).

class google.cloud.speech_v1.types.SpeechRecognitionAlternative

Alternative hypotheses (a.k.a. n-best list).

transcript

Output-only Transcript text representing the words that the user spoke.

confidence

Output-only The confidence estimate between 0.0 and 1.0. A higher number indicates an estimated greater likelihood that the recognized words are correct. This field is typically provided only for the top hypothesis, and only for `is_final=true` results. Clients should not rely on the `confidence` field as it is not guaranteed to be accurate or consistent. The default of 0.0 is a sentinel value indicating `confidence` was not set.

words

Output-only A list of word-specific information for each recognized word.

class google.cloud.speech_v1.types.SpeechRecognitionResult

A speech recognition result corresponding to a portion of the audio.

alternatives

Output-only May contain one or more recognition hypotheses (up to the maximum specified in `max_alternatives`). These alternatives are ordered in terms of accuracy, with the top (first) alternative being the most probable, as ranked by the recognizer.

class google.cloud.speech_v1.types.StreamingRecognitionConfig

Provides information to the recognizer that specifies how to process the request.

config

Required Provides information to the recognizer that specifies how to process the request.

single_utterance

Optional If `false` or omitted, the recognizer will perform continuous recognition (continuing to wait for and process audio even if the user pauses speaking) until the client closes the input stream (gRPC API) or until the maximum time limit has been reached. May return multiple `StreamingRecognitionResults` with the `is_final` flag set to `true`. If `true`, the recognizer will detect a single spoken utterance. When it detects that the user has paused or stopped speaking, it will return an `END_OF_SINGLE_UTTERANCE` event and cease recognition. It will return no more than one `StreamingRecognitionResult` with the `is_final` flag set to `true`.

interim_results

Optional If `true`, interim results (tentative hypotheses) may be returned as they become available (these interim results are indicated with the `is_final=false` flag). If `false` or omitted, only `is_final=true` result(s) are returned.

class google.cloud.speech_v1.types.StreamingRecognitionResult

A streaming speech recognition result corresponding to a portion of the audio that is currently being processed.

alternatives

Output-only May contain one or more recognition hypotheses (up to the maximum specified in `max_alternatives`).

is_final

Output-only If `false`, this `StreamingRecognitionResult` represents an interim result that may change. If `true`, this is the final time the speech service will return this particular `StreamingRecognitionResult`, the recognizer will not return any further hypotheses for this portion of the transcript and corresponding audio.

stability

Output-only An estimate of the likelihood that the recognizer will not change its guess about this interim result. Values range from 0.0 (completely unstable) to 1.0 (completely stable). This field is only provided for interim results (`is_final=false`). The default of 0.0 is a sentinel value indicating stability was not set.

class google.cloud.speech_v1.types.StreamingRecognizeRequest

The top-level message sent by the client for the `StreamingRecognize` method. Multiple `StreamingRecognizeRequest` messages are sent. The first message must contain a `streaming_config` message and must not contain audio data. All subsequent messages must contain audio data and must not contain a `streaming_config` message.

streaming_request

The streaming request, which is either a streaming config or audio content.

streaming_config

Provides information to the recognizer that specifies how to process the request. The first `StreamingRecognizeRequest` message must contain a `streaming_config` message.

audio_content

The audio data to be recognized. Sequential chunks of audio data are sent in sequential `StreamingRecognizeRequest` messages. The first `StreamingRecognizeRequest` message must not contain `audio_content` data and all subsequent `StreamingRecognizeRequest` messages must contain `audio_content` data. The audio bytes must be encoded as specified in `RecognitionConfig`. Note: as with all bytes fields, protobuffers use a pure binary representation (not base64). See [audio limits](#).

class google.cloud.speech_v1.types.StreamingRecognizeResponse

`StreamingRecognizeResponse` is the only message returned to the client by `StreamingRecognize`. A series of one or more `StreamingRecognizeResponse` messages are streamed back to the client.

Here's an example of a series of ten `StreamingRecognizeResponses` that might be returned while processing audio:

1. results { alternatives { transcript: "tube" } stability: 0.01 }
2. results { alternatives { transcript: "to be a" } stability: 0.01 }
3. results { alternatives { transcript: "to be" } stability: 0.9 } results { alternatives { transcript: " or not to be" } stability: 0.01 }
4. results { alternatives { transcript: "to be or not to be" confidence: 0.92 } alternatives { transcript: "to bee or not to bee" } is_final: true }
5. results { alternatives { transcript: " that's" } stability: 0.01 }
6. results { alternatives { transcript: " that is" } stability: 0.9 } results { alternatives { transcript: " the question" } stability: 0.01 }
7. results { alternatives { transcript: " that is the question" confidence: 0.98 } alternatives { transcript: " that was the question" } is_final: true }

Notes:

- Only two of the above responses #4 and #7 contain final results; they are indicated by `is_final: true`. Concatenating these together generates the full transcript: "to be or not to be that is the question".
- The others contain interim results. #3 and #6 contain two interim results: the first portion has a high stability and is less likely to change; the second portion has a low stability and is very likely to change. A UI designer might choose to show only high stability results.

- The specific `stability` and `confidence` values shown above are only for illustrative purposes. Actual values may vary.
- In each response, only one of these fields will be set: `error`, `speech_event_type`, or one or more (repeated) `results`.

error

Output-only If set, returns a `[google.rpc.Status][google.rpc.Status]` message that specifies the error for the operation.

results

Output-only This repeated list contains zero or more results that correspond to consecutive portions of the audio currently being processed. It contains zero or one `is_final=true` result (the newly settled portion), followed by zero or more `is_final=false` results.

speech_event_type

Output-only Indicates the type of speech event.

class `google.cloud.speech_v1.types.WordInfo`

Word-specific information for recognized words. Word information is only included in the response when certain request parameters are set, such as `enable_word_time_offsets`.

start_time

Output-only Time offset relative to the beginning of the audio, and corresponding to the start of the spoken word. This field is only set if `enable_word_time_offsets=true` and only in the top hypothesis. This is an experimental feature and the accuracy of the time offset can vary.

end_time

Output-only Time offset relative to the beginning of the audio, and corresponding to the end of the spoken word. This field is only set if `enable_word_time_offsets=true` and only in the top hypothesis. This is an experimental feature and the accuracy of the time offset can vary.

word

Output-only The word corresponding to this set of information.

15.1 Error Reporting Client

Client for interacting with the Stackdriver Error Reporting API

```
class google.cloud.error_reporting.client.Client (project=None, credentials=None,  
        _http=None, service=None, version=None, _use_grpc=None)
```

Bases: *google.cloud.client.ClientWithProject*

Error Reporting client. Currently Error Reporting is done by creating a Logging client.

Parameters

- **project** (*str*) – the project which the client acts on behalf of. If not passed falls back to the default inferred from the environment.
- **credentials** (*oauth2client.client.OAuth2Credentials* or *NoneType*) – The OAuth2 Credentials to use for the connection owned by this client. If not passed (and if no `_http` object is passed), falls back to the default inferred from the environment.
- **_http** (*Session*) – (Optional) HTTP object to make requests. Can be any object that defines `request()` with the same interface as `requests.Session.request()`. If not passed, an `_http` object is created that is bound to the `credentials` for the current object. This parameter should be considered private, and could change in the future.
- **service** (*str*) – An identifier of the service, such as the name of the executable, job, or Google App Engine service name. This field is expected to have a low number of values that are relatively stable over time, as opposed to version, which can be changed whenever new code is deployed.
- **version** (*str*) – Represents the source code version that the developer provided, which could represent a version label or a Git SHA-1 hash, for example. If the developer did not provide a version, the value is set to default.
- **_use_grpc** (*bool*) – (Optional) Explicitly specifies whether to use the gRPC transport (via GAX) or HTTP. If unset, falls back to the `GOOGLE_CLOUD_DISABLE_GRPC` envi-

ronment variable. This parameter should be considered private, and could change in the future.

Raises `ValueError` if the project is neither passed in nor set in the environment.

SCOPE = ('<https://www.googleapis.com/auth/cloud-platform>',)

The scopes required for authenticating as an API consumer.

report (*message*, *http_context*=None, *user*=None)

Reports a message to Stackdriver Error Reporting

<https://cloud.google.com/error-reporting/docs/formatting-error-messages>

Parameters

- **message** (*str*) – A user-supplied message to report
- **http_context** (*:class`google.cloud.error_reporting.HTTPContext`*) – The HTTP request which was processed when the error was triggered.
- **user** (*str*) – The user who caused or was affected by the crash. This can be a user ID, an email address, or an arbitrary token that uniquely identifies the user. When sending an error report, leave this field empty if the user was not logged in. In this case the Error Reporting system will use other data, such as remote IP address, to distinguish affected users.

Example:

```
>>> client.report("Something went wrong!")
```

report_errors_api

Helper for logging-related API calls.

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/entries> <https://cloud.google.com/logging/docs/reference/v2/rest/v2/projects.logs>

Return type `_gax._ErrorReportingGaxApi` or `_logging._ErrorReportingLoggingAPI`

Returns A class that implements the report errors API.

report_exception (*http_context*=None, *user*=None)

Reports the details of the latest exceptions to Stackdriver Error Reporting.

Parameters

- **http_context** (*:class`google.cloud.error_reporting.HTTPContext`*) – The HTTP request which was processed when the error was triggered.
- **user** (*str*) –

The user who caused or was affected by the crash. This can be a user ID, an email address, or an arbitrary token that uniquely identifies the user. When sending an error report, leave this field empty if the user was not logged in. In this case the Error Reporting system will use other data, such as remote IP address, to distinguish affected users.

Example:

```

>>>     try:
>>>         raise NameError
>>>     except Exception:
>>>         client.report_exception()

```

```

class google.cloud.error_reporting.client.HTTPContext (method=None, url=None,
                                                         user_agent=None,
                                                         referrer=None, re-
                                                         sponse_status_code=None,
                                                         remote_ip=None)

```

Bases: `object`

HTTPContext defines an object that captures the parameter for the `httpRequest` part of Error Reporting API

Parameters

- **method** (*str*) – The type of HTTP request, such as GET, POST, etc.
- **url** (*str*) – The URL of the request
- **user_agent** (*str*) – The user agent information that is provided with the request.
- **referrer** (*str*) – The referrer information that is provided with the request.
- **response_status_code** (*int*) – The HTTP response status code for the request.
- **remote_ip** (*str*) – The IP address from which the request originated. This can be IPv4, IPv6, or a token which is derived from the IP address, depending on the data that has been provided in the error report.

15.2 Error Reporting Utilities

Utility functions for Stackdriver Error Reporting.

`google.cloud.error_reporting.util.build_flask_context(request)`

Builds an HTTP context object from a Flask (Werkzeug) request object.

This helper method extracts the relevant HTTP context from a Flask request object into an object ready to be sent to Error Reporting.

```

>>> @app.errorhandler(HTTPException)
... def handle_error(exc):
...     client.report_exception(
...         http_context=build_flask_context(request))
...     # rest of error response code here

```

Parameters **request** (`werkzeug.wrappers.request`) – The Flask request object to convert.

Return type `HTTPContext`

Returns An HTTPContext object ready to be sent to the Stackdriver Error Reporting API.

15.3 Authentication and Configuration

- For an overview of authentication in `google-cloud-python`, see [Authentication](#).

- In addition to any authentication configuration, you should also set the `GOOGLE_CLOUD_PROJECT` environment variable for the project you'd like to interact with. If you are Google App Engine or Google Compute Engine this will be detected automatically.
- After configuring your environment, create a *Client*

```
>>> from google.cloud import error_reporting
>>> client = error_reporting.Client()
```

or pass in credentials and project explicitly

```
>>> from google.cloud import error_reporting
>>> client = error_reporting.Client(project='my-project', credentials=creds)
```

Error Reporting associates errors with a service, which is an identifier for an executable, App Engine service, or job. The default service is “python”, but a default can be specified for the client on construction time. You can also optionally specify a version for that service, which defaults to “default.”

```
>>> from google.cloud import error_reporting
>>> client = error_reporting.Client(project='my-project',
...                               service="login_service",
...                               version="0.1.0")
```

15.4 Reporting an exception

Report a stacktrace to Stackdriver Error Reporting after an exception

```
>>> from google.cloud import error_reporting
>>> client = error_reporting.Client()
>>> try:
>>>     raise NameError
>>> except Exception:
>>>     client.report_exception()
```

By default, the client will report the error using the service specified in the client's constructor, or the default service of “python”.

The user and HTTP context can also be included in the exception. The HTTP context can be constructed using `google.cloud.error_reporting.HTTPContext`. This will be used by Stackdriver Error Reporting to help group exceptions.

```
>>> from google.cloud import error_reporting
>>> client = error_reporting.Client()
>>> user = 'example@gmail.com'
>>> http_context = HTTPContext(method='GET', url='/', userAgent='test agent',
...                           referrer='example.com', responseStatusCode=500,
...                           remote_ip='1.2.3.4')
>>> try:
>>>     raise NameError
>>> except Exception:
>>>     client.report_exception(http_context=http_context, user=user)
```

An automatic helper to build the HTTP Context from a Flask (Werkzeug) request object is provided.

```
>>> from google.cloud.error_reporting import build_flask_context
>>> @app.errorhandler(HTTPException)
```

```
... def handle_error(exc):  
...     client.report_exception(  
...         http_context=build_flask_context(request))  
...     # rest of error response code here
```

15.5 Reporting an error without an exception

Errors can also be reported to Stackdriver Error Reporting outside the context of an exception. The library will include the file path, function name, and line number of the location where the error was reported.

```
>>> from google.cloud import error_reporting  
>>> client = error_reporting.Client()  
>>> error_reporting.report("Found an error!")
```

Similarly to reporting an exception, the user and HTTP context can be provided:

```
>>> from google.cloud import error_reporting  
>>> client = error_reporting.Client()  
>>> user = 'example@gmail.com'  
>>> http_context = HTTPContext(method='GET', url='/', userAgent='test agent',  
...                           referrer='example.com', responseStatusCode=500,  
...                           remote_ip='1.2.3.4')  
>>> error_reporting.report("Found an error!", http_context=http_context, user=user)
```


16.1 Stackdriver Monitoring Client

Client for interacting with the [Google Stackdriver Monitoring API \(V3\)](#).

Example:

```
>>> from google.cloud import monitoring
>>> client = monitoring.Client()
>>> query = client.query(minutes=5)
>>> print(query.as_dataframe()) # Requires pandas.
```

At present, the client supports querying of time series, metric descriptors, and monitored resource descriptors.

```
class google.cloud.monitoring.client.Client (project=None, credentials=None,
                                             _http=None)
Bases: google.cloud.client.ClientWithProject
```

Client to bundle configuration needed for API requests.

Parameters

- **project** (*str*) – The target project. If not passed, falls back to the default inferred from the environment.
- **credentials** (*Credentials*) – (Optional) The OAuth2 Credentials to use for this client. If not passed (and if no `_http` object is passed), falls back to the default inferred from the environment.
- **_http** (*Session*) – (Optional) HTTP object to make requests. Can be any object that defines `request()` with the same interface as `requests.Session.request()`. If not passed, an `_http` object is created that is bound to the `credentials` for the current object. This parameter should be considered private, and could change in the future.

```
SCOPE = ('https://www.googleapis.com/auth/monitoring.read', 'https://www.googleapis.com
```

The scopes required for authenticating as a Monitoring consumer.

fetch_group (*group_id*)

Fetch a group from the API based on it's ID.

Example:

```
>>> try:
>>>     group = client.fetch_group('1234')
>>> except google.cloud.exceptions.NotFound:
>>>     print('That group does not exist!')
```

Parameters **group_id** (*str*) – The ID of the group.

Return type *Group*

Returns The group instance.

Raises `google.cloud.exceptions.NotFound` if the group is not found.

fetch_metric_descriptor (*metric_type*)

Look up a metric descriptor by type.

Example:

```
>>> METRIC = 'compute.googleapis.com/instance/cpu/utilization'
>>> print(client.fetch_metric_descriptor(METRIC))
```

Parameters **metric_type** (*str*) – The metric type name.

Return type *MetricDescriptor*

Returns The metric descriptor instance.

Raises `google.cloud.exceptions.NotFound` if the metric descriptor is not found.

fetch_resource_descriptor (*resource_type*)

Look up a monitored resource descriptor by type.

Example:

```
>>> print(client.fetch_resource_descriptor('gce_instance'))
```

Parameters **resource_type** (*str*) – The resource type name.

Return type *ResourceDescriptor*

Returns The resource descriptor instance.

Raises `google.cloud.exceptions.NotFound` if the resource descriptor is not found.

group (*group_id=None, display_name=None, parent_id=None, filter_string=None, is_cluster=False*)

Factory constructor for group object.

Note: This will not make an HTTP request; it simply instantiates a group object owned by this client.

Parameters

- **group_id** (*str*) – (Optional) The ID of the group.

- **display_name** (*str*) – (Optional) A user-assigned name for this group, used only for display purposes.
- **parent_id** (*str*) – (Optional) The ID of the group's parent, if it has one.
- **filter_string** (*str*) – (Optional) The filter string used to determine which monitored resources belong to this group.
- **is_cluster** (*bool*) – If true, the members of this group are considered to be a cluster. The system can perform additional analysis on groups that are clusters.

Return type `Group`

Returns The group created with the passed-in arguments.

Raises `ValueError` if both `group_id` and `name` are specified.

list_groups ()

List all groups for the project.

Example:

```
>>> for group in client.list_groups():
...     print((group.display_name, group.name))
```

Return type list of `Group`

Returns A list of group instances.

list_metric_descriptors (*filter_string=None, type_prefix=None*)

List all metric descriptors for the project.

Examples:

```
>>> for descriptor in client.list_metric_descriptors():
...     print(descriptor.type)

>>> for descriptor in client.list_metric_descriptors(
...     type_prefix='custom.'):
...     print(descriptor.type)
```

Parameters

- **filter_string** (*str*) – (Optional) An optional filter expression describing the metric descriptors to be returned. See the [filter documentation](#).
- **type_prefix** (*str*) – (Optional) An optional prefix constraining the selected metric types. This adds `metric.type = starts_with("<prefix>")` to the filter.

Return type list of `MetricDescriptor`

Returns A list of metric descriptor instances.

list_resource_descriptors (*filter_string=None*)

List all monitored resource descriptors for the project.

Example:

```
>>> for descriptor in client.list_resource_descriptors():
...     print(descriptor.type)
```

Parameters `filter_string` (*str*) – (Optional) An optional filter expression describing the resource descriptors to be returned. See the [filter documentation](#).

Return type list of *ResourceDescriptor*

Returns A list of resource descriptor instances.

static metric (*type_*, *labels*)

Factory for constructing metric objects.

Metric objects are typically created to write custom metric values. The type should match the metric type specified in the *MetricDescriptor* used to create the custom metric:

```
>>> metric = client.metric('custom.googleapis.com/my_metric',
...                        labels={
...                            'status': 'successful',
...                        })
```

Parameters

- **type** (*str*) – The metric type name.
- **labels** (*dict*) – A mapping from label names to values for all labels enumerated in the associated *MetricDescriptor*.

Return type *Metric*

Returns The metric object.

metric_descriptor (*type_*, *metric_kind*=*'METRIC_KIND_UNSPECIFIED'*, *value_type*=*'VALUE_TYPE_UNSPECIFIED'*, *labels*=(), *unit*=", *description*=", *display_name*=")

Construct a metric descriptor object.

Metric descriptors specify the schema for a particular metric type.

This factory method is used most often in conjunction with the metric descriptor *create()* method to define custom metrics:

```
>>> descriptor = client.metric_descriptor(
...     'custom.googleapis.com/my_metric',
...     metric_kind=MetricKind.GAUGE,
...     value_type=ValueKind.DOUBLE,
...     description='This is a simple example of a custom metric.')
>>> descriptor.create()
```

Here is an example where the custom metric is parameterized by a metric label:

```
>>> label = LabelDescriptor('response_code', LabelValueType.INT64,
...                        description='HTTP status code')
>>> descriptor = client.metric_descriptor(
...     'custom.googleapis.com/my_app/response_count',
...     metric_kind=MetricKind.CUMULATIVE,
...     value_type=ValueKind.INT64,
...     labels=[label],
...     description='Cumulative count of HTTP responses.')
>>> descriptor.create()
```

Parameters

- **type** (*str*) – The metric type including a DNS name prefix. For example: "custom.googleapis.com/my_metric"
- **metric_kind** (*str*) – The kind of measurement. It must be one of `MetricKind.GAUGE`, `MetricKind.DELTA`, or `MetricKind.CUMULATIVE`. See [MetricKind](#).
- **value_type** (*str*) – The value type of the metric. It must be one of `ValueType.BOOL`, `ValueType.INT64`, `ValueType.DOUBLE`, `ValueType.STRING`, or `ValueType.DISTRIBUTION`. See [ValueType](#).
- **labels** (list of [LabelDescriptor](#)) – A sequence of zero or more label descriptors specifying the labels used to identify a specific instance of this metric.
- **unit** (*str*) – An optional unit in which the metric value is reported.
- **description** (*str*) – An optional detailed description of the metric.
- **display_name** (*str*) – An optional concise name for the metric.

Return type `MetricDescriptor`

Returns The metric descriptor created with the passed-in arguments.

query (*metric_type*='compute.googleapis.com/instance/cpu/utilization', *end_time*=None, *days*=0, *hours*=0, *minutes*=0)

Construct a query object for retrieving metric data.

Example:

```
>>> query = client.query(minutes=5)
>>> print(query.as_dataframe()) # Requires pandas.
```

Parameters

- **metric_type** (*str*) – The metric type name. The default value is `Query.DEFAULT_METRIC_TYPE`, but please note that this default value is provided only for demonstration purposes and is subject to change. See the [supported metrics](#).
- **end_time** (`datetime.datetime`) – (Optional) The end time (inclusive) of the time interval for which results should be returned, as a datetime object. The default is the start of the current minute.

The start time (exclusive) is determined by combining the values of `days`, `hours`, and `minutes`, and subtracting the resulting duration from the end time.

It is also allowed to omit the end time and duration here, in which case [select_interval\(\)](#) must be called before the query is executed.

- **days** (*int*) – The number of days in the time interval.
- **hours** (*int*) – The number of hours in the time interval.
- **minutes** (*int*) – The number of minutes in the time interval.

Return type `Query`

Returns The query object.

Raises `ValueError` if `end_time` is specified but `days`, `hours`, and `minutes` are all zero. If you really want to specify a point in time, use [select_interval\(\)](#).

static resource (*type_, labels*)

Factory for constructing monitored resource objects.

A monitored resource object (*Resource*) is typically used to create a *TimeSeries* object.

For a list of possible monitored resource types and their associated labels, see:

<https://cloud.google.com/monitoring/api/resources>

Parameters

- **type** (*str*) – The monitored resource type name.
- **labels** (*dict*) – A mapping from label names to values for all labels enumerated in the associated *ResourceDescriptor*, except that `project_id` can and should be omitted when writing time series data.

Return type *Resource*

Returns A monitored resource object.

static time_series (*metric, resource, value, end_time=None, start_time=None*)

Construct a time series object for a single data point.

Note: While *TimeSeries* objects returned by the API typically have multiple data points, *TimeSeries* objects sent to the API must have at most one point.

For example:

```
>>> timeseries = client.time_series(metric, resource, 1.23,
...                               end_time=end)
```

For more information, see:

https://cloud.google.com/monitoring/api/ref_v3/rest/v3/TimeSeries

Parameters

- **metric** (*Metric*) – A *Metric*.
- **resource** (*Resource*) – A *Resource* object.
- **value** (*bool, int, string, or float*) – The value of the data point to create for the *TimeSeries*.

Note: The Python type of the value will determine the `ValueType` sent to the API, which must match the value type specified in the metric descriptor. For example, a Python float will be sent to the API as a `ValueType.DOUBLE`.

- **end_time** (*datetime*) – The end time for the point to be included in the time series. Assumed to be UTC if no time zone information is present. Defaults to the current time, as obtained by calling `datetime.datetime.utcnow()`.
- **start_time** (*datetime*) – The start time for the point to be included in the time series. Assumed to be UTC if no time zone information is present. Defaults to `None`. If the start time is unspecified, the API interprets the start time to be the same as the end time.

Return type *TimeSeries*

Returns A time series object.

write_point (*metric, resource, value, end_time=None, start_time=None*)

Write a single point for a metric to the API.

This is a convenience method to write a single time series object to the API. To write multiple time series objects to the API as a batch operation, use the `time_series()` factory method to create time series objects and the `write_time_series()` method to write the objects.

Example:

```
>>> client.write_point(metric, resource, 3.14)
```

Parameters

- **metric** (*Metric*) – A *Metric* object.
- **resource** (*Resource*) – A *Resource* object.
- **value** (*bool, int, string, or float*) – The value of the data point to create for the *TimeSeries*.

Note: The Python type of the value will determine the `ValueType` sent to the API, which must match the value type specified in the metric descriptor. For example, a Python float will be sent to the API as a `ValueType.DOUBLE`.

- **end_time** (*datetime*) – The end time for the point to be included in the time series. Assumed to be UTC if no time zone information is present. Defaults to the current time, as obtained by calling `datetime.datetime.utcnow()`.
- **start_time** (*datetime*) – The start time for the point to be included in the time series. Assumed to be UTC if no time zone information is present. Defaults to `None`. If the start time is unspecified, the API interprets the start time to be the same as the end time.

write_time_series (*timeseries_list*)

Write a list of time series objects to the API.

The recommended approach to creating time series objects is using the `time_series()` factory method.

Example:

```
>>> client.write_time_series([ts1, ts2])
```

If you only need to write a single time series object, consider using the `write_point()` method instead.

Parameters **timeseries_list** (list of *TimeSeries*) – A list of time series object to be written to the API. Each time series must contain exactly one point.

16.2 Metric Descriptors

Metric Descriptors for the [Google Stackdriver Monitoring API \(V3\)](#).

class `google.cloud.monitoring.metric.Metric`

Bases: `google.cloud.monitoring.metric.Metric`

A specific metric identified by specifying values for all labels.

The preferred way to construct a metric object is using the `metric()` factory method of the `Client` class.

Parameters

- **type** (*str*) – The metric type name.
- **labels** (*dict*) – A mapping from label names to values for all labels enumerated in the associated *MetricDescriptor*.

Create new instance of *Metric*(type, labels)

```
class google.cloud.monitoring.metric.MetricDescriptor(client, type_, metric_kind='METRIC_KIND_UNSPECIFIED', value_type='VALUE_TYPE_UNSPECIFIED', labels=(), unit="", description="", display_name="", name=None)
```

Bases: *object*

Specification of a metric type and its schema.

The preferred way to construct a metric descriptor object is using the *metric_descriptor()* factory method of the *Client* class.

Parameters

- **client** (*google.cloud.monitoring.client.Client*) – A client for operating on the metric descriptor.
- **type** (*str*) – The metric type including a DNS name prefix. For example: "compute.googleapis.com/instance/cpu/utilization"
- **metric_kind** (*str*) – The kind of measurement. It must be one of *MetricKind*. *GAUGE*, *MetricKind.DELTA*, or *MetricKind.CUMULATIVE*. See *MetricKind*.
- **value_type** (*str*) – The value type of the metric. It must be one of *ValueType*. *BOOL*, *ValueType.INT64*, *ValueType.DOUBLE*, *ValueType.STRING*, or *ValueType.DISTRIBUTION*. See *ValueType*.
- **labels** (list of *LabelDescriptor*) – A sequence of zero or more label descriptors specifying the labels used to identify a specific instance of this metric.
- **unit** (*str*) – An optional unit in which the metric value is reported.
- **description** (*str*) – An optional detailed description of the metric.
- **display_name** (*str*) – An optional concise name for the metric.
- **name** (*str*) – (Optional) The “resource name” of the metric descriptor. For example: "projects/<project_id>/metricDescriptors/<type>". As retrieved from the service, this will always be specified. You can and should omit it when constructing an instance for the purpose of creating a new metric descriptor.

create()

Create a new metric descriptor based on this object.

Example:

```
>>> descriptor = client.metric_descriptor(
...     'custom.googleapis.com/my_metric',
...     metric_kind=MetricKind.GAUGE,
...     value_type=ValueType.DOUBLE,
...     description='This is a simple example of a custom metric.')
>>> descriptor.create()
```

The metric kind must not be `MetricKind.METRIC_KIND_UNSPECIFIED`, and the value type must not be `ValueType.VALUE_TYPE_UNSPECIFIED`.

The name attribute is ignored in preparing the creation request. All attributes are overwritten by the values received in the response (normally affecting only name).

delete()

Delete the metric descriptor identified by this object.

Example:

```
>>> descriptor = client.metric_descriptor(
...     'custom.googleapis.com/my_metric')
>>> descriptor.delete()
```

Only the `client` and `type` attributes are used.

class `google.cloud.monitoring.metric.MetricKind`

Bases: `object`

Choices for the `kind of measurement`.

`METRIC_KIND_UNSPECIFIED = 'METRIC_KIND_UNSPECIFIED'`

Note: An unspecified kind is not allowed in metric descriptors.

class `google.cloud.monitoring.metric.ValueType`

Bases: `object`

Choices for the `metric value type`.

`VALUE_TYPE_UNSPECIFIED = 'VALUE_TYPE_UNSPECIFIED'`

Note: An unspecified type is not allowed in metric descriptors.

16.3 Monitored Resource Descriptors

Monitored Resource Descriptors for the [Google Stackdriver Monitoring API \(V3\)](#).

class `google.cloud.monitoring.resource.Resource`

Bases: `google.cloud.monitoring.resource.Resource`

A monitored resource identified by specifying values for all labels.

The preferred way to construct a resource object is using the `resource()` factory method of the `Client` class.

Parameters

- **type** (*str*) – The resource type name.
- **labels** (*dict*) – A mapping from label names to values for all labels enumerated in the associated `ResourceDescriptor`.

Create new instance of `Resource(type, labels)`

```
class google.cloud.monitoring.resource.ResourceDescriptor(name, type_, display_name, description, labels)
```

Bases: `object`

Specification of a monitored resource type and its schema.

Parameters

- **name** (*str*) – The “resource name” of the monitored resource descriptor: "projects/<project_id>/monitoredResourceDescriptors/<type>"
- **type** (*str*) – The monitored resource type. For example: "gce_instance"
- **display_name** (*str*) – A concise name that might be displayed in user interfaces.
- **description** (*str*) – A detailed description that might be used in documentation.
- **labels** (list of *LabelDescriptor*) – A sequence of label descriptors specifying the labels used to identify a specific instance of this monitored resource.

16.4 Groups

Groups for the [Google Stackdriver Monitoring API \(V3\)](#).

```
class google.cloud.monitoring.group.Group(client, group_id=None, display_name=None, parent_id=None, filter_string=None, is_cluster=False)
```

Bases: `object`

A dynamic collection of monitored resources.

Parameters

- **client** (*google.cloud.monitoring.client.Client*) – A client for operating on the metric descriptor.
- **group_id** (*str*) – (Optional) The ID of the group.
- **display_name** (*str*) – (Optional) A user-assigned name for this group, used only for display purposes.
- **parent_id** (*str*) – (Optional) The ID of the group’s parent, if it has one.
- **filter_string** (*str*) – (Optional) The filter string used to determine which monitored resources belong to this group.
- **is_cluster** (*bool*) – If true, the members of this group are considered to be a cluster. The system can perform additional analysis on groups that are clusters.

create()

Create a new group based on this object via a POST request.

Example:

```
>>> filter_string = 'resource.type = "gce_instance"'
>>> group = client.group(
...     display_name='My group',
...     filter_string=filter_string,
...     parent_id='5678',
...     is_cluster=True)
>>> group.create()
```


The `name` attribute is ignored in preparing the creation request. All attributes are overwritten by the values received in the response (normally affecting only `name`).

`delete()`

Delete the group via a DELETE request.

Example:

```
>>> group = client.group('1234')
>>> group.delete()
```

Only the `client` and `name` attributes are used.

Warning: This method will fail for groups that have one or more children groups.

`exists()`

Test for the existence of the group via a GET request.

Return type `bool`

Returns Boolean indicating existence of the group.

`fetch_parent()`

Returns the parent group of this group via a GET request.

Return type `Group` or `None`

Returns The parent of the group.

`id`

Returns the group ID.

Return type `str` or `None`

Returns the ID of the group based on it's name.

`list_ancestors()`

Lists all ancestors of this group via a GET request.

The groups are returned in order, starting with the immediate parent and ending with the most distant ancestor. If the specified group has no immediate parent, the results are empty.

Return type list of `Group`

Returns A list of group instances.

`list_children()`

Lists all children of this group via a GET request.

Returns groups whose `parent_name` field contains the group name. If no groups have this parent, the results are empty.

Return type list of `Group`

Returns A list of group instances.

`list_descendants()`

Lists all descendants of this group via a GET request.

This returns a superset of the results returned by the `children()` method, and includes children-of-children, and so forth.

Return type list of `Group`

Returns A list of group instances.

list_members (*filter_string=None, end_time=None, start_time=None*)

Lists all members of this group via a GET request.

If no `end_time` is provided then the group membership over the last minute is returned.

Example:

```
>>> for member in group.list_members():
...     print(member)
```

List members that are Compute Engine VM instances:

```
>>> filter_string = 'resource.type = "gce_instance"'
>>> for member in group.list_members(filter_string=filter_string):
...     print(member)
```

List historical members that existed between 4 and 5 hours ago:

```
>>> import datetime
>>> t1 = datetime.datetime.utcnow() - datetime.timedelta(hours=4)
>>> t0 = t1 - datetime.timedelta(hours=1)
>>> for member in group.list_members(end_time=t1, start_time=t0):
...     print(member)
```

Parameters

- **filter_string** (*str*) – (Optional) An optional list filter describing the members to be returned. The filter may reference the type, labels, and metadata of monitored resources that comprise the group. See the [filter documentation](#).
- **end_time** (*datetime.datetime*) – (Optional) The end time (inclusive) of the time interval for which results should be returned, as a datetime object. If `start_time` is specified, then this must also be specified.
- **start_time** (*datetime.datetime*) – (Optional) The start time (exclusive) of the time interval for which results should be returned, as a datetime object.

Return type list of [Resource](#)

Returns A list of resource instances.

Raises [ValueError](#) if the `start_time` is specified, but the `end_time` is missing.

name

Returns the fully qualified name of the group.

Return type [str](#) or [None](#)

Returns The fully qualified name of the group in the format “projects/<project>/groups/<id>”.

parent_name

Returns the fully qualified name of the parent group.

Return type [str](#) or [None](#)

Returns The fully qualified name of the parent group.

path

URL path to this group.

Return type [str](#)

Returns the path based on project and group name.

Raises `ValueError` if `name` is not specified.

reload()

Sync local group information via a GET request.

Warning: This will overwrite any local changes you've made and not saved via `update()`.

update()

Update the group via a PUT request.

16.5 Time Series Query

Time series query for the [Google Stackdriver Monitoring API \(V3\)](#).

class `google.cloud.monitoring.query.Aligner`

Bases: `object`

Allowed values for the [supported aligners](#).

class `google.cloud.monitoring.query.Query` (*client*, *metric_type*='compute.googleapis.com/instance/cpu/utilization', *end_time*=None, *days*=0, *hours*=0, *minutes*=0)

Bases: `object`

Query object for retrieving metric data.

The preferred way to construct a query object is using the `query()` method of the `Client` class.

Parameters

- **client** (`google.cloud.monitoring.client.Client`) – The client to use.
- **metric_type** (`str`) – The metric type name. The default value is `Query.DEFAULT_METRIC_TYPE`, but please note that this default value is provided only for demonstration purposes and is subject to change. See the [supported metrics](#).
- **end_time** (`datetime.datetime`) – (Optional) The end time (inclusive) of the time interval for which results should be returned, as a datetime object. The default is the start of the current minute.

The start time (exclusive) is determined by combining the values of `days`, `hours`, and `minutes`, and subtracting the resulting duration from the end time.

It is also allowed to omit the end time and duration here, in which case `select_interval()` must be called before the query is executed.
- **days** (`int`) – The number of days in the time interval.
- **hours** (`int`) – The number of hours in the time interval.
- **minutes** (`int`) – The number of minutes in the time interval.

Raises `ValueError` if `end_time` is specified but `days`, `hours`, and `minutes` are all zero. If you really want to specify a point in time, use `select_interval()`.

align (*per_series_aligner*, *seconds*=0, *minutes*=0, *hours*=0)

Copy the query and add temporal alignment.

If `per_series_aligner` is not `Aligner.ALIGN_NONE`, each time series will contain data points only on the period boundaries.

Example:

```
query = query.align(Aligner.ALIGN_MEAN, minutes=5)
```

It is also possible to specify the aligner as a literal string:

```
query = query.align('ALIGN_MEAN', minutes=5)
```

Parameters

- **per_series_aligner** (*str*) – The approach to be used to align individual time series. For example: `Aligner.ALIGN_MEAN`. See *Aligner* and the descriptions of the supported aligners.
- **seconds** (*int*) – The number of seconds in the alignment period.
- **minutes** (*int*) – The number of minutes in the alignment period.
- **hours** (*int*) – The number of hours in the alignment period.

Return type *Query*

Returns The new query object.

as_dataframe (*label=None, labels=None*)

Return all the selected time series as a `pandas` dataframe.

Note: Use of this method requires that you have `pandas` installed.

Examples:

```
# Generate a dataframe with a multi-level column header including
# the resource type and all available resource and metric labels.
# This can be useful for seeing what labels are available.
dataframe = query.as_dataframe()

# Generate a dataframe using a particular label for the column
# names.
dataframe = query.as_dataframe(label='instance_name')

# Generate a dataframe with a multi-level column header.
dataframe = query.as_dataframe(labels=['zone', 'instance_name'])

# Generate a dataframe with a multi-level column header, assuming
# the metric is issued by more than one type of resource.
dataframe = query.as_dataframe(
    labels=['resource_type', 'instance_id'])
```

Parameters

- **label** (*str*) – (Optional) The label name to use for the dataframe header. This can be the name of a resource label or metric label (e.g., "instance_name"), or the string "resource_type".
- **labels** (*list of strings, or None*) – A list or tuple of label names to use for the dataframe header. If more than one label name is provided, the resulting dataframe will have a multi-level column header. Providing values for both `label` and `labels` is an error.

Return type `pandas.DataFrame`

Returns A dataframe where each column represents one time series.

copy()

Copy the query object.

Return type `Query`

Returns The new query object.

filter

The filter string.

This is constructed from the metric type, the resource type, and selectors for the group ID, monitored projects, resource labels, and metric labels.

iter (*headers_only=False, page_size=None*)

Yield all time series objects selected by the query.

The generator returned iterates over `TimeSeries` objects containing points ordered from oldest to newest.

Note that the `Query` object itself is an iterable, such that the following are equivalent:

```
for timeseries in query:
    ...

for timeseries in query.iter():
    ...
```

Parameters

- **headers_only** (*bool*) – Whether to omit the point data from the time series objects.
- **page_size** (*int*) – (Optional) Positive number specifying the maximum number of points to return per page. This can be used to control how far the iterator reads ahead.

Raises `ValueError` if the query time interval has not been specified.

metric_type

The metric type name.

reduce (*cross_series_reducer, *group_by_fields*)

Copy the query and add cross-series reduction.

Cross-series reduction combines time series by aggregating their data points.

For example, you could request an aggregated time series for each combination of project and zone as follows:

```
query = query.reduce(Reducer.REDUCE_MEAN,
                    'resource.project_id', 'resource.zone')
```

Parameters

- **cross_series_reducer** (*str*) – The approach to be used to combine time series. For example: `Reducer.REDUCE_MEAN`. See [Reducer](#) and the descriptions of the [supported reducers](#).

- **group_by_fields** (*strs*) – Fields to be preserved by the reduction. For example, specifying just "resource.zone" will result in one time series per zone. The default is to aggregate all of the time series into just one.

Return type *Query*

Returns The new query object.

select_group (*group_id*)

Copy the query and add filtering by group.

Example:

```
query = query.select_group('1234567')
```

Parameters **group_id** (*str*) – The ID of a group to filter by.

Return type *Query*

Returns The new query object.

select_interval (*end_time*, *start_time=None*)

Copy the query and set the query time interval.

Example:

```
import datetime

now = datetime.datetime.utcnow()
query = query.select_interval(
    end_time=now,
    start_time=now - datetime.timedelta(minutes=5))
```

As a convenience, you can alternatively specify the end time and an interval duration when you create the query initially.

Parameters

- **end_time** (*datetime.datetime*) – The end time (inclusive) of the time interval for which results should be returned, as a datetime object.
- **start_time** (*datetime.datetime*) – (Optional) The start time (exclusive) of the time interval for which results should be returned, as a datetime object. If not specified, the interval is a point in time.

Return type *Query*

Returns The new query object.

select_metrics (**args*, ***kwargs*)

Copy the query and add filtering by metric labels.

Examples:

```
query = query.select_metrics(instance_name='myinstance')
query = query.select_metrics(instance_name_prefix='mycluster-')
```

A keyword argument <label>=<value> ordinarily generates a filter expression of the form:

```
metric.label.<label> = "<value>"
```

However, by adding `"_prefix"` or `"_suffix"` to the keyword, you can specify a partial match.

`<label>_prefix=<value>` generates:

```
metric.label.<label> = starts_with("<value>")
```

`<label>_suffix=<value>` generates:

```
metric.label.<label> = ends_with("<value>")
```

If the label's value type is INT64, a similar notation can be used to express inequalities:

`<label>_less=<value>` generates:

```
metric.label.<label> < <value>
```

`<label>_lessequal=<value>` generates:

```
metric.label.<label> <= <value>
```

`<label>_greater=<value>` generates:

```
metric.label.<label> > <value>
```

`<label>_greaterequal=<value>` generates:

```
metric.label.<label> >= <value>
```

Parameters

- **args** (*tuple*) – Raw filter expression strings to include in the conjunction. If just one is provided and no keyword arguments are provided, it can be a disjunction.
- **kwargs** (*dict*) – Label filters to include in the conjunction as described above.

Return type *Query*

Returns The new query object.

select_projects (**args*)

Copy the query and add filtering by monitored projects.

This is only useful if the target project represents a Stackdriver account containing the specified monitored projects.

Examples:

```
query = query.select_projects('project-1')
query = query.select_projects('project-1', 'project-2')
```

Parameters **args** (*tuple*) – Project IDs limiting the resources to be included in the query.

Return type *Query*

Returns The new query object.

select_resources (**args, **kwargs*)

Copy the query and add filtering by resource labels.

Examples:

```
query = query.select_resources(zone='us-central1-a')
query = query.select_resources(zone_prefix='europe-')
query = query.select_resources(resource_type='gce_instance')
```

A keyword argument `<label>=<value>` ordinarily generates a filter expression of the form:

```
resource.label.<label> = "<value>"
```

However, by adding `"_prefix"` or `"_suffix"` to the keyword, you can specify a partial match.

`<label>_prefix=<value>` generates:

```
resource.label.<label> = starts_with("<value>")
```

`<label>_suffix=<value>` generates:

```
resource.label.<label> = ends_with("<value>")
```

As a special case, `"resource_type"` is treated as a special pseudo-label corresponding to the filter object `resource.type`. For example, `resource_type=<value>` generates:

```
resource.type = "<value>"
```

See the [defined resource types](#).

Note: The label `"instance_name"` is a metric label, not a resource label. You would filter on it using `select_metrics(instance_name=...)`.

Parameters

- **args** (*tuple*) – Raw filter expression strings to include in the conjunction. If just one is provided and no keyword arguments are provided, it can be a disjunction.
- **kwargs** (*dict*) – Label filters to include in the conjunction as described above.

Return type *Query*

Returns The new query object.

class `google.cloud.monitoring.query.Reducer`

Bases: `object`

Allowed values for the [supported reducers](#).

16.6 Time Series

Time series for the [Google Stackdriver Monitoring API \(V3\)](#).

Features intentionally omitted from this first version of the client library:

- Writing time series.
- Natural representation of distribution values.

class google.cloud.monitoring.timeseries.**Point**

Bases: *google.cloud.monitoring.timeseries.Point*

A single point in a time series.

Parameters

- **end_time** (*str*) – The end time in RFC3339 UTC “Zulu” format.
- **start_time** (*str*) – (Optional) The start time in RFC3339 UTC “Zulu” format.
- **value** (*object*) – The metric value. This can be a scalar or a distribution.

Create new instance of Point(end_time, start_time, value)

class google.cloud.monitoring.timeseries.**TimeSeries**

Bases: *google.cloud.monitoring.timeseries.TimeSeries*

A single time series of metric values.

The preferred way to construct a *TimeSeries* object is using the *time_series()* factory method of the *Client* class.

Parameters

- **metric** (*Metric*) – A metric object.
- **resource** (*Resource*) – A resource object.
- **metric_kind** (*str*) – The kind of measurement: *MetricKind.GAUGE*, *MetricKind.DELTA*, or *MetricKind.CUMULATIVE*. See *MetricKind*.
- **value_type** (*str*) – The value type of the metric: *ValueType.BOOL*, *ValueType.INT64*, *ValueType.DOUBLE*, *ValueType.STRING*, or *ValueType.DISTRIBUTION*. See *ValueType*.
- **points** (list of *Point*) – A list of point objects.

Create new instance of TimeSeries(metric, resource, metric_kind, value_type, points)

header (*points=None*)

Copy everything but the point data.

Parameters **points** (list of *Point*, or None) – An optional point list.

Return type *TimeSeries*

Returns The new time series object.

labels

A single dictionary with values for all the labels.

This combines *resource.labels* and *metric.labels* and also adds "resource_type".

16.7 Label Descriptors

Label Descriptors for the *Stackdriver Monitoring API (V3)*.

class google.cloud.monitoring.label.**LabelDescriptor** (*key*, *value_type='STRING'*, *description=""*)

Bases: *object*

Schema specification and documentation for a single label.

Parameters

- **key** (*str*) – The name of the label.
- **value_type** (*str*) – The type of the label. It must be one of `LabelValueType.STRING`, `LabelValueType.BOOL`, or `LabelValueType.INT64`. See [LabelValueType](#).
- **description** (*str*) – A human-readable description for the label.

class google.cloud.monitoring.label.**LabelValueType**

Bases: `object`

Allowed values for the `type` of a label.

16.8 Introduction

With the Stackdriver Monitoring API, you can work with Stackdriver metric data pertaining to monitored resources in Google Cloud Platform (GCP) or elsewhere.

Essential concepts:

- Metric data is associated with a **monitored resource**. A monitored resource has a *resource type* and a set of *resource labels* — key-value pairs — that identify the particular resource.
- A **metric** further identifies the particular kind of data that is being collected. It has a *metric type* and a set of *metric labels* that, when combined with the resource labels, identify a particular time series.
- A **time series** is a collection of data points associated with points or intervals in time.

Please refer to the documentation for the [Stackdriver Monitoring API](#) for more information.

At present, this client library supports the following features of the API:

- Querying of time series.
- Querying of metric descriptors and monitored resource descriptors.
- Creation and deletion of metric descriptors for custom metrics.
- Writing of custom metric data.

16.9 The Stackdriver Monitoring Client Object

The Stackdriver Monitoring client library generally makes its functionality available as methods of the monitoring *Client* class. A *Client* instance holds authentication credentials and the ID of the target project with which the metric data of interest is associated. This project ID will often refer to a [Stackdriver account](#) binding multiple GCP projects and AWS accounts. It can also simply be the ID of a monitored project.

Most often the authentication credentials will be determined implicitly from your environment. See [Authentication](#) for more information.

It is thus typical to create a client object as follows:

```
>>> from google.cloud import monitoring
>>> client = monitoring.Client(project='target-project')
```

If you are running in Google Compute Engine or Google App Engine, the current project is the default target project. This default can be further overridden with the `GOOGLE_CLOUD_PROJECT` environment variable. Using the default target project is even easier:

```
>>> client = monitoring.Client()
```

If necessary, you can pass in credentials and project explicitly:

```
>>> client = monitoring.Client(project='target-project', credentials=...)
```

16.10 Monitored Resource Descriptors

The available monitored resource types are defined by *monitored resource descriptors*. You can fetch a list of these with the `list_resource_descriptors()` method:

```
>>> for descriptor in client.list_resource_descriptors():
...     print(descriptor.type)
```

Each *ResourceDescriptor* has a type, a display name, a description, and a list of *LabelDescriptor* instances. See the documentation about [Monitored Resources](#) for more information.

16.11 Metric Descriptors

The available metric types are defined by *metric descriptors*. They include [platform metrics](#), [agent metrics](#), and [custom metrics](#). You can list all of these with the `list_metric_descriptors()` method:

```
>>> for descriptor in client.list_metric_descriptors():
...     print(descriptor.type)
```

See *MetricDescriptor* and the [Metric Descriptors API](#) documentation for more information.

You can create new metric descriptors to define custom metrics in the `custom.googleapis.com` namespace. You do this by creating a *MetricDescriptor* object using the client's `metric_descriptor()` factory and then calling the object's `create()` method:

```
>>> from google.cloud.monitoring import MetricKind, ValueType
>>> descriptor = client.metric_descriptor(
...     'custom.googleapis.com/my_metric',
...     metric_kind=MetricKind.GAUGE,
...     value_type=ValueTypes.DOUBLE,
...     description='This is a simple example of a custom metric.')
>>> descriptor.create()
```

You can delete such a metric descriptor as follows:

```
>>> descriptor = client.metric_descriptor(
...     'custom.googleapis.com/my_metric')
>>> descriptor.delete()
```

To define a custom metric parameterized by one or more labels, you must build the appropriate *LabelDescriptor* objects and include them in the *MetricDescriptor* object before you call `create()`:

```
>>> from google.cloud.monitoring import LabelDescriptor, LabelValueType
>>> label = LabelDescriptor('response_code', LabelValueType.INT64,
...     description='HTTP status code')
>>> descriptor = client.metric_descriptor(
...     'custom.googleapis.com/my_app/response_count',
```

```
...     metric_kind=MetricKind.CUMULATIVE,
...     value_type=ValueTypes.INT64,
...     labels=[label],
...     description='Cumulative count of HTTP responses.')
>>> descriptor.create()
```

16.12 Groups

A group is a dynamic collection of *monitored resources* whose membership is defined by a [filter](#). These groups are usually created via the [Stackdriver dashboard](#). You can list all the groups in a project with the `list_groups()` method:

```
>>> for group in client.list_groups():
...     print(group.id, group.display_name, group.parent_id)
('a001', 'Production', None)
('a002', 'Front-end', 'a001')
('1003', 'Back-end', 'a001')
```

See [Group](#) and the API documentation for [Groups](#) and [Group members](#) for more information.

You can get a specific group based on its ID as follows:

```
>>> group = client.fetch_group('a001')
```

You can get the current members of this group using the `list_members()` method:

```
>>> for member in group.list_members():
...     print(member)
```

Passing in `end_time` and `start_time` to the above method will return historical members based on the current filter of the group. The group membership changes over time, as *monitored resources* come and go, and as they change properties.

You can create new groups to define new collections of *monitored resources*. You do this by creating a [Group](#) object using the client's `group()` factory and then calling the object's `create()` method:

```
>>> filter_string = 'resource.zone = "us-central1-a"'
>>> group = client.group(
...     display_name='My group',
...     filter_string=filter_string,
...     parent_id='a001',
...     is_cluster=True)
>>> group.create()
>>> group.id
'1234'
```

You can further manipulate an existing group by first initializing a [Group](#) object with its ID or name, and then calling various methods on it.

Delete a group:

```
>>> group = client.group('1234')
>>> group.exists()
True
>>> group.delete()
```

Update a group:

```
>>> group = client.group('1234')
>>> group.exists()
True
>>> group.reload()
>>> group.display_name = 'New Display Name'
>>> group.update()
```

16.13 Time Series Queries

A time series includes a collection of data points and a set of resource and metric label values. See [TimeSeries](#) and the [Time Series](#) API documentation for more information.

While you can obtain time series objects by iterating over a [Query](#) object, usually it is more useful to retrieve time series data in the form of a `pandas.DataFrame`, where each column corresponds to a single time series. For this, you must have `pandas` installed; it is not a required dependency of `google-cloud-python`.

You can display CPU utilization across your GCE instances over a five minute duration ending at the start of the current minute as follows:

```
>>> METRIC = 'compute.googleapis.com/instance/cpu/utilization'
>>> query = client.query(METRIC, minutes=5)
>>> print(query.as_dataframe())
```

[Query](#) objects provide a variety of methods for refining the query. You can request temporal alignment and cross-series reduction, and you can filter by label values. See the client [query\(\)](#) method and the [Query](#) class for more information.

For example, you can display CPU utilization during the last hour across GCE instances with names beginning with "mycluster-", averaged over five-minute intervals and aggregated per zone, as follows:

```
>>> from google.cloud.monitoring import Aligner, Reducer
>>> METRIC = 'compute.googleapis.com/instance/cpu/utilization'
>>> query = (client.query(METRIC, hours=1)
...         .select_metrics(instance_name_prefix='mycluster-')
...         .align(Aligner.ALIGN_MEAN, minutes=5)
...         .reduce(Reducer.REDUCE_MEAN, 'resource.zone'))
>>> print(query.as_dataframe())
```

16.14 Writing Custom Metrics

The Stackdriver Monitoring API can be used to write data points to custom metrics. Please refer to the documentation on [Custom Metrics](#) for more information.

To write a data point to a custom metric, you must provide an instance of [Metric](#) specifying the metric type as well as the values for the metric labels. You will need to have either created the metric descriptor earlier (see the [Metric Descriptors](#) section) or rely on metric type auto-creation (see [Auto-creation of custom metrics](#)).

You will also need to provide a [Resource](#) instance specifying a monitored resource type as well as values for all of the monitored resource labels, except for `project_id`, which is ignored when it's included in writes to the API. A good choice is to use the underlying physical resource where your application code runs – e.g., a monitored resource type of `gce_instance` or `aws_ec2_instance`. In some limited circumstances, such as when only a single process writes to the custom metric, you may choose to use the `global` monitored resource type.

See [Monitored resource types](#) for more information about particular monitored resource types.

```
>>> from google.cloud import monitoring
>>> # Create a Resource object for the desired monitored resource type.
>>> resource = client.resource(
...     'gce_instance',
...     labels={
...         'instance_id': '1234567890123456789',
...         'zone': 'us-central1-f'
...     }
... )
>>> # Create a Metric object, specifying the metric type as well as values for any_
↪metric labels.
>>> metric = client.metric(
...     type_='custom.googleapis.com/my_metric',
...     labels={
...         'status': 'successful'
...     }
... )
```

With a `Metric` and `Resource` in hand, the `Client` can be used to write `Point` values.

When writing points, the Python type of the value must match the *value type* of the metric descriptor associated with the metric. For example, a Python float will map to `ValueType.DOUBLE`.

Stackdriver Monitoring supports several *metric kinds*: `GAUGE`, `CUMULATIVE`, and `DELTA`. However, `DELTA` is not supported for custom metrics.

`GAUGE` metrics represent only a single point in time, so only the `end_time` should be specified:

```
>>> client.write_point(metric=metric, resource=resource,
...                     value=3.14, end_time=end_time) # API call
```

By default, `end_time` defaults to `utcnow()`, so metrics can be written to the current time as follows:

```
>>> client.write_point(metric, resource, 3.14) # API call
```

`CUMULATIVE` metrics enable the monitoring system to compute rates of increase on metrics that sometimes reset, such as after a process restart. Without cumulative metrics, this reset would otherwise show up as a huge negative spike. For cumulative metrics, the same start time should be re-used repeatedly as more points are written to the time series.

In the examples below, the `end_time` again defaults to the current time:

```
>>> RESET = datetime.utcnow()
>>> client.write_point(metric, resource, 3, start_time=RESET) # API call
>>> client.write_point(metric, resource, 6, start_time=RESET) # API call
```

To write multiple `TimeSeries` in a single batch, you can use `write_time_series()`:

```
>>> ts1 = client.time_series(metric1, resource, 3.14, end_time=end_time)
>>> ts2 = client.time_series(metric2, resource, 42, end_time=end_time)
>>> client.write_time_series([ts1, ts2]) # API call
```

While multiple time series can be written in a single batch, each `TimeSeries` object sent to the API must only include a single point.

All timezone-naive Python `datetime` objects are assumed to be UTC.

17.1 Stackdriver Logging Client

Client for interacting with the Google Stackdriver Logging API.

class `google.cloud.logging.client.Client` (*project=None, credentials=None, _http=None, _use_grpc=None*)

Bases: `google.cloud.client.ClientWithProject`

Client to bundle configuration needed for API requests.

Parameters

- **project** (*str*) – the project which the client acts on behalf of. If not passed, falls back to the default inferred from the environment.
- **credentials** (*Credentials*) – (Optional) The OAuth2 Credentials to use for this client. If not passed (and if no `_http` object is passed), falls back to the default inferred from the environment.
- **_http** (*Session*) – (Optional) HTTP object to make requests. Can be any object that defines `request()` with the same interface as `requests.Session.request()`. If not passed, an `_http` object is created that is bound to the `credentials` for the current object. This parameter should be considered private, and could change in the future.
- **_use_grpc** (*bool*) – (Optional) Explicitly specifies whether to use the gRPC transport (via GAX) or HTTP. If unset, falls back to the `GOOGLE_CLOUD_DISABLE_GRPC` environment variable. This parameter should be considered private, and could change in the future.

SCOPE = ('https://www.googleapis.com/auth/logging.read', 'https://www.googleapis.com/auth/logging.admin')
The scopes required for authenticating as a Logging consumer.

get_default_handler()

Return the default logging handler based on the local environment.

Return type `logging.Handler`

Returns The default log handler based on the environment

list_entries (*projects=None, filter_=None, order_by=None, page_size=None, page_token=None*)
Return a page of log entries.

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/entries/list>

Parameters

- **projects** (*list of strings*) – project IDs to include. If not passed, defaults to the project bound to the client.
- **filter** (*str*) – a filter expression. See https://cloud.google.com/logging/docs/view/advanced_filters
- **order_by** (*str*) – One of ASCENDING or DESCENDING.
- **page_size** (*int*) – maximum number of entries to return, If not passed, defaults to a value set by the API.
- **page_token** (*str*) – opaque marker for the next “page” of entries. If not passed, the API will return the first page of entries.

Return type Iterator

Returns Iterator of `_BaseEntry` accessible to the current client.

list_metrics (*page_size=None, page_token=None*)
List metrics for the project associated with this client.

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/projects.metrics/list>

Parameters

- **page_size** (*int*) – maximum number of metrics to return, If not passed, defaults to a value set by the API.
- **page_token** (*str*) – opaque marker for the next “page” of metrics. If not passed, the API will return the first page of metrics.

Return type Iterator

Returns Iterator of `Metric` accessible to the current client.

list_sinks (*page_size=None, page_token=None*)
List sinks for the project associated with this client.

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/projects.sinks/list>

Parameters

- **page_size** (*int*) – maximum number of sinks to return, If not passed, defaults to a value set by the API.
- **page_token** (*str*) – opaque marker for the next “page” of sinks. If not passed, the API will return the first page of sinks.

Return type Iterator

Returns Iterator of `Sink` accessible to the current client.

logger (*name*)
Creates a logger bound to the current client.

Parameters **name** (*str*) – the name of the logger to be constructed.

Return type `google.cloud.logging.logger.Logger`

Returns Logger created with the current client.

logging_api

Helper for logging-related API calls.

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/entries> <https://cloud.google.com/logging/docs/reference/v2/rest/v2/projects.logs>

metric (*name*, *filter_=None*, *description=""*)

Creates a metric bound to the current client.

Parameters

- **name** (*str*) – the name of the metric to be constructed.
- **filter** (*str*) – the advanced logs filter expression defining the entries tracked by the metric. If not passed, the instance should already exist, to be refreshed via `Metric.reload()`.
- **description** (*str*) – the description of the metric to be constructed. If not passed, the instance should already exist, to be refreshed via `Metric.reload()`.

Return type `google.cloud.logging.metric.Metric`

Returns Metric created with the current client.

metrics_api

Helper for log metric-related API calls.

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/projects.metrics>

setup_logging (*log_level=20*, *excluded_loggers=('google.cloud', 'google.auth', 'google_auth_httplib2')*)

Attach default Stackdriver logging handler to the root logger.

This method uses the default log handler, obtained by `get_default_handler()`, and attaches it to the root Python logger, so that a call such as `logging.warn`, as well as all child loggers, will report to Stackdriver logging.

Parameters

- **log_level** (*int*) – (Optional) Python logging log level. Defaults to `logging.INFO`.
- **excluded_loggers** (*tuple*) – (Optional) The loggers to not attach the handler to. This will always include the loggers in the path of the logging client itself.

sink (*name*, *filter_=None*, *destination=None*)

Creates a sink bound to the current client.

Parameters

- **name** (*str*) – the name of the sink to be constructed.
- **filter** (*str*) – (optional) the advanced logs filter expression defining the entries exported by the sink. If not passed, the instance should already exist, to be refreshed via `Sink.reload()`.
- **destination** (*str*) – destination URI for the entries exported by the sink. If not passed, the instance should already exist, to be refreshed via `Sink.reload()`.

Return type `google.cloud.logging.sink.Sink`

Returns Sink created with the current client.

sinks_api

Helper for log sink-related API calls.

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/projects.sinks>

17.2 Logger

Define API Loggers.

class `google.cloud.logging.logger.Batch` (*logger, client, resource=None*)

Bases: `object`

Context manager: collect entries to log via a single API call.

Helper returned by `Logger.batch()`

Parameters

- **logger** (`google.cloud.logging.logger.Logger`) – the logger to which entries will be logged.
- **client** (`google.cloud.logging.client.Client`) – The client to use.
- **resource** (`Resource`) – (Optional) Monitored resource of the batch, defaults to `None`, which requires that every entry should have a resource specified. Since the methods used to write entries default the entry's resource to the global resource type, this parameter is only required if explicitly set to `None`. If no entries' resource are set to `None`, this parameter will be ignored on the server.

commit (*client=None*)

Send saved log entries as a single API call.

Parameters **client** (`Client` or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current batch.

log_proto (*message, labels=None, insert_id=None, severity=None, http_request=None, timestamp=None, resource=Resource(type='global', labels={})*)

Add a protobuf entry to be logged during `commit()`.

Parameters

- **message** (*protobuf message*) – the protobuf entry
- **labels** (*dict*) – (optional) mapping of labels for the entry.
- **insert_id** (*str*) – (optional) unique ID for log entry.
- **severity** (*str*) – (optional) severity of event being logged.
- **http_request** (*dict*) – (optional) info about HTTP request associated with the entry.
- **timestamp** (`datetime.datetime`) – (optional) timestamp of event being logged.
- **resource** (`Resource`) – (Optional) Monitored resource of the entry. Defaults to the global resource type. If set to `None`, the resource of the batch is used for this entry. If both this resource and the Batch resource are `None`, the API will return an error.

log_struct (*info, labels=None, insert_id=None, severity=None, http_request=None, timestamp=None, resource=Resource(type='global', labels={})*)

Add a struct entry to be logged during `commit()`.

Parameters

- **info** (*dict*) – the struct entry

- **labels** (*dict*) – (optional) mapping of labels for the entry.
- **insert_id** (*str*) – (optional) unique ID for log entry.
- **severity** (*str*) – (optional) severity of event being logged.
- **http_request** (*dict*) – (optional) info about HTTP request associated with the entry.
- **timestamp** (*datetime.datetime*) – (optional) timestamp of event being logged.
- **resource** (*Resource*) – (Optional) Monitored resource of the entry. Defaults to the global resource type. If set to None, the resource of the batch is used for this entry. If both this resource and the Batch resource are None, the API will return an error.

log_text (*text*, *labels=None*, *insert_id=None*, *severity=None*, *http_request=None*, *timestamp=None*, *resource=Resource(type='global', labels={})*)

Add a text entry to be logged during `commit()`.

Parameters

- **text** (*str*) – the text entry
- **labels** (*dict*) – (optional) mapping of labels for the entry.
- **insert_id** (*str*) – (optional) unique ID for log entry.
- **severity** (*str*) – (optional) severity of event being logged.
- **http_request** (*dict*) – (optional) info about HTTP request associated with the entry.
- **timestamp** (*datetime.datetime*) – (optional) timestamp of event being logged.
- **resource** (*Resource*) – (Optional) Monitored resource of the entry. Defaults to the global resource type. If set to None, the resource of the batch is used for this entry. If both this resource and the Batch resource are None, the API will return an error.

class google.cloud.logging.logger.**Logger** (*name*, *client*, *labels=None*)

Bases: `object`

Loggers represent named targets for log entries.

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/projects.logs>

Parameters

- **name** (*str*) – the name of the logger
- **client** (*google.cloud.logging.client.Client*) – A client which holds credentials and project configuration for the logger (which requires a project).
- **labels** (*dict*) – (optional) mapping of default labels for entries written via this logger.

batch (*client=None*)

Return a batch to use as a context manager.

Parameters **client** (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current topic.

Return type *Batch*

Returns A batch to use as a context manager.

client

Client bound to the logger.

delete (*client=None*)

API call: delete all entries in a logger via a DELETE request

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/projects.logs/delete>

Parameters **client** (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current logger.

full_name

Fully-qualified name used in logging APIs

list_entries (*projects=None, filter_=None, order_by=None, page_size=None, page_token=None*)

Return a page of log entries.

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/entries/list>

Parameters

- **projects** (*list of strings*) – project IDs to include. If not passed, defaults to the project bound to the client.
- **filter** (*str*) – a filter expression. See https://cloud.google.com/logging/docs/view/advanced_filters
- **order_by** (*str*) – One of ASCENDING or DESCENDING.
- **page_size** (*int*) – maximum number of entries to return, If not passed, defaults to a value set by the API.
- **page_token** (*str*) – opaque marker for the next “page” of entries. If not passed, the API will return the first page of entries.

Return type *Iterator*

Returns *Iterator* of `_BaseEntry` accessible to the current logger.

log_proto (*message, client=None, labels=None, insert_id=None, severity=None, http_request=None, timestamp=None, resource=Resource(type='global', labels={})*)

API call: log a protobuf message via a POST request

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/entries/list>

Parameters

- **message** (*Message*) – The protobuf message to be logged.
- **client** (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current logger.
- **labels** (*dict*) – (optional) mapping of labels for the entry.
- **insert_id** (*str*) – (optional) unique ID for log entry.
- **severity** (*str*) – (optional) severity of event being logged.
- **http_request** (*dict*) – (optional) info about HTTP request associated with the entry.
- **resource** (*Resource*) – Monitored resource of the entry, defaults to the global resource type.
- **timestamp** (*datetime.datetime*) – (optional) timestamp of event being logged.

log_struct (*info, client=None, labels=None, insert_id=None, severity=None, http_request=None, timestamp=None, resource=Resource(type='global', labels={})*)

API call: log a structured message via a POST request

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/entries/write>

Parameters

- **info** (*dict*) – the log entry information
- **client** (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current logger.
- **labels** (*dict*) – (optional) mapping of labels for the entry.
- **insert_id** (*str*) – (optional) unique ID for log entry.
- **severity** (*str*) – (optional) severity of event being logged.
- **http_request** (*dict*) – (optional) info about HTTP request associated with the entry.
- **resource** (*Resource*) – Monitored resource of the entry, defaults to the global resource type.
- **timestamp** (*datetime.datetime*) – (optional) timestamp of event being logged.

log_text (*text*, *client=None*, *labels=None*, *insert_id=None*, *severity=None*, *http_request=None*, *timestamp=None*, *resource=Resource(type='global', labels={})*)

API call: log a text message via a POST request

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/entries/write>

Parameters

- **text** (*str*) – the log message.
- **client** (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current logger.
- **labels** (*dict*) – (optional) mapping of labels for the entry.
- **insert_id** (*str*) – (optional) unique ID for log entry.
- **severity** (*str*) – (optional) severity of event being logged.
- **http_request** (*dict*) – (optional) info about HTTP request associated with the entry
- **resource** (*Resource*) – Monitored resource of the entry, defaults to the global resource type.
- **timestamp** (*datetime.datetime*) – (optional) timestamp of event being logged.

path

URI path for use in logging APIs

project

Project bound to the logger.

17.3 Entries

Log entries within the Google Stackdriver Logging API.

```
class google.cloud.logging.entries.ProtobufEntry (payload, logger, insert_id=None,
                                                    timestamp=None, labels=None,
                                                    severity=None, http_request=None,
                                                    resource=None)
```

Bases: `google.cloud.logging.entries._BaseEntry`

Entry created with `protoPayload`.

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/LogEntry>

Parameters

- **payload** (*str*, *dict* or *any_pb2.Any*) – The payload passed as `textPayload`, `jsonPayload`, or `protoPayload`. This also may be passed as a raw `any_pb2.Any` if the `protoPayload` could not be deserialized.
- **logger** (*Logger*) – the logger used to write the entry.
- **insert_id** (*str*) – (optional) the ID used to identify an entry uniquely.
- **timestamp** (*datetime.datetime*) – (optional) timestamp for the entry
- **labels** (*dict*) – (optional) mapping of labels for the entry
- **severity** (*str*) – (optional) severity of event being logged.
- **http_request** (*dict*) – (optional) info about HTTP request associated with the entry
- **resource** (*Resource*) – (Optional) Monitored resource of the entry

parse_message (*message*)

Parse payload into a protobuf message.

Mutates the passed-in message in place.

Parameters *message* (*Protobuf message*) – the message to be logged

```
class google.cloud.logging.entries.StructEntry (payload, logger, insert_id=None,
                                                timestamp=None, labels=None,
                                                severity=None, http_request=None,
                                                resource=None)
```

Bases: `google.cloud.logging.entries._BaseEntry`

Entry created with `jsonPayload`.

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/LogEntry>

```
class google.cloud.logging.entries.TextEntry (payload, logger, insert_id=None, times-
                                              tamp=None, labels=None, severity=None,
                                              http_request=None, resource=None)
```

Bases: `google.cloud.logging.entries._BaseEntry`

Entry created with `textPayload`.

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/LogEntry>

`google.cloud.logging.entries.logger_name_from_path` (*path*)

Validate a logger URI path and get the logger name.

Parameters *path* (*str*) – URI path for a logger API request.

Return type *str*

Returns Logger name parsed from path.

Raises `ValueError` if the path is ill-formed or if the project from the path does not agree with the project passed in.

17.4 Metrics

Define Stackdriver Logging API Metrics.

```
class google.cloud.logging.metric.Metric(name, filter_=None, client=None, description="")
```

Bases: `object`

Metrics represent named filters for log entries.

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/projects.metrics>

Parameters

- **name** (*str*) – the name of the metric
- **filter** (*str*) – the advanced logs filter expression defining the entries tracked by the metric. If not passed, the instance should already exist, to be refreshed via `reload()`.
- **client** (*google.cloud.logging.client.Client*) – A client which holds credentials and project configuration for the metric (which requires a project).
- **description** (*str*) – an optional description of the metric.

client

Client bound to the logger.

create (*client=None*)

API call: create the metric via a PUT request

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/projects.metrics/create>

Parameters **client** (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current metric.

delete (*client=None*)

API call: delete a metric via a DELETE request

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/projects.metrics/delete>

Parameters **client** (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current metric.

exists (*client=None*)

API call: test for the existence of the metric via a GET request

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/projects.metrics/get>

Parameters **client** (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current metric.

Return type `bool`

Returns Boolean indicating existence of the metric.

classmethod from_api_repr (*resource, client*)

Factory: construct a metric given its API representation

Parameters

- **resource** (*dict*) – metric resource representation returned from the API
- **client** (*google.cloud.logging.client.Client*) – Client which holds credentials and project configuration for the metric.

Return type *google.cloud.logging.metric.Metric*

Returns Metric parsed from `resource`.

full_name

Fully-qualified name used in metric APIs

path

URL path for the metric's APIs

project

Project bound to the logger.

reload (*client=None*)

API call: sync local metric configuration via a GET request

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/projects.metrics/get>

Parameters **client** (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current metric.

update (*client=None*)

API call: update metric configuration via a PUT request

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/projects.metrics/update>

Parameters **client** (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current metric.

17.5 Sinks

Define Stackdriver Logging API Sinks.

class `google.cloud.logging.sink.Sink` (*name, filter_=None, destination=None, client=None*)

Bases: `object`

Sinks represent filtered exports for log entries.

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/projects.sinks>

Parameters

- **name** (*str*) – the name of the sink
- **filter** (*str*) – the advanced logs filter expression defining the entries exported by the sink. If not passed, the instance should already exist, to be refreshed via `reload()`.
- **destination** (*str*) – destination URI for the entries exported by the sink. If not passed, the instance should already exist, to be refreshed via `reload()`.
- **client** (`google.cloud.logging.client.Client`) – A client which holds credentials and project configuration for the sink (which requires a project).

client

Client bound to the sink.

create (*client=None*)

API call: create the sink via a PUT request

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/projects.sinks/create>

Parameters **client** (*Client* or *NoneType*) – the client to use. If not passed, falls back to the `client` stored on the current sink.

delete (*client=None*)

API call: delete a sink via a DELETE request

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/projects.sinks/delete>

Parameters `client` (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current sink.

exists (*client=None*)

API call: test for the existence of the sink via a GET request

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/projects.sinks/get>

Parameters `client` (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current sink.

Return type `bool`

Returns Boolean indicating existence of the sink.

classmethod `from_api_repr` (*resource, client*)

Factory: construct a sink given its API representation

Parameters

- **resource** (*dict*) – sink resource representation returned from the API
- **client** (*google.cloud.logging.client.Client*) – Client which holds credentials and project configuration for the sink.

Return type *google.cloud.logging.sink.Sink*

Returns Sink parsed from `resource`.

Raises `ValueError` if `client` is not `None` and the project from the resource does not agree with the project from the client.

full_name

Fully-qualified name used in sink APIs

path

URL path for the sink's APIs

project

Project bound to the sink.

reload (*client=None*)

API call: sync local sink configuration via a GET request

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/projects.sinks/get>

Parameters `client` (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current sink.

update (*client=None*)

API call: update sink configuration via a PUT request

See <https://cloud.google.com/logging/docs/reference/v2/rest/v2/projects.sinks/update>

Parameters `client` (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current sink.

17.6 Integration with Python logging module

It's possible to tie the Python `logging` module directly into Google Cloud Logging. To use it, create a `CloudLoggingHandler` instance from your Logging client.

```
>>> import logging
>>> import google.cloud.logging # Don't conflict with standard logging
>>> from google.cloud.logging.handlers import CloudLoggingHandler
>>> client = google.cloud.logging.Client()
>>> handler = CloudLoggingHandler(client)
>>> cloud_logger = logging.getLogger('cloudLogger')
>>> cloud_logger.setLevel(logging.INFO) # defaults to WARN
>>> cloud_logger.addHandler(handler)
>>> cloud_logger.error('bad news')
```

Note:

This handler by default uses an asynchronous transport that sends log entries on a background thread. However, the API call will still be made in the same process. For other transport options, see the transports section.

All logs will go to a single custom log, which defaults to “python”. The name of the Python logger will be included in the structured log entry under the “python_logger” field. You can change it by providing a name to the handler:

```
>>> handler = CloudLoggingHandler(client, name="mycustomlog")
```

It is also possible to attach the handler to the root Python logger, so that for example a plain `logging.warn` call would be sent to Cloud Logging, as well as any other loggers created. However, you must avoid infinite recursion from the logging calls the client itself makes. A helper method `setup_logging` is provided to configure this automatically:

```
>>> import logging
>>> import google.cloud.logging # Don't conflict with standard logging
>>> from google.cloud.logging.handlers import CloudLoggingHandler, setup_logging
>>> client = google.cloud.logging.Client()
>>> handler = CloudLoggingHandler(client)
>>> logging.getLogger().setLevel(logging.INFO) # defaults to WARN
>>> setup_logging(handler)
>>> logging.error('bad news')
```

You can also exclude certain loggers:

```
>>> setup_logging(handler, excluded_loggers=('werkzeug',))
```

17.6.1 Python logging handler transports

The Python logging handler can use different transports. The default is `google.cloud.logging.handlers.BackgroundThreadTransport`.

1. `google.cloud.logging.handlers.BackgroundThreadTransport` this is the default. It writes entries on a background `python.threading.Thread`.
1. `google.cloud.logging.handlers.SyncTransport` this handler does a direct API call on each logging statement to write the entry.

17.7 Python Logging Module Handler

Python `logging` handlers for Stackdriver Logging.

```
class google.cloud.logging.handlers.handlers.CloudLoggingHandler (client,
                                                                name='python',
                                                                trans-
                                                                port=<class
                                                                'google.cloud.logging.handlers.transp
                                                                re-
                                                                source=Resource(type='global',
                                                                labels={}),
                                                                la-
                                                                bels=None)
```

Bases: `logging.StreamHandler`

Handler that directly makes Stackdriver logging API calls.

This is a Python standard `logging` handler using that can be used to route Python standard logging messages directly to the Stackdriver Logging API.

This handler supports both an asynchronous and synchronous transport.

Parameters

- **client** (*google.cloud.logging.client*) – the authenticated Google Cloud Logging client for this handler to use
- **name** (*str*) – the name of the custom log in Stackdriver Logging. Defaults to 'python'. The name of the Python logger will be represented in the `python_logger` field.
- **transport** (*type*) – Class for creating new transport objects. It should extend from the base *Transport* type and implement `:meth'.Transport.send'`. Defaults to *BackgroundThreadTransport*. The other option is *SyncTransport*.
- **resource** (*Resource*) – (Optional) Monitored resource of the entry, defaults to the global resource type.
- **labels** (*dict*) – (Optional) Mapping of labels for the entry.

Example:

```
import logging
import google.cloud.logging
from google.cloud.logging.handlers import CloudLoggingHandler

client = google.cloud.logging.Client()
handler = CloudLoggingHandler(client)

cloud_logger = logging.getLogger('cloudLogger')
cloud_logger.setLevel(logging.INFO)
cloud_logger.addHandler(handler)

cloud_logger.error('bad news') # API call
```

emit (*record*)

Actually log the specified logging record.

Overrides the default emit behavior of `StreamHandler`.

See <https://docs.python.org/2/library/logging.html#handler-objects>

Parameters **record** (*logging.LogRecord*) – The record to be logged.

```
google.cloud.logging.handlers.handlers.setup_logging(handler, excluded_loggers=('google.cloud',
                                                                    'google.auth',
                                                                    'google_auth_httplib2'),
                                                                    log_level=20)
```

Attach a logging handler to the Python root logger

Excludes loggers that this library itself uses to avoid infinite recursion.

Parameters

- **handler** (`logging.handler`) – the handler to attach to the global handler
- **excluded_loggers** (`tuple`) – (Optional) The loggers to not attach the handler to. This will always include the loggers in the path of the logging client itself.
- **log_level** (`int`) – (Optional) Python logging log level. Defaults to `logging.INFO`.

Example:

```
import logging
import google.cloud.logging
from google.cloud.logging.handlers import CloudLoggingHandler

client = google.cloud.logging.Client()
handler = CloudLoggingHandler(client)
google.cloud.logging.handlers.setup_logging(handler)
logging.getLogger().setLevel(logging.DEBUG)

logging.error('bad news') # API call
```

17.8 Google App Engine flexible Log Handler

Logging handler for App Engine Flexible

Sends logs to the Stackdriver Logging API with the appropriate resource and labels for App Engine logs.

```
class google.cloud.logging.handlers.app_engine.AppEngineHandler(client, transport=<class
                                                                    'google.cloud.logging.handlers.transpo
```

Bases: `google.cloud.logging.handlers.handlers.CloudLoggingHandler`

A logging handler that sends App Engine-formatted logs to Stackdriver.

Parameters

- **client** (`Client`) – The authenticated Google Cloud Logging client for this handler to use.
- **transport** (`type`) – The transport class. It should be a subclass of `Transport`. If unspecified, `BackgroundThreadTransport` will be used.

get_gae_labels()

Return the labels for GAE app.

If the trace ID can be detected, it will be included as a label. Currently, no other labels are included.

Return type `dict`

Returns Labels for GAE app.

get_gae_resource()

Return the GAE resource using the environment variables.

Return type `Resource`

Returns Monitored resource for GAE.

17.9 Google Container Engine Log Handler

Logging handler for Google Container Engine (GKE).

Formats log messages in a JSON format, so that Kubernetes clusters with the fluentd Google Cloud plugin installed can format their log messages so that metadata such as log level is properly captured.

class `google.cloud.logging.handlers.container_engine.ContainerEngineHandler` (*stream=None*)

Bases: `logging.StreamHandler`

Handler to format log messages the format expected by GKE fluent.

This handler is written to format messages for the Google Container Engine (GKE) fluentd plugin, so that metadata such as log level are properly set.

Initialize the handler.

If stream is not specified, `sys.stderr` is used.

format (*record*)

Format the message into JSON expected by fluentd.

Parameters **record** (`LogRecord`) – the log record

Return type `str`

Returns A JSON string formatted for GKE fluentd.

17.10 Python Logging Handler Sync Transport

Transport for Python logging handler.

Logs directly to the the Stackdriver Logging API with a synchronous call.

class `google.cloud.logging.handlers.transports.sync.SyncTransport` (*client*, *name*)

Bases: `google.cloud.logging.handlers.transports.base.Transport`

Basic synchronous transport.

Uses this library's Logging client to directly make the API call.

send (*record*, *message*, *resource=None*, *labels=None*)

Overrides `transport.send()`.

Parameters

- **record** (`logging.LogRecord`) – Python log record that the handler was called with.
- **message** (`str`) – The message from the `LogRecord` after being formatted by the associated log formatters.
- **resource** (`Resource`) – (Optional) Monitored resource of the entry.
- **labels** (`dict`) – (Optional) Mapping of labels for the entry.

17.11 Python Logging Handler Threaded Transport

Transport for Python logging handler

Uses a background worker to log to Stackdriver Logging asynchronously.

class google.cloud.logging.handlers.transports.background_thread.**BackgroundThreadTransport**

Bases: `google.cloud.logging.handlers.transports.base.Transport`

Asynchronous transport that uses a background thread.

Parameters

- **client** (*Client*) – The Logging client.
- **name** (*str*) – the name of the logger.
- **grace_period** (*float*) – The amount of time to wait for pending logs to be submitted when the process is shutting down.
- **batch_size** (*int*) – The maximum number of items to send at a time in the background thread.

flush ()

Submit any pending log records.

send (*record, message, resource=None, labels=None*)

Overrides Transport.send().

Parameters

- **record** (`logging.LogRecord`) – Python log record that the handler was called with.
- **message** (*str*) – The message from the LogRecord after being formatted by the associated log formatters.
- **resource** (`Resource`) – (Optional) Monitored resource of the entry.
- **labels** (*dict*) – (Optional) Mapping of labels for the entry.

17.12 Python Logging Handler Sync Transport

Module containing base class for logging transport.

class google.cloud.logging.handlers.transports.base.**Transport**

Bases: `object`

Base class for Google Cloud Logging handler transports.

Subclasses of `Transport` must have constructors that accept a client and name object, and must override `send()`.

flush ()

Submit any pending log records.

For blocking/sync transports, this is a no-op.

send (*record, message, resource=None, labels=None*)

Transport send to be implemented by subclasses.

Parameters

- **record** (`logging.LogRecord`) – Python log record that the handler was called with.
- **message** (`str`) – The message from the `LogRecord` after being formatted by the associated log formatters.
- **resource** (`Resource`) – (Optional) Monitored resource of the entry.
- **labels** (`dict`) – (Optional) Mapping of labels for the entry.

17.13 Authentication and Configuration

- For an overview of authentication in `google-cloud-python`, see [Authentication](#).
- In addition to any authentication configuration, you should also set the `GOOGLE_CLOUD_PROJECT` environment variable for the project you'd like to interact with. If you are Google App Engine or Google Compute Engine this will be detected automatically.
- The library now enables the gRPC transport for the logging API by default, assuming that the required dependencies are installed and importable. To *disable* this transport, set the `GOOGLE_CLOUD_DISABLE_GRPC` environment variable to a non-empty string, e.g.: `$ export GOOGLE_CLOUD_DISABLE_GRPC=true`.
- After configuring your environment, create a `Client`

```
from google.cloud import logging
client = logging.Client()
```

or pass in credentials and project explicitly

```
from google.cloud import logging
client = logging.Client(project='my-project', credentials=credentials)
```

17.14 Writing log entries

To write log entries, first create a `Logger`, passing the “log name” with which to associate the entries:

```
logger = client.logger(LOG_NAME)
```

Write a simple text entry to the logger.

```
logger.log_text("A simple entry") # API call
```

Write a dictionary entry to the logger.

```
logger.log_struct({
    'message': 'My second entry',
    'weather': 'partly cloudy',
}) # API call
```

17.15 Retrieving log entries

Fetch entries for the default project.

```
for entry in client.list_entries(): # API call(s)
    do_something_with(entry)
```

Fetch entries across multiple projects.

```
PROJECT_IDS = ['one-project', 'another-project']
for entry in client.list_entries(project_ids=PROJECT_IDS): # API call(s)
    do_something_with(entry)
```

Filter entries retrieved using the [Advanced Logs Filters](#) syntax

Fetch entries for the default project.

```
FILTER = 'logName:log_name AND textPayload:simple'
for entry in client.list_entries(filter_=FILTER): # API call(s)
    do_something_with(entry)
```

Sort entries in descending timestamp order.

```
from google.cloud.logging import DESCENDING
for entry in client.list_entries(order_by=DESCENDING): # API call(s)
    do_something_with(entry)
```

Retrieve entries in batches of 10, iterating until done.

```
iterator = client.list_entries()
pages = iterator.pages

page1 = next(pages) # API call
for entry in page1:
    do_something_with(entry)

page2 = next(pages) # API call
for entry in page2:
    do_something_with(entry)
```

Retrieve entries for a single logger, sorting in descending timestamp order:

```
from google.cloud.logging import DESCENDING
for entry in logger.list_entries(order_by=DESCENDING): # API call(s)
    do_something_with(entry)
```

17.16 Delete all entries for a logger

```
logger.delete() # API call
```

17.17 Manage log metrics

Metrics are counters of entries which match a given filter. They can be used within Stackdriver Monitoring to create charts and alerts.

List all metrics for a project:


```
for metric in client.list_metrics(): # API call(s)
    do_something_with(metric)
```

Create a metric:

```
metric = client.metric(
    METRIC_NAME, filter_=FILTER, description=DESCRIPTION)
assert not metric.exists() # API call
metric.create() # API call
assert metric.exists() # API call
```

Refresh local information about a metric:

```
existing_metric = client.metric(METRIC_NAME)
existing_metric.reload() # API call
```

Update a metric:

```
existing_metric.filter_ = UPDATED_FILTER
existing_metric.description = UPDATED_DESCRIPTION
existing_metric.update() # API call
```

Delete a metric:

```
metric.delete()
```

17.18 Export log entries using sinks

Sinks allow exporting entries which match a given filter to Cloud Storage buckets, BigQuery datasets, or Cloud Pub/Sub topics.

17.18.1 Export to Cloud Storage

Make sure that the storage bucket you want to export logs too has `cloud-logs@google.com` as the owner. See [Setting permissions for Cloud Storage](#).

Add `cloud-logs@google.com` as the owner of the bucket:

```
bucket.acl.reload() # API call
logs_group = bucket.acl.group('cloud-logs@google.com')
logs_group.grant_owner()
bucket.acl.add_entity(logs_group)
bucket.acl.save() # API call
```

Create a Cloud Storage sink:

```
DESTINATION = 'storage.googleapis.com/%s' % (bucket.name,)
sink = client.sink(SINK_NAME, filter_=FILTER, destination=DESTINATION)
assert not sink.exists() # API call
sink.create() # API call
assert sink.exists() # API call
```

17.18.2 Export to BigQuery

To export logs to BigQuery you must log into the Cloud Platform Console and add `cloud-logs@google.com` to a dataset.

See: [Setting permissions for BigQuery](#)

```
from google.cloud.bigquery.dataset import AccessGrant
grants = dataset.access_grants
grants.append(AccessGrant(
    'WRITER', 'groupByEmail', 'cloud-logs@google.com'))
dataset.access_grants = grants
dataset.update()  # API call
```

Create a BigQuery sink:

```
DESTINATION = 'bigquery.googleapis.com%s' % (dataset.path,)
sink = client.sink(SINK_NAME, filter_=FILTER, destination=DESTINATION)
assert not sink.exists()  # API call
sink.create()  # API call
assert sink.exists()  # API call
```

17.18.3 Export to Pub/Sub

To export logs to BigQuery you must log into the Cloud Platform Console and add `cloud-logs@google.com` to a topic.

See: [Setting permissions for Pub/Sub](#)

```
policy = topic.get_iam_policy()  # API call
policy.owners.add(policy.group('cloud-logs@google.com'))
topic.set_iam_policy(policy)  # API call
```

Create a Cloud Pub/Sub sink:

```
DESTINATION = 'pubsub.googleapis.com/%s' % (topic.full_name,)
sink = client.sink(SINK_NAME, filter_=FILTER, destination=DESTINATION)
assert not sink.exists()  # API call
sink.create()  # API call
assert sink.exists()  # API call
```

17.18.4 Manage Sinks

List all sinks for a project:

```
for sink in client.list_sinks():  # API call(s)
    do_something_with(sink)
```

Refresh local information about a sink:

```
existing_sink = client.sink(SINK_NAME)
existing_sink.reload()
```

Update a sink:

```
existing_sink.filter_ = UPDATED_FILTER
existing_sink.update()
```

Delete a sink:

```
sink.delete()
```

17.19 Integration with Python logging module

It's possible to tie the Python `logging` module directly into Google Stackdriver Logging. There are different handler options to accomplish this. To automatically pick the default for your current environment, use `get_default_handler()`.

```
import logging
handler = client.get_default_handler()
cloud_logger = logging.getLogger('cloudLogger')
cloud_logger.setLevel(logging.INFO)
cloud_logger.addHandler(handler)
cloud_logger.error('bad news')
```

It is also possible to attach the handler to the root Python logger, so that for example a plain `logging.warn` call would be sent to Stackdriver Logging, as well as any other loggers created. A helper method `setup_logging()` is provided to configure this automatically.

```
client.setup_logging(log_level=logging.INFO)
```

Note: To reduce cost and quota usage, do not enable Stackdriver logging handlers while testing locally.

You can also exclude certain loggers:

```
client.setup_logging(log_level=logging.INFO,
                    excluded_loggers=('werkzeug',))
```

17.19.1 Cloud Logging Handler

If you prefer not to use `get_default_handler()`, you can directly create a `CloudLoggingHandler` instance which will write directly to the API.

```
from google.cloud.logging.handlers import CloudLoggingHandler
handler = CloudLoggingHandler(client)
cloud_logger = logging.getLogger('cloudLogger')
cloud_logger.setLevel(logging.INFO)
cloud_logger.addHandler(handler)
cloud_logger.error('bad news')
```

Note: This handler by default uses an asynchronous transport that sends log entries on a background thread. However, the API call will still be made in the same process. For other transport options, see the transports section.

All logs will go to a single custom log, which defaults to “python”. The name of the Python logger will be included in the structured log entry under the “python_logger” field. You can change it by providing a name to the handler:

```
handler = CloudLoggingHandler(client, name='mycustomlog')
```

17.19.2 Cloud Logging Handler transports

The `CloudLoggingHandler` logging handler can use different transports. The default is `BackgroundThreadTransport`.

1. `BackgroundThreadTransport` this is the default. It writes entries on a background python . `threading.Thread`.
1. `SyncTransport` this handler does a direct API call on each logging statement to write the entry.

17.19.3 fluentd logging handlers

Besides `CloudLoggingHandler`, which writes directly to the API, two other handlers are provided. `AppEngineHandler`, which is recommended when running on the Google App Engine Flexible vanilla runtimes (i.e. your `app.yaml` contains `runtime: python`), and `ContainerEngineHandler`, which is recommended when running on Google Container Engine with the Stackdriver Logging plugin enabled.

`get_default_handler()` and `setup_logging()` will attempt to use the environment to automatically detect whether the code is running in these platforms and use the appropriate handler.

In both cases, the fluentd agent is configured to automatically parse log files in an expected format and forward them to Stackdriver logging. The handlers provided help set the correct metadata such as log level so that logs can be filtered accordingly.

18.1 Blobs / Objects

Create / interact with Google Cloud Storage blobs.

```
class google.cloud.storage.blob.Blob (name, bucket, chunk_size=None, encryption_key=None)
```

Bases: `google.cloud.storage._helpers._PropertyMixin`

A wrapper around Cloud Storage's concept of an Object.

Parameters

- **name** (*str*) – The name of the blob. This corresponds to the unique path of the object in the bucket. If bytes, will be converted to a unicode object. Blob / object names can contain any sequence of valid unicode characters, of length 1-1024 bytes when UTF-8 encoded.
- **bucket** (*google.cloud.storage.bucket.Bucket*) – The bucket to which this blob belongs.
- **chunk_size** (*int*) – The size of a chunk of data whenever iterating (1 MB). This must be a multiple of 256 KB per the API specification.
- **encryption_key** (*bytes*) – Optional 32 byte encryption key for customer-supplied encryption. See <https://cloud.google.com/storage/docs/encryption#customer-supplied>.

acl

Create our ACL on demand.

cache_control

HTTP 'Cache-Control' header for this object.

See [RFC 7234](#) and [API reference docs](#).

If the property is not set locally, returns `None`.

Return type `str` or `NoneType`

chunk_size

Get the blob's default chunk size.

Return type `int` or `NoneType`

Returns The current blob's chunk size, if it is set.

client

The client bound to this blob.

component_count

Number of underlying components that make up this object.

See https://cloud.google.com/storage/docs/json_api/v1/objects

Return type `int` or `NoneType`

Returns The component count (in case of a composed object) or `None` if the property is not set locally. This property will not be set on objects not created via `compose`.

compose (*sources*, *client=None*)

Concatenate source blobs into this one.

Parameters

- **sources** (list of *Blob*) – blobs whose contents will be composed into this blob.
- **client** (*Client* or `NoneType`) – Optional. The client to use. If not passed, falls back to the `client` stored on the blob's bucket.

Raises `ValueError` if this blob does not have its `content_type` set.

content_disposition

HTTP 'Content-Disposition' header for this object.

See [RFC 6266](#) and [API reference docs](#).

If the property is not set locally, returns `None`.

Return type `str` or `NoneType`

content_encoding

HTTP 'Content-Encoding' header for this object.

See [RFC 7231](#) and [API reference docs](#).

If the property is not set locally, returns `None`.

Return type `str` or `NoneType`

content_language

HTTP 'Content-Language' header for this object.

See [BCP47](#) and [API reference docs](#).

If the property is not set locally, returns `None`.

Return type `str` or `NoneType`

content_type

HTTP 'Content-Type' header for this object.

See [RFC 2616](#) and [API reference docs](#).

If the property is not set locally, returns `None`.

Return type `str` or `NoneType`

crc32c

CRC32C checksum for this object.

See [RFC 4960](#) and [API reference docs](#).

If the property is not set locally, returns `None`.

Return type `str` or `NoneType`

create_resumable_upload_session (*content_type=None, size=None, origin=None, client=None*)

Create a resumable upload session.

Resumable upload sessions allow you to start an upload session from one client and complete the session in another. This method is called by the initiator to set the metadata and limits. The initiator then passes the session URL to the client that will upload the binary data. The client performs a PUT request on the session URL to complete the upload. This process allows untrusted clients to upload to an access-controlled bucket. For more details, see the [documentation on signed URLs](#).

The content type of the upload will be determined in order of precedence:

- The value passed in to this method (if not `None`)
- The value stored on the current blob
- The default value ('application/octet-stream')

Note: The effect of uploading to an existing blob depends on the “versioning” and “lifecycle” policies defined on the blob’s bucket. In the absence of those policies, upload will overwrite any existing contents.

See the [object versioning](#) and [lifecycle](#) API documents for details.

If `encryption_key` is set, the blob will be encrypted with a [customer-supplied](#) encryption key.

Parameters

- **size** (*int*) – (Optional). The maximum number of bytes that can be uploaded using this session. If the size is not known when creating the session, this should be left blank.
- **content_type** (*str*) – (Optional) Type of content being uploaded.
- **origin** (*str*) – (Optional) If set, the upload can only be completed by a user-agent that uploads from the given origin. This can be useful when passing the session to a web client.
- **client** (*Client*) – (Optional) The client to use. If not passed, falls back to the `client` stored on the blob’s bucket.

Return type `str`

Returns The resumable upload session URL. The upload can be completed by making an HTTP PUT request with the file’s contents.

Raises `google.cloud.exceptions.GoogleCloudError` if the session creation response returns an error status.

delete (*client=None*)

Deletes a blob from Cloud Storage.

Parameters **client** (*Client* or `NoneType`) – Optional. The client to use. If not passed, falls back to the `client` stored on the blob’s bucket.

Return type `Blob`

Returns The blob that was just deleted.

Raises `google.cloud.exceptions.NotFound` (propagated from `google.cloud.storage.bucket.Bucket.delete_blob()`).

download_as_string (*client=None*)

Download the contents of this blob as a string.

Parameters **client** (*Client* or `NoneType`) – Optional. The client to use. If not passed, falls back to the `client` stored on the blob's bucket.

Return type `bytes`

Returns The data stored in this blob.

Raises `google.cloud.exceptions.NotFound`

download_to_file (*file_obj, client=None*)

Download the contents of this blob into a file-like object.

Note: If the server-set property, `media_link`, is not yet initialized, makes an additional API request to load it.

Downloading a file that has been encrypted with a `customer-supplied` encryption key:

```
from google.cloud.storage import Blob

client = storage.Client(project='my-project')
bucket = client.get_bucket('my-bucket')
encryption_key = 'c7f32af42e45e85b9848a6a14dd2a8f6'
blob = Blob('secure-data', bucket, encryption_key=encryption_key)
blob.upload_from_string('my secret message.')
with open('/tmp/my-secure-file', 'wb') as file_obj:
    blob.download_to_file(file_obj)
```

The `encryption_key` should be a str or bytes with a length of at least 32.

For more fine-grained over the download process, check out [google-resumable-media](#). For example, this library allows downloading **parts** of a blob rather than the whole thing.

Parameters

- **file_obj** (*file*) – A file handle to which to write the blob's data.
- **client** (*Client* or `NoneType`) – Optional. The client to use. If not passed, falls back to the `client` stored on the blob's bucket.

Raises `google.cloud.exceptions.NotFound`

download_to_filename (*filename, client=None*)

Download the contents of this blob into a named file.

Parameters

- **filename** (*str*) – A filename to be passed to `open`.
- **client** (*Client* or `NoneType`) – Optional. The client to use. If not passed, falls back to the `client` stored on the blob's bucket.

Raises `google.cloud.exceptions.NotFound`

etag

Retrieve the ETag for the object.

See [RFC 2616 \(etags\)](#) and [API reference docs](#).

Return type `str` or `NoneType`

Returns The blob etag or `None` if the property is not set locally.

exists (*client=None*)

Determines whether or not this blob exists.

Parameters **client** (*Client* or `NoneType`) – Optional. The client to use. If not passed, falls back to the `client` stored on the blob's bucket.

Return type `bool`

Returns True if the blob exists in Cloud Storage.

generate_signed_url (*expiration, method='GET', content_type=None, generation=None, response_disposition=None, response_type=None, client=None, credentials=None*)

Generates a signed URL for this blob.

Note: If you are on Google Compute Engine, you can't generate a signed URL. Follow [Issue 50](#) for updates on this. If you'd like to be able to generate a signed URL from GCE, you can use a standard service account from a JSON file rather than a GCE service account.

If you have a blob that you want to allow access to for a set amount of time, you can use this method to generate a URL that is only valid within a certain time period.

This is particularly useful if you don't want publicly accessible blobs, but don't want to require users to explicitly log in.

Parameters

- **expiration** (*int, long, datetime.datetime, datetime.timedelta*) – When the signed URL should expire.
- **method** (*str*) – The HTTP verb that will be used when requesting the URL.
- **content_type** (*str*) – (Optional) The content type of the object referenced by resource.
- **generation** (*str*) – (Optional) A value that indicates which generation of the resource to fetch.
- **response_disposition** (*str*) – (Optional) Content disposition of responses to requests for the signed URL. For example, to enable the signed URL to initiate a file of `blog.png`, use the value `'attachment; filename=blog.png'`.
- **response_type** (*str*) – (Optional) Content type of responses to requests for the signed URL. Used to over-ride the content type of the underlying blob/object.
- **client** (*Client* or `NoneType`) – (Optional) The client to use. If not passed, falls back to the `client` stored on the blob's bucket.
- **credentials** (*oauth2client.client.OAuth2Credentials* or `NoneType`) – (Optional) The OAuth2 credentials to use to sign the URL. Defaults to the credentials stored on the client used.

Return type `str`

Returns A signed URL you can use to access the resource until expiration.

generation

Retrieve the generation for the object.

See https://cloud.google.com/storage/docs/json_api/v1/objects

Return type `int` or `NoneType`

Returns The generation of the blob or `None` if the property is not set locally.

get_iam_policy (*client=None*)

Retrieve the IAM policy for the object.

See https://cloud.google.com/storage/docs/json_api/v1/objects/getIamPolicy

Parameters **client** (*Client* or `NoneType`) – Optional. The client to use. If not passed, falls back to the `client` stored on the current object's bucket.

Return type `google.cloud.iam.Policy`

Returns the policy instance, based on the resource returned from the `getIamPolicy` API request.

id

Retrieve the ID for the object.

See https://cloud.google.com/storage/docs/json_api/v1/objects

Return type `str` or `NoneType`

Returns The ID of the blob or `None` if the property is not set locally.

make_public (*client=None*)

Make this blob public giving all users read access.

Parameters **client** (*Client* or `NoneType`) – Optional. The client to use. If not passed, falls back to the `client` stored on the blob's bucket.

md5_hash

MD5 hash for this object.

See [RFC 1321](#) and [API reference docs](#).

If the property is not set locally, returns `None`.

Return type `str` or `NoneType`

media_link

Retrieve the media download URI for the object.

See https://cloud.google.com/storage/docs/json_api/v1/objects

Return type `str` or `NoneType`

Returns The media link for the blob or `None` if the property is not set locally.

metadata

Retrieve arbitrary/application specific metadata for the object.

See https://cloud.google.com/storage/docs/json_api/v1/objects

Setter Update arbitrary/application specific metadata for the object.

Getter Retrieve arbitrary/application specific metadata for the object.

Return type `dict` or `NoneType`

Returns The metadata associated with the blob or `None` if the property is not set locally.

metageneration

Retrieve the metageneration for the object.

See https://cloud.google.com/storage/docs/json_api/v1/objects

Return type `int` or `NoneType`

Returns The metageneration of the blob or `None` if the property is not set locally.

owner

Retrieve info about the owner of the object.

See https://cloud.google.com/storage/docs/json_api/v1/objects

Return type `dict` or `NoneType`

Returns Mapping of owner's role/ID. If the property is not set locally, returns `None`.

patch (*client=None*)

Sends all changed properties in a PATCH request.

Updates the `_properties` with the response from the backend.

Parameters **client** (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current object.

path

Getter property for the URL path to this Blob.

Return type `str`

Returns The URL path to this Blob.

static path_helper (*bucket_path, blob_name*)

Relative URL path for a blob.

Parameters

- **bucket_path** (*str*) – The URL path for a bucket.
- **blob_name** (*str*) – The name of the blob.

Return type `str`

Returns The relative URL path for `blob_name`.

public_url

The public URL for this blob's object.

Return type `string`

Returns The public URL for this blob.

reload (*client=None*)

Reload properties from Cloud Storage.

Parameters **client** (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current object.

rewrite (*source, token=None, client=None*)

Rewrite source blob into this one.

Parameters

- **source** (*Blob*) – blob whose contents will be rewritten into this blob.
- **token** (*str*) – Optional. Token returned from an earlier, not-completed call to rewrite the same source blob. If passed, result will include updated status, total bytes written.
- **client** (*Client* or `NoneType`) – Optional. The client to use. If not passed, falls back to the `client` stored on the blob's bucket.

Return type `tuple`

Returns `(token, bytes_rewritten, total_bytes)`, where `token` is a rewrite token (`None` if the rewrite is complete), `bytes_rewritten` is the number of bytes rewritten so far, and `total_bytes` is the total number of bytes to be rewritten.

self_link

Retrieve the URI for the object.

See https://cloud.google.com/storage/docs/json_api/v1/objects

Return type `str` or `NoneType`

Returns The self link for the blob or `None` if the property is not set locally.

set_iam_policy (*policy*, *client=None*)

Update the IAM policy for the bucket.

See https://cloud.google.com/storage/docs/json_api/v1/objects/setIamPolicy

Parameters

- **policy** (*google.cloud.iam.Policy*) – policy instance used to update bucket’s IAM policy.
- **client** (*Client* or `NoneType`) – Optional. The client to use. If not passed, falls back to the `client` stored on the current bucket.

Return type *google.cloud.iam.Policy*

Returns the policy instance, based on the resource returned from the `setIamPolicy` API request.

size

Size of the object, in bytes.

See https://cloud.google.com/storage/docs/json_api/v1/objects

Return type `int` or `NoneType`

Returns The size of the blob or `None` if the property is not set locally.

storage_class

Retrieve the storage class for the object.

This can only be set at blob / object **creation** time. If you’d like to change the storage class **after** the blob / object already exists in a bucket, call `update_storage_class()` (which uses the “`storage.objects.rewrite`” method).

See <https://cloud.google.com/storage/docs/storage-classes>

Return type `str` or `NoneType`

Returns If set, one of “MULTI_REGIONAL”, “REGIONAL”, “NEARLINE”, “COLDLINE”, “STANDARD”, or “DURABLE_REDUCED_AVAILABILITY”, else `None`.

test_iam_permissions (*permissions*, *client=None*)

API call: test permissions

See https://cloud.google.com/storage/docs/json_api/v1/objects/testIamPermissions

Parameters

- **permissions** (*list of string*) – the permissions to check
- **client** (*Client* or `NoneType`) – Optional. The client to use. If not passed, falls back to the `client` stored on the current bucket.

Return type list of string

Returns the permissions returned by the `testIamPermissions` API request.

time_created

Retrieve the timestamp at which the object was created.

See https://cloud.google.com/storage/docs/json_api/v1/objects

Return type `datetime.datetime` or `NoneType`

Returns Datetime object parsed from RFC3339 valid timestamp, or `None` if the property is not set locally.

time_deleted

Retrieve the timestamp at which the object was deleted.

See https://cloud.google.com/storage/docs/json_api/v1/objects

Return type `datetime.datetime` or `NoneType`

Returns Datetime object parsed from RFC3339 valid timestamp, or `None` if the property is not set locally. If the blob has not been deleted, this will never be set.

update (*client=None*)

Sends all properties in a PUT request.

Updates the `_properties` with the response from the backend.

Parameters **client** (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current object.

update_storage_class (*new_class, client=None*)

Update blob's storage class via a rewrite-in-place.

See <https://cloud.google.com/storage/docs/per-object-storage-class>

Parameters

- **new_class** (*str*) – new storage class for the object
- **client** (*Client*) – Optional. The client to use. If not passed, falls back to the `client` stored on the blob's bucket.

updated

Retrieve the timestamp at which the object was updated.

See https://cloud.google.com/storage/docs/json_api/v1/objects

Return type `datetime.datetime` or `NoneType`

Returns Datetime object parsed from RFC3339 valid timestamp, or `None` if the property is not set locally.

upload_from_file (*file_obj, rewind=False, size=None, content_type=None, num_retries=None, client=None*)

Upload the contents of this blob from a file-like object.

The content type of the upload will be determined in order of precedence:

- The value passed in to this method (if not `None`)
- The value stored on the current blob
- The default value ('application/octet-stream')

Note: The effect of uploading to an existing blob depends on the “versioning” and “lifecycle” policies defined on the blob’s bucket. In the absence of those policies, upload will overwrite any existing contents.

See the [object versioning](#) and [lifecycle](#) API documents for details.

Uploading a file with a [customer-supplied](#) encryption key:

```
from google.cloud.storage import Blob

client = storage.Client(project='my-project')
bucket = client.get_bucket('my-bucket')
encryption_key = 'aa426195405adee2c8081bb9e7e74b19'
blob = Blob('secure-data', bucket, encryption_key=encryption_key)
with open('my-file', 'rb') as my_file:
    blob.upload_from_file(my_file)
```

The `encryption_key` should be a str or bytes with a length of at least 32.

For more fine-grained over the upload process, check out [google-resumable-media](#).

Parameters

- **file_obj** (*file*) – A file handle open for reading.
- **rewind** (*bool*) – If True, seek to the beginning of the file handle before writing the file to Cloud Storage.
- **size** (*int*) – The number of bytes to be uploaded (which will be read from `file_obj`). If not provided, the upload will be concluded once `file_obj` is exhausted.
- **content_type** (*str*) – Optional type of content being uploaded.
- **num_retries** (*int*) – Number of upload retries. (Deprecated: This argument will be removed in a future release.)
- **client** (*Client*) – (Optional) The client to use. If not passed, falls back to the `client` stored on the blob’s bucket.

Raises `GoogleCloudError` if the upload response returns an error status.

upload_from_filename (*filename*, *content_type=None*, *client=None*)

Upload this blob’s contents from the content of a named file.

The content type of the upload will be determined in order of precedence:

- The value passed in to this method (if not `None`)
- The value stored on the current blob
- The value given by `mimetypes.guess_type`
- The default value (`'application/octet-stream'`)

Note: The effect of uploading to an existing blob depends on the “versioning” and “lifecycle” policies defined on the blob’s bucket. In the absence of those policies, upload will overwrite any existing contents.

See the [object versioning](#) and [lifecycle](#) API documents for details.

Parameters

- **filename** (*str*) – The path to the file.

- **content_type** (*str*) – Optional type of content being uploaded.
- **client** (*Client*) – (Optional) The client to use. If not passed, falls back to the `client` stored on the blob's bucket.

upload_from_string (*data*, *content_type='text/plain'*, *client=None*)

Upload contents of this blob from the provided string.

Note: The effect of uploading to an existing blob depends on the “versioning” and “lifecycle” policies defined on the blob's bucket. In the absence of those policies, upload will overwrite any existing contents.

See the [object versioning](#) and [lifecycle](#) API documents for details.

Parameters

- **data** (*bytes* or *str*) – The data to store in this blob. If the value is text, it will be encoded as UTF-8.
- **content_type** (*str*) – Optional type of content being uploaded. Defaults to 'text/plain'.
- **client** (*Client* or *NoneType*) – Optional. The client to use. If not passed, falls back to the `client` stored on the blob's bucket.

18.2 Buckets

Create / interact with Google Cloud Storage buckets.

class `google.cloud.storage.bucket.Bucket` (*client*, *name=None*)

Bases: `google.cloud.storage._helpers._PropertyMixin`

A class representing a Bucket on Cloud Storage.

Parameters

- **client** (*google.cloud.storage.client.Client*) – A client which holds credentials and project configuration for the bucket (which requires a project).
- **name** (*str*) – The name of the bucket. Bucket names must start and end with a number or letter.

acl

Create our ACL on demand.

blob (*blob_name*, *chunk_size=None*, *encryption_key=None*)

Factory constructor for blob object.

Note: This will not make an HTTP request; it simply instantiates a blob object owned by this bucket.

Parameters

- **blob_name** (*str*) – The name of the blob to be instantiated.
- **chunk_size** (*int*) – The size of a chunk of data whenever iterating (1 MB). This must be a multiple of 256 KB per the API specification.

- **encryption_key** (*bytes*) – Optional 32 byte encryption key for customer-supplied encryption.

Return type `google.cloud.storage.blob.Blob`

Returns The blob object created.

client

The client bound to this bucket.

configure_website (*main_page_suffix=None, not_found_page=None*)

Configure website-related properties.

See <https://cloud.google.com/storage/docs/hosting-static-website>

Note: This (apparently) only works if your bucket name is a domain name (and to do that, you need to get approved somehow...).

If you want this bucket to host a website, just provide the name of an index page and a page to use when a blob isn't found:

```
client = storage.Client()
bucket = client.get_bucket(bucket_name)
bucket.configure_website('index.html', '404.html')
```

You probably should also make the whole bucket public:

```
bucket.make_public(recursive=True, future=True)
```

This says: “Make the bucket public, and all the stuff already in the bucket, and anything else I add to the bucket. Just make it all public.”

Parameters

- **main_page_suffix** (*str*) – The page to use as the main page of a directory. Typically something like `index.html`.
- **not_found_page** (*str*) – The file to use when a page isn't found.

copy_blob (*blob, destination_bucket, new_name=None, client=None, preserve_acl=True*)

Copy the given blob to the given bucket, optionally with a new name.

Parameters

- **blob** (*google.cloud.storage.blob.Blob*) – The blob to be copied.
- **destination_bucket** (*google.cloud.storage.bucket.Bucket*) – The bucket into which the blob should be copied.
- **new_name** (*str*) – (optional) the new name for the copied file.
- **client** (*Client* or *NoneType*) – Optional. The client to use. If not passed, falls back to the `client` stored on the current bucket.
- **preserve_acl** (*bool*) – Optional. Copies ACL from old blob to new blob. Default: `True`.

Return type `google.cloud.storage.blob.Blob`

Returns The new Blob.

cors

Retrieve or set CORS policies configured for this bucket.

See <http://www.w3.org/TR/cors/> and https://cloud.google.com/storage/docs/json_api/v1/buckets

Note: The getter for this property returns a list which contains *copies* of the bucket's CORS policy mappings. Mutating the list or one of its dicts has no effect unless you then re-assign the dict via the setter. E.g.:

```
>>> policies = bucket.cors
>>> policies.append({'origin': '/foo', ...})
>>> policies[1]['maxAgeSeconds'] = 3600
>>> del policies[0]
>>> bucket.cors = policies
>>> bucket.update()
```

Setter Set CORS policies for this bucket.

Getter Gets the CORS policies for this bucket.

Return type list of dictionaries

Returns A sequence of mappings describing each CORS policy.

create (*client=None*)

Creates current bucket.

If the bucket already exists, will raise `google.cloud.exceptions.Conflict`.

This implements “storage.buckets.insert”.

Parameters **client** (*Client* or `NoneType`) – Optional. The client to use. If not passed, falls back to the `client` stored on the current bucket.

default_object_acl

Create our defaultObjectACL on demand.

delete (*force=False, client=None*)

Delete this bucket.

The bucket **must** be empty in order to submit a delete request. If `force=True` is passed, this will first attempt to delete all the objects / blobs in the bucket (i.e. try to empty the bucket).

If the bucket doesn't exist, this will raise `google.cloud.exceptions.NotFound`. If the bucket is not empty (and `force=False`), will raise `google.cloud.exceptions.Conflict`.

If `force=True` and the bucket contains more than 256 objects / blobs this will cowardly refuse to delete the objects (or the bucket). This is to prevent accidental bucket deletion and to prevent extremely long runtime of this method.

Parameters

- **force** (*bool*) – If True, empties the bucket's objects then deletes it.
- **client** (*Client* or `NoneType`) – Optional. The client to use. If not passed, falls back to the `client` stored on the current bucket.

Raises `ValueError` if `force` is True and the bucket contains more than 256 objects / blobs.

delete_blob (*blob_name, client=None*)

Deletes a blob from the current bucket.

If the blob isn't found (backend 404), raises a `google.cloud.exceptions.NotFound`.

For example:

```
from google.cloud.exceptions import NotFound
client = storage.Client()
bucket = client.get_bucket('my-bucket')
blobs = list(bucket.list_blobs())
assert len(blobs) > 0
# [<Blob: my-bucket, my-file.txt>]
bucket.delete_blob('my-file.txt')
try:
    bucket.delete_blob('doesnt-exist')
except NotFound:
    pass
```

Parameters

- **blob_name** (*str*) – A blob name to delete.
- **client** (*Client* or *NoneType*) – Optional. The client to use. If not passed, falls back to the `client` stored on the current bucket.

Raises `google.cloud.exceptions.NotFound` (to suppress the exception, call `delete_blobs`, passing a no-op `on_error` callback, e.g.:

```
bucket.delete_blobs([blob], on_error=lambda blob: None)
```

delete_blobs (*blobs*, *on_error=None*, *client=None*)

Deletes a list of blobs from the current bucket.

Uses `delete_blob()` to delete each individual blob.

Parameters

- **blobs** (*list*) – A list of *Blob*-s or blob names to delete.
- **on_error** (*callable*) – (Optional) Takes single argument: `blob`. Called once for each blob raising `NotFound`; otherwise, the exception is propagated.
- **client** (*Client*) – (Optional) The client to use. If not passed, falls back to the `client` stored on the current bucket.

Raises `NotFound` (if *on_error* is not passed).

disable_logging ()

Disable access logging for this bucket.

See <https://cloud.google.com/storage/docs/access-logs#disabling>

disable_website ()

Disable the website configuration for this bucket.

This is really just a shortcut for setting the website-related attributes to `None`.

enable_logging (*bucket_name*, *object_prefix=""*)

Enable access logging for this bucket.

See <https://cloud.google.com/storage/docs/access-logs>

Parameters

- **bucket_name** (*str*) – name of bucket in which to store access logs

- **object_prefix** (*str*) – prefix for access log filenames

etag

Retrieve the ETag for the bucket.

See <https://tools.ietf.org/html/rfc2616#section-3.11> and https://cloud.google.com/storage/docs/json_api/v1/buckets

Return type `str` or `NoneType`

Returns The bucket etag or `None` if the property is not set locally.

exists (*client=None*)

Determines whether or not this bucket exists.

Parameters **client** (*Client* or `NoneType`) – Optional. The client to use. If not passed, falls back to the `client` stored on the current bucket.

Return type `bool`

Returns True if the bucket exists in Cloud Storage.

generate_upload_policy (*conditions, expiration=None, client=None*)

Create a signed upload policy for uploading objects.

This method generates and signs a policy document. You can use [policy documents](#) to allow visitors to a website to upload files to Google Cloud Storage without giving them direct write access.

For example:

```
bucket = client.bucket('my-bucket')
conditions = [
    ['starts-with', '$key', ''],
    {'acl': 'public-read'}]

policy = bucket.generate_upload_policy(conditions)

# Generate an upload form using the form fields.
policy_fields = ''.join(
    '<input type="hidden" name="{key}" value="{value}">'.format(
        key=key, value=value)
    for key, value in policy.items()
)

upload_form = (
    '<form action="http://{bucket_name}.storage.googleapis.com"'
    '  method="post" enctype="multipart/form-data">'
    '<input type="text" name="key" value="my-test-key">'
    '<input type="hidden" name="bucket" value="{bucket_name}">'
    '<input type="hidden" name="acl" value="public-read">'
    '<input name="file" type="file">'
    '<input type="submit" value="Upload">'
    '{policy_fields}'
    '</form>'.format(bucket_name=bucket.name, policy_fields=policy_
↪fields)

print(upload_form)
```

Parameters

- **expiration** (*datetime*) – Optional expiration in UTC. If not specified, the policy will expire in 1 hour.
- **conditions** (*list*) – A list of conditions as described in the [policy documents](#) documentation.
- **client** (*Client*) – Optional. The client to use. If not passed, falls back to the `client` stored on the current bucket.

Return type `dict`

Returns A dictionary of (form field name, form field value) of form fields that should be added to your HTML upload form in order to attach the signature.

get_blob (*blob_name*, *client=None*, *encryption_key=None*, ***kwargs*)

Get a blob object by name.

This will return `None` if the blob doesn't exist:

```
client = storage.Client()
bucket = client.get_bucket('my-bucket')
assert isinstance(bucket.get_blob('/path/to/blob.txt'), Blob)
# <Blob: my-bucket, /path/to/blob.txt>
assert not bucket.get_blob('/does-not-exist.txt')
# None
```

Parameters

- **blob_name** (*str*) – The name of the blob to retrieve.
- **client** (*Client* or *NoneType*) – Optional. The client to use. If not passed, falls back to the `client` stored on the current bucket.
- **encryption_key** (*bytes*) – Optional 32 byte encryption key for customer-supplied encryption. See <https://cloud.google.com/storage/docs/encryption#customer-supplied>.
- **kwargs** (*dict*) – Keyword arguments to pass to the *Blob* constructor.

Return type `google.cloud.storage.blob.Blob` or `None`

Returns The blob object if it exists, otherwise `None`.

get_iam_policy (*client=None*)

Retrieve the IAM policy for the bucket.

See https://cloud.google.com/storage/docs/json_api/v1/buckets/getIamPolicy

Parameters **client** (*Client* or *NoneType*) – Optional. The client to use. If not passed, falls back to the `client` stored on the current bucket.

Return type `google.cloud.iam.Policy`

Returns the policy instance, based on the resource returned from the `getIamPolicy` API request.

get_logging ()

Return info about access logging for this bucket.

See <https://cloud.google.com/storage/docs/access-logs#status>

Return type `dict` or `None`

Returns a dict w/ keys, `logBucket` and `logObjectPrefix` (if logging is enabled), or `None` (if not).

id

Retrieve the ID for the bucket.

See https://cloud.google.com/storage/docs/json_api/v1/buckets

Return type str or NoneType

Returns The ID of the bucket or None if the property is not set locally.

labels

Retrieve or set labels assigned to this bucket.

See https://cloud.google.com/storage/docs/json_api/v1/buckets#labels

Note: The getter for this property returns a dict which is a *copy* of the bucket's labels. Mutating that dict has no effect unless you then re-assign the dict via the setter. E.g.:

```
>>> labels = bucket.labels
>>> labels['new_key'] = 'some-label'
>>> del labels['old_key']
>>> bucket.labels = labels
>>> bucket.update()
```

Setter Set labels for this bucket.

Getter Gets the labels for this bucket.

Return type dict

Returns Name-value pairs (string->string) labelling the bucket.

lifecycle_rules

Retrieve or set lifecycle rules configured for this bucket.

See <https://cloud.google.com/storage/docs/lifecycle> and https://cloud.google.com/storage/docs/json_api/v1/buckets

Note: The getter for this property returns a list which contains *copies* of the bucket's lifecycle rules mappings. Mutating the list or one of its dicts has no effect unless you then re-assign the dict via the setter. E.g.:

```
>>> rules = bucket.lifecycle_rules
>>> rules.append({'origin': '/foo', ...})
>>> rules[1]['rule']['action']['type'] = 'Delete'
>>> del rules[0]
>>> bucket.lifecycle_rules = rules
>>> bucket.update()
```

Setter Set lifestyle rules for this bucket.

Getter Gets the lifestyle rules for this bucket.

Return type list(dict)

Returns A sequence of mappings describing each lifecycle rule.

list_blobs (*max_results=None, page_token=None, prefix=None, delimiter=None, versions=None, projection='noAcl', fields=None, client=None*)

Return an iterator used to find blobs in the bucket.

Parameters

- **max_results** (*int*) – (Optional) Maximum number of blobs to return.
- **page_token** (*str*) – (Optional) Opaque marker for the next “page” of blobs. If not passed, will return the first page of blobs.
- **prefix** (*str*) – (Optional) prefix used to filter blobs.
- **delimiter** (*str*) – (Optional) Delimiter, used with `prefix` to emulate hierarchy.
- **versions** (*bool*) – (Optional) Whether object versions should be returned as separate blobs.
- **projection** (*str*) – (Optional) If used, must be ‘full’ or ‘noAcl’. Defaults to ‘noAcl’. Specifies the set of properties to return.
- **fields** (*str*) – (Optional) Selector specifying which fields to include in a partial response. Must be a list of fields. For example to get a partial response with just the next page token and the language of each blob returned: ‘items/contentLanguage, nextPageToken’.
- **client** (*Client*) – (Optional) The client to use. If not passed, falls back to the `client` stored on the current bucket.

Return type `Iterator`

Returns Iterator of all *Blob* in this bucket matching the arguments.

location

Retrieve location configured for this bucket.

See https://cloud.google.com/storage/docs/json_api/v1/buckets and <https://cloud.google.com/storage/docs/bucket-locations>

If the property is not set locally, returns `None`.

Return type `str` or `NoneType`

make_public (*recursive=False, future=False, client=None*)

Make a bucket public.

If `recursive=True` and the bucket contains more than 256 objects / blobs this will cowardly refuse to make the objects public. This is to prevent extremely long runtime of this method.

Parameters

- **recursive** (*bool*) – If `True`, this will make all blobs inside the bucket public as well.
- **future** (*bool*) – If `True`, this will make all objects created in the future public as well.
- **client** (*Client* or `NoneType`) – Optional. The client to use. If not passed, falls back to the `client` stored on the current bucket.

metageneration

Retrieve the metageneration for the bucket.

See https://cloud.google.com/storage/docs/json_api/v1/buckets

Return type `int` or `NoneType`

Returns The metageneration of the bucket or `None` if the property is not set locally.

owner

Retrieve info about the owner of the bucket.

See https://cloud.google.com/storage/docs/json_api/v1/buckets

Return type dict or `NoneType`

Returns Mapping of owner's role/ID. If the property is not set locally, returns `None`.

patch (*client=None*)

Sends all changed properties in a PATCH request.

Updates the `_properties` with the response from the backend.

Parameters **client** (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current object.

path

The URL path to this bucket.

static path_helper (*bucket_name*)

Relative URL path for a bucket.

Parameters **bucket_name** (*str*) – The bucket name in the path.

Return type *str*

Returns The relative URL path for `bucket_name`.

project_number

Retrieve the number of the project to which the bucket is assigned.

See https://cloud.google.com/storage/docs/json_api/v1/buckets

Return type int or `NoneType`

Returns The project number that owns the bucket or `None` if the property is not set locally.

reload (*client=None*)

Reload properties from Cloud Storage.

Parameters **client** (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current object.

rename_blob (*blob, new_name, client=None*)

Rename the given blob using copy and delete operations.

Effectively, copies blob to the same bucket with a new name, then deletes the blob.

Warning: This method will first duplicate the data and then delete the old blob. This means that with very large objects renaming could be a very (temporarily) costly or a very slow operation.

Parameters

- **blob** (*google.cloud.storage.blob.Blob*) – The blob to be renamed.
- **new_name** (*str*) – The new name for this blob.
- **client** (*Client* or `NoneType`) – Optional. The client to use. If not passed, falls back to the `client` stored on the current bucket.

Return type `Blob`

Returns The newly-renamed blob.

self_link

Retrieve the URI for the bucket.

See https://cloud.google.com/storage/docs/json_api/v1/buckets

Return type str or NoneType

Returns The self link for the bucket or None if the property is not set locally.

set_iam_policy (*policy*, *client=None*)

Update the IAM policy for the bucket.

See https://cloud.google.com/storage/docs/json_api/v1/buckets/setIamPolicy

Parameters

- **policy** (*google.cloud.iam.Policy*) – policy instance used to update bucket’s IAM policy.
- **client** (*Client* or NoneType) – Optional. The client to use. If not passed, falls back to the `client` stored on the current bucket.

Return type *google.cloud.iam.Policy*

Returns the policy instance, based on the resource returned from the `setIamPolicy` API request.

storage_class

Retrieve or set the storage class for the bucket.

See <https://cloud.google.com/storage/docs/storage-classes>

Setter Set the storage class for this bucket.

Getter Gets the the storage class for this bucket.

Return type str or NoneType

Returns If set, one of “MULTI_REGIONAL”, “REGIONAL”, “NEARLINE”, “COLDLINE”, “STANDARD”, or “DURABLE_REDUCED_AVAILABILITY”, else None.

test_iam_permissions (*permissions*, *client=None*)

API call: test permissions

See https://cloud.google.com/storage/docs/json_api/v1/buckets/testIamPermissions

Parameters

- **permissions** (*list of string*) – the permissions to check
- **client** (*Client* or NoneType) – Optional. The client to use. If not passed, falls back to the `client` stored on the current bucket.

Return type list of string

Returns the permissions returned by the `testIamPermissions` API request.

time_created

Retrieve the timestamp at which the bucket was created.

See https://cloud.google.com/storage/docs/json_api/v1/buckets

Return type *datetime.datetime* or NoneType

Returns Datetime object parsed from RFC3339 valid timestamp, or None if the property is not set locally.

update (*client=None*)

Sends all properties in a PUT request.

Updates the `_properties` with the response from the backend.

Parameters `client` (*Client* or `NoneType`) – the client to use. If not passed, falls back to the `client` stored on the current object.

versioning_enabled

Is versioning enabled for this bucket?

See <https://cloud.google.com/storage/docs/object-versioning> for details.

Setter Update whether versioning is enabled for this bucket.

Getter Query whether versioning is enabled for this bucket.

Return type `bool`

Returns True if enabled, else False.

18.3 ACL

Manipulate access control lists that Cloud Storage provides.

`google.cloud.storage.bucket.Bucket` has a getting method that creates an ACL object under the hood, and you can interact with that using `google.cloud.storage.bucket.Bucket.acl()`:

```
client = storage.Client()
bucket = client.get_bucket(bucket_name)
acl = bucket.acl
```

Adding and removing permissions can be done with the following methods (in increasing order of granularity):

- `ACL.all()` corresponds to access for all users.
- `ACL.all_authenticated()` corresponds to access for all users that are signed into a Google account.
- `ACL.domain()` corresponds to access on a per Google Apps domain (ie, `example.com`).
- `ACL.group()` corresponds to access on a per group basis (either by ID or e-mail address).
- `ACL.user()` corresponds to access on a per user basis (either by ID or e-mail address).

And you are able to grant and revoke the following roles:

- **Reading:** `_ACLEntity.grant_read()` and `_ACLEntity.revoke_read()`
- **Writing:** `_ACLEntity.grant_write()` and `_ACLEntity.revoke_write()`
- **Owning:** `_ACLEntity.grant_owner()` and `_ACLEntity.revoke_owner()`

You can use any of these like any other factory method (these happen to be `_ACLEntity` factories):

```
acl.user('me@example.org').grant_read()
acl.all_authenticated().grant_write()
```

After that, you can save any changes you make with the `google.cloud.storage.acl.ACL.save()` method:

```
acl.save()
```

You can alternatively save any existing `google.cloud.storage.acl.ACL` object (whether it was created by a factory method or not) from a `google.cloud.storage.bucket.Bucket`:

```
bucket.acl.save(acl=acl)
```

To get the list of entity and role for each unique pair, the `ACL` class is iterable:

```
print(list(acl))
# [{'role': 'OWNER', 'entity': 'allUsers'}, ...]
```

This list of tuples can be used as the `entity` and `role` fields when sending metadata for ACLs to the API.

class `google.cloud.storage.acl.ACL`

Bases: `object`

Container class representing a list of access controls.

PREDEFINED_JSON_ACLS = `frozenset(['publicRead', 'bucketOwnerFullControl', 'bucketOwner'])`

See <https://cloud.google.com/storage/docs/access-control/lists#predefined-acl>

add_entity(*entity*)

Add an entity to the ACL.

Parameters *entity* (`_ACLEntity`) – The entity to add to this ACL.

all()

Factory method for an Entity representing all users.

Return type `_ACLEntity`

Returns An entity representing all users.

all_authenticated()

Factory method for an Entity representing all authenticated users.

Return type `_ACLEntity`

Returns An entity representing all authenticated users.

clear(*client=None*)

Remove all ACL entries.

Note that this won't actually remove *ALL* the rules, but it will remove all the non-default rules. In short, you'll still have access to a bucket that you created even after you clear ACL rules with this method.

Parameters *client* (`Client` or `NoneType`) – Optional. The client to use. If not passed, falls back to the `client` stored on the ACL's parent.

client

Abstract getter for the object client.

domain(*domain*)

Factory method for a domain Entity.

Parameters *domain* (`str`) – The domain for this entity.

Return type `_ACLEntity`

Returns An entity corresponding to this domain.

entity(*entity_type*, *identifier=None*)

Factory method for creating an Entity.

If an entity with the same type and identifier already exists, this will return a reference to that entity. If not, it will create a new one and add it to the list of known entities for this ACL.

Parameters

- **entity_type** (*str*) – The type of entity to create (ie, `user`, `group`, etc)
- **identifier** (*str*) – The ID of the entity (if applicable). This can be either an ID or an e-mail address.

Return type `_ACLEntity`

Returns A new Entity or a reference to an existing identical entity.

entity_from_dict (*entity_dict*)

Build an `_ACLEntity` object from a dictionary of data.

An entity is a mutable object that represents a list of roles belonging to either a user or group or the special types for all users and all authenticated users.

Parameters **entity_dict** (*dict*) – Dictionary full of data from an ACL lookup.

Return type `_ACLEntity`

Returns An Entity constructed from the dictionary.

get_entities ()

Get a list of all Entity objects.

Return type list of `_ACLEntity` objects

Returns A list of all Entity objects.

get_entity (*entity*, *default=None*)

Gets an entity object from the ACL.

Parameters

- **entity** (`_ACLEntity` or string) – The entity to get lookup in the ACL.
- **default** (*anything*) – This value will be returned if the entity doesn't exist.

Return type `_ACLEntity`

Returns The corresponding entity or the value provided to default.

group (*identifier*)

Factory method for a group Entity.

Parameters **identifier** (*str*) – An id or e-mail for this particular group.

Return type `_ACLEntity`

Returns An Entity corresponding to this group.

has_entity (*entity*)

Returns whether or not this ACL has any entries for an entity.

Parameters **entity** (`_ACLEntity`) – The entity to check for existence in this ACL.

Return type `bool`

Returns True if the entity exists in the ACL.

reload (*client=None*)

Reload the ACL data from Cloud Storage.

Parameters **client** (`Client` or `NoneType`) – Optional. The client to use. If not passed, falls back to the `client` stored on the ACL's parent.

reset ()

Remove all entities from the ACL, and clear the loaded flag.

save (*acl=None, client=None*)

Save this ACL for the current bucket.

Parameters

- **acl** (*google.cloud.storage.acl.ACL*, or a compatible list.) – The ACL object to save. If left blank, this will save current entries.
- **client** (*Client* or *NoneType*) – Optional. The client to use. If not passed, falls back to the `client` stored on the ACL's parent.

save_predefined (*predefined, client=None*)

Save this ACL for the current bucket using a predefined ACL.

Parameters

- **predefined** (*str*) – An identifier for a predefined ACL. Must be one of the keys in *PREDEFINED_JSON_ACLS* or *PREDEFINED_XML_ACLS* (which will be aliased to the corresponding JSON name). If passed, *acl* must be *None*.
- **client** (*Client* or *NoneType*) – Optional. The client to use. If not passed, falls back to the `client` stored on the ACL's parent.

user (*identifier*)

Factory method for a user Entity.

Parameters **identifier** (*str*) – An id or e-mail for this particular user.

Return type *_ACLEntity*

Returns An Entity corresponding to this user.

class *google.cloud.storage.acl.BucketACL* (*bucket*)

Bases: *google.cloud.storage.acl.ACL*

An ACL specifically for a bucket.

Parameters **bucket** (*google.cloud.storage.bucket.Bucket*) – The bucket to which this ACL relates.

client

The client bound to this ACL's bucket.

reload_path

Compute the path for GET API requests for this ACL.

save_path

Compute the path for PATCH API requests for this ACL.

class *google.cloud.storage.acl.DefaultObjectACL* (*bucket*)

Bases: *google.cloud.storage.acl.BucketACL*

A class representing the default object ACL for a bucket.

class *google.cloud.storage.acl.ObjectACL* (*blob*)

Bases: *google.cloud.storage.acl.ACL*

An ACL specifically for a Cloud Storage object / blob.

Parameters **blob** (*google.cloud.storage.blob.Blob*) – The blob that this ACL corresponds to.

client

The client bound to this ACL's blob.

reload_path

Compute the path for GET API requests for this ACL.

save_path

Compute the path for PATCH API requests for this ACL.

18.4 Batches

Batch updates / deletes of storage buckets / blobs.

See https://cloud.google.com/storage/docs/json_api/v1/how-tos/batch

class `google.cloud.storage.batch.Batch` (*client*)

Bases: `google.cloud.storage._http.Connection`

Proxy an underlying connection, batching up change operations.

Parameters *client* (`google.cloud.storage.client.Client`) – The client to use for making connections.

current()

Return the topmost batch, or None.

finish()

Submit a single *multipart/mixed* request with deferred requests.

Return type list of tuples

Returns one (headers, payload) tuple per deferred request.

class `google.cloud.storage.batch.MIMEApplicationHTTP` (*method, uri, headers, body*)

Bases: `email.mime.application.MIMEApplication`

MIME type for application/http.

Constructs payload from headers and body

Parameters

- **method** (*str*) – HTTP method
- **uri** (*str*) – URI for HTTP request
- **headers** (*dict*) – HTTP headers
- **body** (*str*) – (Optional) HTTP payload

Client for interacting with the Google Cloud Storage API.

class `google.cloud.storage.client.Client` (*project=None, credentials=None, _http=None*)

Bases: `google.cloud.client.ClientWithProject`

Client to bundle configuration needed for API requests.

Parameters

- **project** (*str*) – the project which the client acts on behalf of. Will be passed when creating a topic. If not passed, falls back to the default inferred from the environment.
- **credentials** (`Credentials`) – (Optional) The OAuth2 Credentials to use for this client. If not passed (and if no `_http` object is passed), falls back to the default inferred from the environment.

- **_http** (*Session*) – (Optional) HTTP object to make requests. Can be any object that defines `request()` with the same interface as `requests.Session.request()`. If not passed, an `_http` object is created that is bound to the `credentials` for the current object. This parameter should be considered private, and could change in the future.

SCOPE = ('https://www.googleapis.com/auth/devstorage.full_control', 'https://www.googleapis.com/auth/devstorage.read_only')

The scopes required for authenticating as a Cloud Storage consumer.

batch()

Factory constructor for batch object.

Note: This will not make an HTTP request; it simply instantiates a batch object owned by this client.

Return type `google.cloud.storage.batch.Batch`

Returns The batch object created.

bucket (*bucket_name*)

Factory constructor for bucket object.

Note: This will not make an HTTP request; it simply instantiates a bucket object owned by this client.

Parameters **bucket_name** (*str*) – The name of the bucket to be instantiated.

Return type `google.cloud.storage.bucket.Bucket`

Returns The bucket object created.

create_bucket (*bucket_name*)

Create a new bucket.

For example:

```
bucket = client.create_bucket('my-bucket')
assert isinstance(bucket, Bucket)
# <Bucket: my-bucket>
```

This implements “storage.buckets.insert”.

If the bucket already exists, will raise `google.cloud.exceptions.Conflict`.

Parameters **bucket_name** (*str*) – The bucket name to create.

Return type `google.cloud.storage.bucket.Bucket`

Returns The newly created bucket.

current_batch

Currently-active batch.

Return type `google.cloud.storage.batch.Batch` or `NoneType` (if no batch is active).

Returns The batch at the top of the batch stack.

get_bucket (*bucket_name*)

Get a bucket by name.

If the bucket isn’t found, this will raise a `google.cloud.storage.exceptions.NotFound`.

For example:

```
try:
    bucket = client.get_bucket('my-bucket')
except google.cloud.exceptions.NotFound:
    print('Sorry, that bucket does not exist!')
```

This implements “storage.buckets.get”.

Parameters `bucket_name` (*str*) – The name of the bucket to get.

Return type `google.cloud.storage.bucket.Bucket`

Returns The bucket matching the name provided.

Raises `google.cloud.exceptions.NotFound`

list_buckets (*max_results=None*, *page_token=None*, *prefix=None*, *projection='noAcl'*, *fields=None*)

Get all buckets in the project associated to the client.

This will not populate the list of blobs available in each bucket.

```
for bucket in client.list_buckets():
    print(bucket)
```

This implements “storage.buckets.list”.

Parameters

- **max_results** (*int*) – Optional. Maximum number of buckets to return.
- **page_token** (*str*) – Optional. Opaque marker for the next “page” of buckets. If not passed, will return the first page of buckets.
- **prefix** (*str*) – Optional. Filter results to buckets whose names begin with this prefix.
- **projection** (*str*) – (Optional) Specifies the set of properties to return. If used, must be ‘full’ or ‘noAcl’. Defaults to ‘noAcl’.
- **fields** (*str*) – (Optional) Selector specifying which fields to include in a partial response. Must be a list of fields. For example to get a partial response with just the next page token and the language of each bucket returned: ‘items/id,nextPageToken’

Return type `Iterator`

Returns Iterator of all `Bucket` belonging to this project.

lookup_bucket (*bucket_name*)

Get a bucket by name, returning `None` if not found.

You can use this if you would rather check for a `None` value than catching an exception:

```
bucket = client.lookup_bucket('doesn't-exist')
assert not bucket
# None
bucket = client.lookup_bucket('my-bucket')
assert isinstance(bucket, Bucket)
# <Bucket: my-bucket>
```

Parameters `bucket_name` (*str*) – The name of the bucket to get.

Return type `google.cloud.storage.bucket.Bucket`

Returns The bucket matching the name provided or None if not found.

19.1 Translation Client

Client for interacting with the Google Cloud Translation API.

`google.cloud.translate_v2.client.BASE = 'base'`
Base translation model.

class `google.cloud.translate_v2.client.Client` (*target_language='en', credentials=None, _http=None*)

Bases: `google.cloud.client.Client`

Client to bundle configuration needed for API requests.

Parameters

- **target_language** (*str*) – (Optional) The target language used for translations and language names. (Defaults to `ENGLISH_ISO_639`.)
- **credentials** (*Credentials*) – (Optional) The OAuth2 Credentials to use for this client. If not passed (and if no `_http` object is passed), falls back to the default inferred from the environment.
- **_http** (*Session*) – (Optional) HTTP object to make requests. Can be any object that defines `request()` with the same interface as `requests.Session.request()`. If not passed, an `_http` object is created that is bound to the `credentials` for the current object. This parameter should be considered private, and could change in the future.

SCOPE = ('https://www.googleapis.com/auth/cloud-platform',)

The scopes required for authenticating.

detect_language (*values*)

Detect the language of a string or list of strings.

See <https://cloud.google.com/translate/docs/detecting-language>

Parameters values (*str or list*) – String or list of strings that will have language detected.

Return type `str` or `list`

Returns

A list of dictionaries for each queried value. Each dictionary typically contains three keys

- `confidence`: The confidence in language detection, a float between 0 and 1.
- `input`: The corresponding input value.
- `language`: The detected language (as an ISO 639-1 language code).

though the key `confidence` may not always be present.

If only a single value is passed, then only a single dictionary will be returned.

Raises `ValueError` if the number of detections is not equal to the number of values.

`ValueError` if a value produces a list of detections with 0 or multiple results in it.

get_languages (*target_language=None*)

Get list of supported languages for translation.

Response

See <https://cloud.google.com/translate/docs/discovering-supported-languages>

Parameters `target_language` (*str*) – (Optional) The language used to localize returned language names. Defaults to the target language on the current client.

Return type `list`

Returns List of dictionaries. Each dictionary contains a supported ISO 639-1 language code (using the dictionary key `language`). If `target_language` is passed, each dictionary will also contain the name of each supported language (localized to the target language).

translate (*values*, *target_language=None*, *format=None*, *source_language=None*, *customization_ids=()*, *model=None*)

Translate a string or list of strings.

See <https://cloud.google.com/translate/docs/translating-text>

Parameters

- **values** (*str* or *list*) – String or list of strings to translate.
- **target_language** (*str*) – The language to translate results into. This is required by the API and defaults to the target language of the current instance.
- **format** (*str*) – (Optional) One of `text` or `html`, to specify if the input text is plain text or HTML.
- **source_language** (*str*) – (Optional) The language of the text to be translated.
- **customization_ids** (*str* or *list*) – (Optional) ID or list of customization IDs for translation. Sets the `cid` parameter in the query.
- **model** (*str*) – (Optional) The model used to translate the text, such as `'base'` or `'nmt'`.

Return type `str` or `list`

Returns

A list of dictionaries for each queried value. Each dictionary typically contains three keys (though not all will be present in all cases)

- `detectedSourceLanguage`: The detected language (as an ISO 639-1 language code) of the text.

- `translatedText`: The translation of the text into the target language.
- `input`: The corresponding input value.
- `model`: The model used to translate the text.

If only a single value is passed, then only a single dictionary will be returned.

Raises `ValueError` if the number of values and translations differ.

```
google.cloud.translate_v2.client.ENGLISH_ISO_639 = 'en'
ISO 639-1 language code for English.
```

```
google.cloud.translate_v2.client.NMT = 'nmt'
Neural Machine Translation model.
```

With [Google Cloud Translation](#), you can dynamically translate text between thousands of language pairs. The Google Cloud Translation API lets websites and programs integrate with Google Cloud Translation programmatically. Google Cloud Translation is available as a paid service. See the [Pricing](#) and [FAQ](#) pages for details.

19.2 Authentication / Configuration

- Use `Client` objects to configure your applications.
- `Client` objects hold a connection to the Cloud Translation service.

19.3 Methods

To create a client:

```
>>> from google.cloud import translate
>>> client = translate.Client()
```

By default, the client targets English when doing detections and translations, but a non-default value can be used as well:

```
>>> from google.cloud import translate
>>> client = translate.Client(target_language='es')
```

The Google Cloud Translation API has three supported methods, and they map to three methods on a client: `get_languages()`, `detect_language()` and `translate()`.

To get a list of languages supported by the Google Cloud Translation API

```
>>> from google.cloud import translate
>>> client = translate.Client()
>>> client.get_languages()
[
  {
    'language': 'af',
    'name': 'Afrikaans',
  },
  ...
]
```

To detect the language that some given text is written in:

```
>>> from google.cloud import translate
>>> client = translate.Client()
>>> client.detect_language(['Me llamo', 'I am'])
[
  {
    'confidence': 0.25830904,
    'input': 'Me llamo',
    'language': 'es',
  }, {
    'confidence': 0.17112699,
    'input': 'I am',
    'language': 'en',
  },
]
```

The `confidence` value is an optional floating point value between 0 and 1. The closer this value is to 1, the higher the confidence level for the language detection. This member is not always available.

To translate text:

```
>>> from google.cloud import translate
>>> client = translate.Client()
>>> client.translate('koszula')
{
  'translatedText': 'shirt',
  'detectedSourceLanguage': 'pl',
  'input': 'koszula',
}
```

or to use a non-default target language:

```
>>> from google.cloud import translate
>>> client = translate.Client()
>>> client.translate(['Me llamo Jeff', 'My name is Jeff'],
...                  target_language='de')
[
  {
    'translatedText': 'Mein Name ist Jeff',
    'detectedSourceLanguage': 'es',
    'input': 'Me llamo Jeff',
  }, {
    'translatedText': 'Mein Name ist Jeff',
    'detectedSourceLanguage': 'en',
    'input': 'My name is Jeff',
  },
]
```

The Google Cloud [Vision \(Vision API docs\)](#) API enables developers to understand the content of an image by encapsulating powerful machine learning models in an easy to use REST API. It quickly classifies images into thousands of categories (e.g., “sailboat”, “lion”, “Eiffel Tower”), detects individual objects and faces within images, and finds and reads printed words contained within images. You can build metadata on your image catalog, moderate offensive content, or enable new marketing scenarios through image sentiment analysis. Analyze images uploaded in the request or integrate with your image storage on Google Cloud Storage.

20.1 Authentication and Configuration

- For an overview of authentication in `google-cloud-python`, see [Authentication](#).
- In addition to any authentication configuration, you should also set the `GOOGLE_CLOUD_PROJECT` environment variable for the project you’d like to interact with. If the `GOOGLE_CLOUD_PROJECT` environment variable is not present, the project ID from JSON file credentials is used.

If you are using Google App Engine or Google Compute Engine this will be detected automatically.

- After configuring your environment, create a `ImageAnnotatorClient`.

```
>>> from google.cloud import vision
>>> client = vision.ImageAnnotatorClient()
```

or pass in credentials explicitly.

```
>>> from google.cloud import vision
>>> client = vision.ImageAnnotatorClient(
...     credentials=creds,
... )
```

20.2 Annotate an Image

You can call the `annotate_image()` method directly:

```
>>> from google.cloud import vision
>>> client = vision.ImageAnnotatorClient()
>>> response = client.annotate_image({
...     'image': {'source': {'image_uri': 'gs://my-test-bucket/image.jpg'}},
...     'features': [{'type': vision.enums.Feature.Type.FACE_DETECTION}],
... })
>>> len(response.annotations)
2
>>> for face in response.annotations[0].faces:
...     print(face.joy)
Likelihood.VERY_LIKELY
Likelihood.VERY_LIKELY
Likelihood.VERY_LIKELY
>>> for logo in response.annotations[0].logos:
...     print(logo.description)
'google'
'github'
```

20.3 Single-feature Shortcuts

If you are only requesting a single feature, you may find it easier to ask for it using our direct methods:

```
>>> from google.cloud import vision
>>> client = vision.ImageAnnotatorClient()
>>> response = client.face_detection({
...     'source': {'image_uri': 'gs://my-test-bucket/image.jpg'},
... })
>>> len(response.annotations)
1
>>> for face in response.annotations[0].faces:
...     print(face.joy)
Likelihood.VERY_LIKELY
Likelihood.VERY_LIKELY
Likelihood.VERY_LIKELY
```

20.4 No results found

If no results for the detection performed can be extracted from the image, then an empty list is returned. This behavior is similar with all detection types.

Example with `logo_detection()`:

```
>>> from google.cloud import vision
>>> client = vision.ImageAnnotatorClient()
>>> with open('./image.jpg', 'rb') as image_file:
...     content = image_file.read()
>>> response = client.logo_detection({
...     'content': content,
... })
```

```
>>> len(response.annotations)
0
```

20.5 API Reference

20.5.1 Vision Client API

```
class google.cloud.vision_v1.ImageAnnotatorClient (channel=None, credentials=None, ssl_credentials=None, scopes=None, client_config=None, lib_name=None, lib_version="", metrics_headers=())
```

Service that performs Google Cloud Vision API detection tasks over client images, such as face, landmark, logo, label, and text detection. The ImageAnnotator service returns detected entities from the images.

Constructor.

Parameters

- **channel** (*Channel*) – A Channel instance through which to make calls.
- **credentials** (*Credentials*) – The authorization credentials to attach to requests. These credentials identify this application to the service.
- **ssl_credentials** (*ChannelCredentials*) – A ChannelCredentials instance for use with an SSL-enabled channel.
- **scopes** (*Sequence[str]*) – A list of OAuth2 scopes to attach to requests.
- **client_config** (*dict*) – A dictionary for call options for each method. See `google.gax.construct_settings()` for the structure of this data. Falls back to the default config if not specified or the specified config is missing data points.
- **lib_name** (*str*) – The API library software used for calling the service. (Unless you are writing an API client itself, leave this as default.)
- **lib_version** (*str*) – The API library software version used for calling the service. (Unless you are writing an API client itself, leave this as default.)
- **metrics_headers** (*dict*) – A dictionary of values for tracking client library metrics. Ultimately serializes to a string (e.g. 'foo/1.2.3 bar/3.14.1'). This argument should be considered private.

Returns: ImageAnnotatorClient

```
annotate_image (request, options=None)
```

Run image detection and annotation for an image.

Example

```
>>> from google.cloud.vision_v1 import ImageAnnotatorClient
>>> client = ImageAnnotatorClient()
>>> request = {
...     'image': {
...         'source': {'image_uri': 'https://foo.com/image.jpg'},
...     },
... }
```

```
... }
>>> response = client.annotate_image(request)
```

Parameters

- **request** (*AnnotateImageRequest*) –
- **options** (*google.gax.CallOptions*) – Overrides the default settings for this call, e.g, timeout, retries, etc.

Returns *AnnotateImageResponse* The API response.

batch_annotate_images (*requests, options=None*)

Run image detection and annotation for a batch of images.

Example

```
>>> from google.cloud import vision_v1
>>>
>>> client = vision_v1.ImageAnnotatorClient()
>>>
>>> requests = []
>>>
>>> response = client.batch_annotate_images(requests)
```

Parameters

- **requests** (*list[Union[dict, AnnotateImageRequest]]*) – Individual image annotation requests for this batch. If a dict is provided, it must be of the same form as the protobuf message *AnnotateImageRequest*
- **options** (*CallOptions*) – Overrides the default settings for this call, e.g, timeout, retries etc.

Returns A *BatchAnnotateImagesResponse* instance.

Raises

- *google.gax.errors.GaxError* if the RPC is aborted.
- *ValueError* if the parameters are invalid.

crop_hints (*image, options=None, **kwargs*)

Return crop hints information.

Parameters

- **image** (*Image*) – The image to analyze.
- **options** (*google.gax.CallOptions*) – Overrides the default settings for this call, e.g, timeout, retries, etc.
- **kwargs** (*dict*) – Additional properties to be set on the *AnnotateImageRequest*.

Returns The API response.

Return type *AnnotateImageResponse*

document_text_detection (*image, options=None, **kwargs*)

Perform document text detection.

Parameters

- **image** (*Image*) – The image to analyze.
- **options** (*google.gax.CallOptions*) – Overrides the default settings for this call, e.g, timeout, retries, etc.
- **kwargs** (*dict*) – Additional properties to be set on the *AnnotateImageRequest*.

Returns The API response.

Return type *AnnotateImageResponse*

```
enums = <module 'google.cloud.vision_v1.gapic.enums' from '/home/docs/checkouts/readth
```

```
face_detection (image, options=None, **kwargs)
```

Perform face detection.

Parameters

- **image** (*Image*) – The image to analyze.
- **options** (*google.gax.CallOptions*) – Overrides the default settings for this call, e.g, timeout, retries, etc.
- **kwargs** (*dict*) – Additional properties to be set on the *AnnotateImageRequest*.

Returns The API response.

Return type *AnnotateImageResponse*

```
image_properties (image, options=None, **kwargs)
```

Return image properties information.

Parameters

- **image** (*Image*) – The image to analyze.
- **options** (*google.gax.CallOptions*) – Overrides the default settings for this call, e.g, timeout, retries, etc.
- **kwargs** (*dict*) – Additional properties to be set on the *AnnotateImageRequest*.

Returns The API response.

Return type *AnnotateImageResponse*

```
label_detection (image, options=None, **kwargs)
```

Perform label detection.

Parameters

- **image** (*Image*) – The image to analyze.
- **options** (*google.gax.CallOptions*) – Overrides the default settings for this call, e.g, timeout, retries, etc.
- **kwargs** (*dict*) – Additional properties to be set on the *AnnotateImageRequest*.

Returns The API response.

Return type *AnnotateImageResponse*

```
landmark_detection (image, options=None, **kwargs)
```

Perform landmark detection.

Parameters

- **image** (*Image*) – The image to analyze.

- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries, etc.
- **kwargs** (`dict`) – Additional properties to be set on the `AnnotateImageRequest`.

Returns The API response.

Return type `AnnotateImageResponse`

logo_detection (`image`, `options=None`, `**kwargs`)

Perform logo detection.

Parameters

- **image** (`Image`) – The image to analyze.
- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries, etc.
- **kwargs** (`dict`) – Additional properties to be set on the `AnnotateImageRequest`.

Returns The API response.

Return type `AnnotateImageResponse`

safe_search_detection (`image`, `options=None`, `**kwargs`)

Perform safe search detection.

Parameters

- **image** (`Image`) – The image to analyze.
- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries, etc.
- **kwargs** (`dict`) – Additional properties to be set on the `AnnotateImageRequest`.

Returns The API response.

Return type `AnnotateImageResponse`

text_detection (`image`, `options=None`, `**kwargs`)

Perform text detection.

Parameters

- **image** (`Image`) – The image to analyze.
- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries, etc.
- **kwargs** (`dict`) – Additional properties to be set on the `AnnotateImageRequest`.

Returns The API response.

Return type `AnnotateImageResponse`

web_detection (`image`, `options=None`, `**kwargs`)

Perform web detection.

Parameters

- **image** (`Image`) – The image to analyze.
- **options** (`google.gax.CallOptions`) – Overrides the default settings for this call, e.g, timeout, retries, etc.
- **kwargs** (`dict`) – Additional properties to be set on the `AnnotateImageRequest`.

Returns The API response.

Return type *AnnotateImageResponse*

20.5.2 Vision Client Types

class google.cloud.vision_v1.types.**AnnotateImageRequest**

Request for performing Google Cloud Vision API tasks over a user-provided image, with user-requested features.

image

The image to be processed.

features

Requested features.

image_context

Additional context that may accompany the image.

class google.cloud.vision_v1.types.**AnnotateImageResponse**

Response to an image annotation request.

face_annotations

If present, face detection has completed successfully.

landmark_annotations

If present, landmark detection has completed successfully.

logo_annotations

If present, logo detection has completed successfully.

label_annotations

If present, label detection has completed successfully.

text_annotations

If present, text (OCR) detection or document (OCR) text detection has completed successfully.

full_text_annotation

If present, text (OCR) detection or document (OCR) text detection has completed successfully. This annotation provides the structural hierarchy for the OCR detected text.

safe_search_annotation

If present, safe-search annotation has completed successfully.

image_properties_annotation

If present, image properties were extracted successfully.

crop_hints_annotation

If present, crop hints have completed successfully.

web_detection

If present, web detection has completed successfully.

error

If set, represents the error message for the operation. Note that filled-in image annotations are guaranteed to be correct, even when `error` is set.

class google.cloud.vision_v1.types.**BatchAnnotateImagesRequest**

Multiple image annotation requests are batched into a single service call.

requests

Individual image annotation requests for this batch.

class google.cloud.vision_v1.types.**BatchAnnotateImagesResponse**

Response to a batch image annotation request.

responses

Individual responses to image annotation requests within the batch.

class google.cloud.vision_v1.types.**Block**

Logical element on the page.

property

Additional information detected for the block.

bounding_box

The bounding box for the block. The vertices are in the order of top-left, top-right, bottom-right, bottom-left. When a rotation of the bounding box is detected the rotation is represented as around the top-left corner as defined when the text is read in the 'natural' orientation. For example: * when the text is horizontal it might look like: 0—1 || 3 —2 * when it's rotated 180 degrees around the top-left corner it becomes: 2—3 || 1 —0 and the vertice order will still be (0, 1, 2, 3).

paragraphs

List of paragraphs in this block (if this blocks is of type text).

block_type

Detected block type (text, image etc) for this block.

class google.cloud.vision_v1.types.**BoundingPoly**

A bounding polygon for the detected image annotation.

vertices

The bounding polygon vertices.

class google.cloud.vision_v1.types.**ColorInfo**

Color information consists of RGB channels, score, and the fraction of the image that the color occupies in the image.

color

RGB components of the color.

score

Image-specific score for this color. Value in range [0, 1].

pixel_fraction

The fraction of pixels the color occupies in the image. Value in range [0, 1].

class google.cloud.vision_v1.types.**CropHint**

Single crop hint that is used to generate a new crop when serving an image.

bounding_poly

The bounding polygon for the crop region. The coordinates of the bounding box are in the original image's scale, as returned in ImageParams.

confidence

Confidence of this being a salient region. Range [0, 1].

importance_fraction

Fraction of importance of this salient region with respect to the original image.

class google.cloud.vision_v1.types.**CropHintsAnnotation**

Set of crop hints that are used to generate new crops when serving images.

class google.cloud.vision_v1.types.**CropHintsParams**

Parameters for crop hints annotation request.

aspect_ratios

Aspect ratios in floats, representing the ratio of the width to the height of the image. For example, if the desired aspect ratio is 4/3, the corresponding float value should be 1.33333. If not specified, the best possible crop is returned. The number of provided aspect ratios is limited to a maximum of 16; any aspect ratios provided after the 16th are ignored.

class `google.cloud.vision_v1.types.DominantColorsAnnotation`

Set of dominant colors and their corresponding scores.

colors

RGB color values with their score and pixel fraction.

class `google.cloud.vision_v1.types.EntityAnnotation`

Set of detected entity features.

mid

Opaque entity ID. Some IDs may be available in [Google Knowledge Graph Search API](#).

locale

The language code for the locale in which the entity textual description is expressed.

description

Entity textual description, expressed in its `locale` language.

score

Overall score of the result. Range [0, 1].

confidence

The accuracy of the entity detection in an image. For example, for an image in which the “Eiffel Tower” entity is detected, this field represents the confidence that there is a tower in the query image. Range [0, 1].

topicality

The relevancy of the ICA (Image Content Annotation) label to the image. For example, the relevancy of “tower” is likely higher to an image containing the detected “Eiffel Tower” than to an image containing a detected distant towering building, even though the confidence that there is a tower in each image may be the same. Range [0, 1].

bounding_poly

Image region to which this entity belongs. Currently not produced for `LABEL_DETECTION` features. For `TEXT_DETECTION` (OCR), `boundingPolys` are produced for the entire text detected in an image region, followed by `boundingPolys` for each word within the detected text.

locations

The location information for the detected entity. Multiple `LocationInfo` elements can be present because one location may indicate the location of the scene in the image, and another location may indicate the location of the place where the image was taken. Location information is usually present for landmarks.

properties

Some entities may have optional user-supplied `Property` (name/value) fields, such a score or string that qualifies the entity.

class `google.cloud.vision_v1.types.FaceAnnotation`

A face annotation object contains the results of face detection.

type

Face landmark type.

position

Face landmark position.

bounding_poly

The bounding polygon around the face. The coordinates of the bounding box are in the original image's scale, as returned in `ImageParams`. The bounding box is computed to “frame” the face in accordance with human expectations. It is based on the landmarker results. Note that one or more x and/or y coordinates may not be generated in the `BoundingBoxPoly` (the polygon will be unbounded) if only a partial face appears in the image to be annotated.

fd_bounding_poly

The `fd_bounding_poly` bounding polygon is tighter than the `boundingPoly`, and encloses only the skin part of the face. Typically, it is used to eliminate the face from any image analysis that detects the “amount of skin” visible in an image. It is not based on the landmarker results, only on the initial face detection, hence the fd (face detection) prefix.

landmarks

Detected face landmarks.

roll_angle

Roll angle, which indicates the amount of clockwise/anti- clockwise rotation of the face relative to the image vertical about the axis perpendicular to the face. Range [-180,180].

pan_angle

Yaw angle, which indicates the leftward/rightward angle that the face is pointing relative to the vertical plane perpendicular to the image. Range [-180,180].

tilt_angle

Pitch angle, which indicates the upwards/downwards angle that the face is pointing relative to the image's horizontal plane. Range [-180,180].

detection_confidence

Detection confidence. Range [0, 1].

landmarking_confidence

Face landmarking confidence. Range [0, 1].

joy_likelihood

Joy likelihood.

sorrow_likelihood

Sorrow likelihood.

anger_likelihood

Anger likelihood.

surprise_likelihood

Surprise likelihood.

under_exposed_likelihood

Under-exposed likelihood.

blurred_likelihood

Blurred likelihood.

headwear_likelihood

Headwear likelihood.

class Landmark

A face-specific landmark (for example, a face feature). Landmark positions may fall outside the bounds of the image if the face is near one or more edges of the image. Therefore it is NOT guaranteed that $0 \leq x < \text{width}$ or $0 \leq y < \text{height}$.

class google.cloud.vision_v1.types.Feature

Users describe the type of Google Cloud Vision API tasks to perform over images by using *Features*. Each

Feature indicates a type of image detection task to perform. Features encode the Cloud Vision API vertical to operate on and the number of top-scoring results to return.

type

The feature type.

max_results

Maximum number of results of this type.

class google.cloud.vision_v1.types.**Image**

Client image to perform Google Cloud Vision API tasks over.

content

Image content, represented as a stream of bytes. Note: as with all `bytes` fields, `protobuffers` use a pure binary representation, whereas `JSON` representations use `base64`.

source

Google Cloud Storage image location. If both `content` and `source` are provided for an image, `content` takes precedence and is used to perform the image annotation request.

class google.cloud.vision_v1.types.**ImageContext**

Image context and/or feature-specific parameters.

lat_long_rect

lat/long rectangle that specifies the location of the image.

language_hints

List of languages to use for `TEXT_DETECTION`. In most cases, an empty value yields the best results since it enables automatic language detection. For languages based on the Latin alphabet, setting `language_hints` is not needed. In rare cases, when the language of the text in the image is known, setting a hint will help get better results (although it will be a significant hindrance if the hint is wrong). Text detection returns an error if one or more of the specified languages is not one of the [supported languages](#).

crop_hints_params

Parameters for crop hints annotation request.

class google.cloud.vision_v1.types.**ImageProperties**

Stores image properties, such as dominant colors.

dominant_colors

If present, dominant colors completed successfully.

class google.cloud.vision_v1.types.**ImageSource**

External image source (Google Cloud Storage image location).

gcs_image_uri

NOTE: For new code `image_uri` below is preferred. Google Cloud Storage image URI, which must be in the following form: `gs://bucket_name/object_name` (for details, see [Google Cloud Storage Request URIs](#)). NOTE: Cloud Storage object versioning is not supported.

image_uri

Image URI which supports: 1) Google Cloud Storage image URI, which must be in the following form: `gs://bucket_name/object_name` (for details, see [Google Cloud Storage Request URIs](#)). NOTE: Cloud Storage object versioning is not supported. 2) Publicly accessible image `HTTP/HTTPS` URL. This is preferred over the legacy `gcs_image_uri` above. When both `gcs_image_uri` and `image_uri` are specified, `image_uri` takes precedence.

class google.cloud.vision_v1.types.**LatLongRect**

Rectangle determined by min and max `LatLng` pairs.

min_lat_lng

Min lat/long pair.

max_lat_lng

Max lat/long pair.

class google.cloud.vision_v1.types.LocationInfo

Detected entity location information.

lat_lng

lat/long location coordinates.

class google.cloud.vision_v1.types.Page

Detected page from OCR.

property

Additional information detected on the page.

width

Page width in pixels.

height

Page height in pixels.

blocks

List of blocks of text, images etc on this page.

class google.cloud.vision_v1.types.Paragraph

Structural unit of text representing a number of words in certain order.

property

Additional information detected for the paragraph.

bounding_box

The bounding box for the paragraph. The vertices are in the order of top-left, top-right, bottom-right, bottom-left. When a rotation of the bounding box is detected the rotation is represented as around the top-left corner as defined when the text is read in the ‘natural’ orientation. For example: * when the text is horizontal it might look like: 0—1 || 3 —2 * when it’s rotated 180 degrees around the top-left corner it becomes: 2—3 || 1 —0 and the vertice order will still be (0, 1, 2, 3).

words

List of words in this paragraph.

class google.cloud.vision_v1.types.Position

A 3D position in the image, used primarily for Face detection landmarks. A valid Position must have both x and y coordinates. The position coordinates are in the same scale as the original image.

x

X coordinate.

y

Y coordinate.

z

Z coordinate (or depth).

class google.cloud.vision_v1.types.Property

A Property consists of a user-supplied name/value pair.

name

Name of the property.

value

Value of the property.

class google.cloud.vision_v1.types.SafeSearchAnnotation

Set of features pertaining to the image, computed by computer vision methods over safe-search verticals (for example, adult, spoof, medical, violence).

adult

Represents the adult content likelihood for the image.

spoof

Spoof likelihood. The likelihood that an modification was made to the image's canonical version to make it appear funny or offensive.

medical

Likelihood that this is a medical image.

violence

Violence likelihood.

class google.cloud.vision_v1.types.Symbol

A single symbol representation.

property

Additional information detected for the symbol.

bounding_box

The bounding box for the symbol. The vertices are in the order of top-left, top-right, bottom-right, bottom-left. When a rotation of the bounding box is detected the rotation is represented as around the top-left corner as defined when the text is read in the 'natural' orientation. For example: * when the text is horizontal it might look like: 0—1 | | 3 —2 * when it's rotated 180 degrees around the top-left corner it becomes: 2—3 | | 1 —0 and the vertice order will still be (0, 1, 2, 3).

text

The actual UTF-8 representation of the symbol.

class google.cloud.vision_v1.types.TextAnnotation

TextAnnotation contains a structured representation of OCR extracted text. The hierarchy of an OCR extracted text structure is like this: TextAnnotation -> Page -> Block -> Paragraph -> Word -> Symbol Each structural component, starting from Page, may further have their own properties. Properties describe detected languages, breaks etc.. Please refer to the [google.cloud.vision.v1.TextAnnotation.TextProperty][google.cloud.vision.v1.TextAnnotation.TextProperty] message definition below for more detail.

language_code

The BCP-47 language code, such as "en-US" or "sr-Latn". For more information, see http://www.unicode.org/reports/tr35/#Unicode_locale_identifier.

confidence

Confidence of detected language. Range [0, 1].

is_prefix

True if break prepends the element.

detected_languages

A list of detected languages together with confidence.

detected_break

Detected start or end of a text segment.

pages

List of pages detected by OCR.

text

UTF-8 text detected on the pages.

class DetectedBreak

Detected start or end of a structural component.

class DetectedLanguage

Detected language for a structural component.

class TextProperty

Additional information detected on the structural component.

class google.cloud.vision_v1.types.Vertex

X coordinate.

y

Y coordinate.

class google.cloud.vision_v1.types.WebDetection

Relevant information for the image from the Internet.

entity_id

Opaque entity ID.

score

Overall relevancy score for the web page. Not normalized and not comparable across different image queries.

description

Canonical description of the entity, in English.

url

The result web page URL.

web_entities

Deduced entities from similar images on the Internet.

full_matching_images

Fully matching images from the Internet. They're definite near-dups and most often a copy of the query image with merely a size change.

partial_matching_images

Partial matching images from the Internet. Those images are similar enough to share some key-point features. For example an original image will likely have partial matching for its crops.

pages_with_matching_images

Web pages containing the matching images from the Internet.

class WebEntity

Entity deduced from similar images on the Internet.

class WebImage

Metadata for online images.

class WebPage

Metadata for web pages.

class google.cloud.vision_v1.types.Word

A word representation.

property

Additional information detected for the word.

bounding_box

The bounding box for the word. The vertices are in the order of top-left, top-right, bottom-right, bottom-left. When a rotation of the bounding box is detected the rotation is represented as around the top-left

corner as defined when the text is read in the ‘natural’ orientation. For example: * when the text is horizontal it might look like: 0—1 || 3 —2 * when it’s rotated 180 degrees around the top-left corner it becomes: 2—3 || 1 —0 and the vertice order will still be (0, 1, 2, 3).

symbols

List of symbols in the word. The order of the symbols follows the natural reading order.

21.1 Getting started

The google-cloud library is pip install-able:

```
$ pip install google-cloud
```

21.1.1 Cloud Datastore

Google Cloud Datastore is a fully managed, schemaless database for storing non-relational data.

```
from google.cloud import datastore

client = datastore.Client()
key = client.key('Person')

entity = datastore.Entity(key=key)
entity['name'] = 'Your name'
entity['age'] = 25
client.put(entity)
```

21.1.2 Cloud Storage

Google Cloud Storage allows you to store data on Google infrastructure.

```
from google.cloud import storage

client = storage.Client()
bucket = client.get_bucket('<your-bucket-name>')
blob = bucket.blob('my-test-file.txt')
blob.upload_from_string('this is test content!')
```

21.1.3 Resources

- [GitHub](#)
- [Issues](#)
- [Stack Overflow](#)
- [PyPI](#)

g

- `google.cloud.bigquery.client`, 15
- `google.cloud.bigquery.dataset`, 18
- `google.cloud.bigquery.job`, 22
- `google.cloud.bigquery.query`, 38
- `google.cloud.bigquery.schema`, 41
- `google.cloud.bigquery.table`, 42
- `google.cloud.bigtable.client`, 60
- `google.cloud.bigtable.cluster`, 62
- `google.cloud.bigtable.column_family`, 74
- `google.cloud.bigtable.instance`, 64
- `google.cloud.bigtable.row`, 76
- `google.cloud.bigtable.row_data`, 84
- `google.cloud.bigtable.row_filters`, 87
- `google.cloud.bigtable.table`, 69
- `google.cloud.client`, 9
- `google.cloud.datastore`, 115
 - `batch`, 112
 - `client`, 99
 - `entity`, 102
 - `helpers`, 114
 - `key`, 104
 - `query`, 107
 - `transaction`, 110
- `google.cloud.dns.changes`, 136
- `google.cloud.dns.client`, 131
- `google.cloud.dns.resource_record_set`, 135
- `google.cloud.dns.zone`, 132
- `google.cloud.environment_vars`, 11
- `google.cloud.error_reporting.client`, 259
- `google.cloud.error_reporting.util`, 261
- `google.cloud.exceptions`, 10
- `google.cloud.iam`, 11
- `google.cloud.language_v1`, 144
- `google.cloud.language_v1.types`, 147
- `google.cloud.language_v1beta2`, 153
- `google.cloud.language_v1beta2.types`, 157
- `google.cloud.logging.client`, 289

- `google.cloud.logging.entries`, 295
- `google.cloud.logging.handlers.app_engine`, 302
- `google.cloud.logging.handlers.container_engine`, 303
- `google.cloud.logging.handlers.handlers`, 300
- `google.cloud.logging.handlers.transports.background`, 304
- `google.cloud.logging.handlers.transports.base`, 304
- `google.cloud.logging.handlers.transports.sync`, 303
- `google.cloud.logging.logger`, 292
- `google.cloud.logging.metric`, 296
- `google.cloud.logging.sink`, 298
- `google.cloud.monitoring.client`, 265
- `google.cloud.monitoring.group`, 274
- `google.cloud.monitoring.label`, 283
- `google.cloud.monitoring.metric`, 271
- `google.cloud.monitoring.query`, 277
- `google.cloud.monitoring.resource`, 273
- `google.cloud.monitoring.timeseries`, 282
- `google.cloud.operation`, 7
- `google.cloud.pubsub_v1.publisher.client`, 166
- `google.cloud.pubsub_v1.subscriber.client`, 174
- `google.cloud.pubsub_v1.types`, 188
- `google.cloud.resource_manager.client`, 197
- `google.cloud.resource_manager.project`, 199
- `google.cloud.runtimeconfig`, 208
 - `client`, 203
 - `config`, 204
 - `variable`, 206
- `google.cloud.spanner.batch`, 241
- `google.cloud.spanner.client`, 224
- `google.cloud.spanner.database`, 230

`google.cloud.spanner.instance`, 227
`google.cloud.spanner.keyset`, 240
`google.cloud.spanner.pool`, 236
`google.cloud.spanner.session`, 233
`google.cloud.spanner.snapshot`, 241
`google.cloud.spanner.streamed`, 242
`google.cloud.spanner.transaction`, 242
`google.cloud.speech_v1`, 250
`google.cloud.speech_v1.types`, 253
`google.cloud.storage.acl`, 331
`google.cloud.storage.batch`, 335
`google.cloud.storage.blob`, 311
`google.cloud.storage.bucket`, 321
`google.cloud.storage.client`, 335
`google.cloud.translate_v2.client`, 339
`google.cloud.vision_v1`, 345
`google.cloud.vision_v1.types`, 349

A

- AbstractSessionPool (class in google.cloud.spanner.pool), 236
- access_grants (google.cloud.bigquery.dataset.Dataset attribute), 19
- AccessGrant (class in google.cloud.bigquery.dataset), 18
- ack() (google.cloud.pubsub_v1.subscriber.message.Message method), 187
- ack_deadline_seconds (google.cloud.pubsub_v1.types.ModifyAckDeadlineRequest attribute), 191
- ack_deadline_seconds (google.cloud.pubsub_v1.types.Subscription attribute), 194
- ack_id (google.cloud.pubsub_v1.types.ReceivedMessage attribute), 192
- ack_ids (google.cloud.pubsub_v1.types.AcknowledgeRequest attribute), 188
- ack_ids (google.cloud.pubsub_v1.types.ModifyAckDeadlineRequest attribute), 191
- ack_ids (google.cloud.pubsub_v1.types.StreamingPullRequest attribute), 193
- acknowledge() (google.cloud.pubsub_v1.subscriber.client.Client method), 174
- AcknowledgeRequest (class in google.cloud.pubsub_v1.types), 188
- ACL (class in google.cloud.storage.acl), 332
- acl (google.cloud.storage.blob.Blob attribute), 311
- acl (google.cloud.storage.bucket.Bucket attribute), 321
- add_done_callback() (google.cloud.bigquery.job.CopyJob method), 22
- add_done_callback() (google.cloud.bigquery.job.ExtractTableToStorageJob method), 26
- add_done_callback() (google.cloud.bigquery.job.LoadTableFromStorageJob method), 30
- add_done_callback() (google.cloud.bigquery.job.QueryJob method), 34
- add_entity() (google.cloud.storage.acl.ACL method), 332
- add_filter() (google.cloud.datastore.Query method), 125
- add_filter() (google.cloud.datastore.query.Query method), 107
- add_record_set() (google.cloud.dns.changes.Changes method), 136
- additions (google.cloud.dns.changes.Changes attribute), 136
- ADMIN_SCOPE (in google.cloud.bigtable.client), 60
- adult (google.cloud.vision_v1.types.SafeSearchAnnotation attribute), 355
- align() (google.cloud.monitoring.query.Query method), 277
- Aligner (class in google.cloud.monitoring.query), 277
- all() (google.cloud.storage.acl.ACL method), 332
- all_authenticated() (google.cloud.storage.acl.ACL method), 332
- all_users() (google.cloud.iam.Policy static method), 11
- allocate_ids() (google.cloud.datastore.Client method), 118
- allocate_ids() (google.cloud.datastore.client.Client method), 99
- allow_jagged_rows (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 30
- allow_large_results (google.cloud.bigquery.job.QueryJob attribute), 34
- allow_quoted_newlines (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 30
- alternatives (google.cloud.speech_v1.types.SpeechRecognitionResult attribute), 255
- alternatives (google.cloud.speech_v1.types.StreamingRecognitionResult attribute), 255
- analyze_entities() (google.cloud.language_v1.LanguageServiceClient method), 144
- analyze_entities() (google.cloud.language_v1beta2.LanguageServiceClient method), 153
- analyze_entity_sentiment() (google.cloud.language_v1.LanguageServiceClient method), 145
- analyze_entity_sentiment() (google.cloud.language_v1beta2.LanguageServiceClient method), 154
- analyze_sentiment() (google.cloud.language_v1.LanguageServiceClient method), 145

- method), 146
- analyze_sentiment() (google.cloud.language_v1beta2.LanguageServiceClient google.cloud.language_v1.types), 149
- method), 154
- analyze_syntax() (google.cloud.language_v1.LanguageServiceClient google.cloud.language_v1beta2.types), 158
- method), 146
- analyze_syntax() (google.cloud.language_v1beta2.LanguageServiceClient google.cloud.language_v1.types), 149
- method), 155
- AnalyzeEntitiesRequest (class in google.cloud.language_v1beta2.types), 159
- google.cloud.language_v1.types), 147
- AnalyzeEntitiesRequest (class in google.cloud.language_v1.types), 149
- google.cloud.language_v1beta2.types), 157
- AnalyzeEntitiesResponse (class in google.cloud.language_v1beta2.types), 159
- google.cloud.language_v1.types), 148
- AnalyzeEntitiesResponse (class in google.cloud.language_v1beta2.types), 157
- google.cloud.language_v1beta2.types), 147
- AnalyzeEntitySentimentRequest (class in google.cloud.language_v1.types), 148
- google.cloud.language_v1beta2.types), 157
- AnalyzeEntitySentimentRequest (class in google.cloud.language_v1.types), 148
- google.cloud.language_v1beta2.types), 157
- AnalyzeEntitySentimentResponse (class in google.cloud.language_v1.types), 148
- google.cloud.language_v1beta2.types), 157
- AnalyzeEntitySentimentResponse (class in google.cloud.language_v1.types), 148
- google.cloud.language_v1beta2.types), 157
- AnalyzeSentimentRequest (class in google.cloud.language_v1.types), 148
- google.cloud.language_v1beta2.types), 157
- AnalyzeSentimentRequest (class in google.cloud.language_v1.types), 148
- google.cloud.language_v1beta2.types), 157
- AnalyzeSentimentResponse (class in google.cloud.language_v1.types), 148
- google.cloud.language_v1beta2.types), 157
- AnalyzeSentimentResponse (class in google.cloud.language_v1.types), 148
- google.cloud.language_v1beta2.types), 157
- AnalyzeSyntaxRequest (class in google.cloud.language_v1.types), 148
- google.cloud.language_v1beta2.types), 158
- AnalyzeSyntaxRequest (class in google.cloud.language_v1.types), 148
- google.cloud.language_v1beta2.types), 158
- AnalyzeSyntaxResponse (class in google.cloud.language_v1.types), 149
- google.cloud.language_v1beta2.types), 158
- AnalyzeSyntaxResponse (class in google.cloud.language_v1.types), 149
- google.cloud.language_v1beta2.types), 158
- ancestor (google.cloud.datastore.Query attribute), 126
- ancestor (google.cloud.datastore.query.Query attribute), 108
- anger_likelihood (google.cloud.vision_v1.types.FaceAnnotation attribute), 352
- annotate_image() (google.cloud.vision_v1.ImageAnnotatorClient attribute), 253
- method), 345
- annotate_text() (google.cloud.language_v1.LanguageServiceClient method), 11
- method), 147
- annotate_text() (google.cloud.language_v1beta2.LanguageServiceClient attribute), 30
- method), 156
- AnnotateImageRequest (class in google.cloud.vision_v1.types), 349
- AnnotateImageResponse (class in google.cloud.vision_v1.types), 349
- AnnotateTextRequest (class in google.cloud.language_v1.types), 149
- AnnotateTextRequest (class in google.cloud.language_v1.types), 149
- AnnotateTextRequest.Features (class in google.cloud.language_v1beta2.types), 159
- AnnotateTextRequest.Features (class in google.cloud.language_v1beta2.types), 159
- AnnotateTextResponse (class in google.cloud.language_v1.types), 149
- AnnotateTextResponse (class in google.cloud.language_v1beta2.types), 159
- append_cell_value() (google.cloud.bigtable.row.AppendRow method), 77
- append_value() (google.cloud.bigtable.row_data.PartialCellData method), 84
- AppendRow (class in google.cloud.bigtable.row), 76
- AppEngineHandler (class in google.cloud.logging.handlers.app_engine), 302
- ApplyLabelFilter (class in google.cloud.bigtable.row_filters), 87
- as_dataframe() (google.cloud.monitoring.query.Query method), 278
- aspect (google.cloud.language_v1.types.PartOfSpeech attribute), 151
- aspect (google.cloud.language_v1beta2.types.PartOfSpeech attribute), 161
- aspect_ratios (google.cloud.vision_v1.types.CropHintsParams attribute), 350
- attributes (google.cloud.pubsub_v1.subscriber.message.Message attribute), 187
- attributes (google.cloud.pubsub_v1.types.PubsubMessage attribute), 191
- attributes (google.cloud.pubsub_v1.types.PushConfig attribute), 192
- attributes (Message attribute), 187
- audio (google.cloud.speech_v1.types.LongRunningRecognizeRequest attribute), 253
- audio (google.cloud.speech_v1.types.RecognizeRequest attribute), 254
- audio_content (google.cloud.speech_v1.types.StreamingRecognizeRequest attribute), 256
- audio_source (google.cloud.speech_v1.types.RecognitionAudio attribute), 253
- authenticated_users() (google.cloud.iam.Policy static method), 11
- autodetect (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 30
- AutoDetectSchema (class in google.cloud.bigquery.job), 22

B

BackgroundThreadTransport (class in

- google.cloud.logging.handlers.transports.background_thread (in module google.cloud.logging.handlers.transports), 11
 - 304
 - BASE (in module google.cloud.translate_v2.client), 339
 - Batch (class in google.cloud.datastore), 116
 - Batch (class in google.cloud.datastore.batch), 112
 - Batch (class in google.cloud.logging.logger), 292
 - Batch (class in google.cloud.spanner.batch), 241
 - Batch (class in google.cloud.storage.batch), 335
 - batch() (google.cloud.datastore.Client method), 118
 - batch() (google.cloud.datastore.client.Client method), 100
 - batch() (google.cloud.logging.logger.Logger method), 293
 - batch() (google.cloud.pubsub_v1.publisher.client.Client method), 166
 - batch() (google.cloud.spanner.database.Database method), 231
 - batch() (google.cloud.spanner.session.Session method), 234
 - batch() (google.cloud.storage.client.Client method), 336
 - batch_annotate_images() (google.cloud.vision_v1.ImageAnnotatorClient method), 346
 - BatchAnnotateImagesRequest (class in google.cloud.vision_v1.types), 349
 - BatchAnnotateImagesResponse (class in google.cloud.vision_v1.types), 349
 - BatchCheckout (class in google.cloud.spanner.database), 230
 - BatchSettings (class in google.cloud.pubsub_v1.types), 188
 - begin() (google.cloud.bigquery.job.CopyJob method), 23
 - begin() (google.cloud.bigquery.job.ExtractTableToStorageJob method), 26
 - begin() (google.cloud.bigquery.job.LoadTableFromStorageJob method), 30
 - begin() (google.cloud.bigquery.job.QueryJob method), 34
 - begin() (google.cloud.datastore.Batch method), 116
 - begin() (google.cloud.datastore.batch.Batch method), 113
 - begin() (google.cloud.datastore.Transaction method), 129
 - begin() (google.cloud.datastore.transaction.Transaction method), 111
 - begin() (google.cloud.spanner.snapshot.Snapshot method), 241
 - begin() (google.cloud.spanner.transaction.Transaction method), 242
 - begin_offset (google.cloud.language_v1.types.TextSpan attribute), 152
 - begin_offset (google.cloud.language_v1beta2.types.TextSpan attribute), 162
 - begin_pending_transactions() (google.cloud.spanner.pool.TransactionPingingPool method), 240
 - BIGTABLE_EMULATOR (in module google.cloud.environment_vars), 11
 - bind() (google.cloud.spanner.pool.AbstractSessionPool method), 236
 - bind() (google.cloud.spanner.pool.BurstyPool method), 237
 - bind() (google.cloud.spanner.pool.FixedSizePool method), 238
 - bind() (google.cloud.spanner.pool.PingingPool method), 239
 - bind() (google.cloud.spanner.pool.TransactionPingingPool method), 240
 - Blob (class in google.cloud.storage.blob), 311
 - blob() (google.cloud.storage.bucket.Bucket method), 321
 - Block (class in google.cloud.vision_v1.types), 350
 - block_type (google.cloud.vision_v1.types.Block attribute), 350
 - BlockAllFilter (class in google.cloud.bigtable.row_filters), 87
 - blocks (google.cloud.vision_v1.types.Page attribute), 354
 - blurred_likelihood (google.cloud.vision_v1.types.FaceAnnotation attribute), 352
 - bounding_box (google.cloud.vision_v1.types.Block attribute), 350
 - bounding_box (google.cloud.vision_v1.types.Paragraph attribute), 354
 - bounding_box (google.cloud.vision_v1.types.Symbol attribute), 355
 - bounding_box (google.cloud.vision_v1.types.Word attribute), 356
 - bounding_poly (google.cloud.vision_v1.types.CropHint attribute), 350
 - bounding_poly (google.cloud.vision_v1.types.EntityAnnotation attribute), 351
 - bounding_poly (google.cloud.vision_v1.types.FaceAnnotation attribute), 351
 - BoundingPoly (class in google.cloud.vision_v1.types), 350
 - Bucket (class in google.cloud.storage.bucket), 321
 - bucket() (google.cloud.storage.client.Client method), 336
 - BucketACL (class in google.cloud.storage.acl), 334
 - build_flask_context() (in module google.cloud.error_reporting.util), 261
 - BurstyPool (class in google.cloud.spanner.pool), 237
- ## C
- cache_control (google.cloud.storage.blob.Blob attribute), 311
 - cache_hit (google.cloud.bigquery.query.QueryResults attribute), 38
 - cancel() (google.cloud.bigquery.job.CopyJob method), 23
 - cancel() (google.cloud.bigquery.job.ExtractTableToStorageJob method), 26
 - cancel() (google.cloud.bigquery.job.LoadTableFromStorageJob method), 30

- cancel() (google.cloud.bigquery.job.QueryJob method), 34
- cancel() (google.cloud.bigtable.row_data.PartialRowsData method), 85
- cancelled() (google.cloud.bigquery.job.CopyJob method), 23
- cancelled() (google.cloud.bigquery.job.ExtractTableToStorageJob method), 26
- cancelled() (google.cloud.bigquery.job.LoadTableFromStorageJob method), 30
- cancelled() (google.cloud.bigquery.job.QueryJob method), 34
- case (google.cloud.language_v1.types.PartOfSpeech attribute), 151
- case (google.cloud.language_v1beta2.types.PartOfSpeech attribute), 161
- categories (google.cloud.language_v1beta2.types.AnnotateTextRequest attribute), 159
- categories (google.cloud.language_v1beta2.types.ClassifyTextRequest attribute), 159
- Cell (class in google.cloud.bigtable.row_data), 84
- cells (google.cloud.bigtable.row_data.PartialRowData attribute), 85
- CellsColumnLimitFilter (class in google.cloud.bigtable.row_filters), 87
- CellsRowLimitFilter (class in google.cloud.bigtable.row_filters), 88
- CellsRowOffsetFilter (class in google.cloud.bigtable.row_filters), 88
- Changes (class in google.cloud.dns.changes), 136
- changes() (google.cloud.dns.zone.ManagedZone method), 132
- chunk_size (google.cloud.storage.blob.Blob attribute), 311
- ClassificationCategory (class in google.cloud.language_v1beta2.types), 159
- classify_text (google.cloud.language_v1beta2.types.AnnotateTextRequest attribute), 158
- classify_text() (google.cloud.language_v1beta2.LanguageServiceClient method), 156
- ClassifyTextRequest (class in google.cloud.language_v1beta2.types), 159
- ClassifyTextResponse (class in google.cloud.language_v1beta2.types), 159
- clear() (google.cloud.bigtable.row.AppendRow method), 77
- clear() (google.cloud.bigtable.row.ConditionalRow method), 79
- clear() (google.cloud.bigtable.row.DirectRow method), 81
- clear() (google.cloud.spanner.pool.AbstractSessionPool method), 236
- clear() (google.cloud.spanner.pool.BurstyPool method), 237
- clear() (google.cloud.spanner.pool.FixedSizePool method), 238
- clear() (google.cloud.spanner.pool.PingingPool method), 239
- clear() (google.cloud.storage.acl.ACL method), 332
- Client (class in google.cloud.bigquery.client), 15
- Client (class in google.cloud.bigtable.client), 60
- Client (class in google.cloud.client), 9
- Client (class in google.cloud.datastore), 118
- Client (class in google.cloud.datastore.client), 99
- Client (class in google.cloud.dns.client), 131
- Client (class in google.cloud.error_reporting.client), 259
- Client (class in google.cloud.logging.client), 289
- Client (class in google.cloud.monitoring.client), 265
- Client (class in google.cloud.pubsub_v1.publisher.client), 166
- Client (class in google.cloud.pubsub_v1.subscriber.client), 174
- Client (class in google.cloud.resource_manager.client), 197
- Client (class in google.cloud.runtimeconfig), 208
- Client (class in google.cloud.runtimeconfig.client), 203
- Client (class in google.cloud.spanner.client), 225
- Client (class in google.cloud.storage.client), 335
- Client (class in google.cloud.translate_v2.client), 339
- client (google.cloud.logging.logger.Logger attribute), 293
- client (google.cloud.logging.metric.Metric attribute), 297
- client (google.cloud.logging.sink.Sink attribute), 298
- client (google.cloud.runtimeconfig.config.Config attribute), 204
- client (google.cloud.runtimeconfig.variable.Variable attribute), 206
- client (google.cloud.storage.acl.ACL attribute), 332
- client (google.cloud.storage.acl.BucketACL attribute), 334
- client (google.cloud.storage.acl.ObjectACL attribute), 334
- client (google.cloud.storage.blob.Blob attribute), 312
- client (google.cloud.storage.bucket.Bucket attribute), 322
- ClientWithProject (class in google.cloud.client), 10
- close() (google.cloud.pubsub_v1.subscriber.policy.thread.Policy method), 186
- CloudLoggingHandler (class in google.cloud.logging.handlers.handlers), 300
- Cluster (class in google.cloud.bigtable.cluster), 62
- cluster() (google.cloud.bigtable.instance.Instance method), 65
- color (google.cloud.vision_v1.types.ColorInfo attribute), 350
- ColorInfo (class in google.cloud.vision_v1.types), 350
- colors (google.cloud.vision_v1.types.DominantColorsAnnotation attribute), 351
- column_family() (google.cloud.bigtable.table.Table

- method), 69
- ColumnFamily (class in google.cloud.bigtable.column_family), 74
- ColumnQualifierRegexFilter (class in google.cloud.bigtable.row_filters), 88
- ColumnRangeFilter (class in google.cloud.bigtable.row_filters), 88
- commit() (google.cloud.bigtable.row.AppendRow method), 77
- commit() (google.cloud.bigtable.row.ConditionalRow method), 79
- commit() (google.cloud.bigtable.row.DirectRow method), 81
- commit() (google.cloud.datastore.Batch method), 117
- commit() (google.cloud.datastore.batch.Batch method), 113
- commit() (google.cloud.datastore.Transaction method), 129
- commit() (google.cloud.datastore.transaction.Transaction method), 111
- commit() (google.cloud.logging.logger.Batch method), 292
- commit() (google.cloud.spanner.batch.Batch method), 241
- commit() (google.cloud.spanner.transaction.Transaction method), 242
- committed (google.cloud.spanner.batch.Batch attribute), 242
- committed (google.cloud.spanner.transaction.Transaction attribute), 242
- complete (google.cloud.bigquery.query.QueryResults attribute), 38
- complete (google.cloud.operation.Operation attribute), 7
- completed_key() (google.cloud.datastore.Key method), 123
- completed_key() (google.cloud.datastore.key.Key method), 105
- component_count (google.cloud.storage.blob.Blob attribute), 312
- compose() (google.cloud.storage.blob.Blob method), 312
- Compression (class in google.cloud.bigquery.job), 22
- compression (google.cloud.bigquery.job.ExtractTableToStorage job attribute), 27
- ConditionalRow (class in google.cloud.bigtable.row), 78
- ConditionalRowFilter (class in google.cloud.bigtable.row_filters), 89
- confidence (google.cloud.language_v1beta2.types.ClassificationAttribute), 159
- confidence (google.cloud.speech_v1.types.SpeechRecognitionAlternative attribute), 255
- confidence (google.cloud.vision_v1.types.CropHint attribute), 350
- confidence (google.cloud.vision_v1.types.EntityAnnotation attribute), 351
- confidence (google.cloud.vision_v1.types.TextAnnotation attribute), 355
- Config (class in google.cloud.runtimeconfig.config), 204
- config (google.cloud.speech_v1.types.LongRunningRecognizeRequest attribute), 253
- config (google.cloud.speech_v1.types.RecognizeRequest attribute), 254
- config (google.cloud.speech_v1.types.StreamingRecognitionConfig attribute), 255
- config() (google.cloud.runtimeconfig.Client method), 208
- config() (google.cloud.runtimeconfig.client.Client method), 203
- configure_website() (google.cloud.storage.bucket.Bucket method), 322
- consume_all() (google.cloud.bigtable.row_data.PartialRowsData method), 85
- consume_all() (google.cloud.spanner.streamed.StreamedResultSet method), 243
- consume_next() (google.cloud.bigtable.row_data.PartialRowsData method), 85
- consume_next() (google.cloud.spanner.streamed.StreamedResultSet method), 243
- ContainerEngineHandler (class in google.cloud.logging.handlers.container_engine), 303
- content (google.cloud.language_v1.types.Document attribute), 150
- content (google.cloud.language_v1.types.TextSpan attribute), 152
- content (google.cloud.language_v1beta2.types.Document attribute), 160
- content (google.cloud.language_v1beta2.types.TextSpan attribute), 162
- content (google.cloud.speech_v1.types.RecognitionAudio attribute), 253
- content (google.cloud.vision_v1.types.Image attribute), 353
- content_disposition (google.cloud.storage.blob.Blob attribute), 312
- content_encoding (google.cloud.storage.blob.Blob attribute), 312
- content_language (google.cloud.storage.blob.Blob attribute), 312
- content_type (google.cloud.storage.blob.Blob attribute), 312
- copy() (google.cloud.bigtable.client.Client method), 61
- copy() (google.cloud.bigtable.cluster.Cluster method), 63
- copy() (google.cloud.bigtable.instance.Instance method), 65
- copy() (google.cloud.monitoring.query.Query method), 279
- copy() (google.cloud.spanner.client.Client method), 225
- copy() (google.cloud.spanner.instance.Instance method), 227

[copy_blob\(\)](#) (google.cloud.storage.bucket.Bucket method), [322](#)
[copy_table\(\)](#) (google.cloud.bigquery.client.Client method), [15](#)
[CopyJob](#) (class in google.cloud.bigquery.job), [22](#)
[cors](#) (google.cloud.storage.bucket.Bucket attribute), [322](#)
[crc32c](#) (google.cloud.storage.blob.Blob attribute), [312](#)
[create\(\)](#) (google.cloud.bigquery.dataset.Dataset method), [19](#)
[create\(\)](#) (google.cloud.bigquery.table.Table method), [42](#)
[create\(\)](#) (google.cloud.bigtable.cluster.Cluster method), [63](#)
[create\(\)](#) (google.cloud.bigtable.column_family.ColumnFamily method), [75](#)
[create\(\)](#) (google.cloud.bigtable.instance.Instance method), [65](#)
[create\(\)](#) (google.cloud.bigtable.table.Table method), [69](#)
[create\(\)](#) (google.cloud.dns.changes.Changes method), [136](#)
[create\(\)](#) (google.cloud.dns.zone.ManagedZone method), [133](#)
[create\(\)](#) (google.cloud.logging.metric.Metric method), [297](#)
[create\(\)](#) (google.cloud.logging.sink.Sink method), [298](#)
[create\(\)](#) (google.cloud.monitoring.group.Group method), [274](#)
[create\(\)](#) (google.cloud.monitoring.metric.MetricDescriptor method), [272](#)
[create\(\)](#) (google.cloud.resource_manager.project.Project method), [199](#)
[create\(\)](#) (google.cloud.spanner.database.Database method), [231](#)
[create\(\)](#) (google.cloud.spanner.instance.Instance method), [227](#)
[create\(\)](#) (google.cloud.spanner.session.Session method), [234](#)
[create\(\)](#) (google.cloud.storage.bucket.Bucket method), [323](#)
[create_bucket\(\)](#) (google.cloud.storage.client.Client method), [336](#)
[create_disposition](#) (google.cloud.bigquery.job.CopyJob attribute), [23](#)
[create_disposition](#) (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), [30](#)
[create_disposition](#) (google.cloud.bigquery.job.QueryJob attribute), [35](#)
[create_resumable_upload_session\(\)](#) (google.cloud.storage.blob.Blob method), [313](#)
[create_snapshot\(\)](#) (google.cloud.pubsub_v1.subscriber.client.Client method), [175](#)
[create_subscription\(\)](#) (google.cloud.pubsub_v1.subscriber.client.Client method), [176](#)
[create_topic\(\)](#) (google.cloud.pubsub_v1.publisher.client.Client method), [167](#)
[created](#) (google.cloud.bigquery.dataset.Dataset attribute), [19](#)
[created](#) (google.cloud.bigquery.job.CopyJob attribute), [23](#)
[created](#) (google.cloud.bigquery.job.ExtractTableToStorageJob attribute), [27](#)
[created](#) (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), [30](#)
[created](#) (google.cloud.bigquery.job.QueryJob attribute), [35](#)
[created](#) (google.cloud.bigquery.table.Table attribute), [42](#)
[created](#) (google.cloud.dns.zone.ManagedZone attribute), [133](#)
[CreateDisposition](#) (class in google.cloud.bigquery.job), [25](#)
[CreateSnapshotRequest](#) (class in google.cloud.pubsub_v1.types), [188](#)
[credentials](#) (google.cloud.bigtable.client.Client attribute), [61](#)
[credentials](#) (google.cloud.spanner.client.Client attribute), [225](#)
[crop_hints\(\)](#) (google.cloud.vision_v1.ImageAnnotatorClient method), [346](#)
[crop_hints_annotation](#) (google.cloud.vision_v1.types.AnnotateImageResponse attribute), [349](#)
[crop_hints_params](#) (google.cloud.vision_v1.types.ImageContext attribute), [353](#)
[CropHint](#) (class in google.cloud.vision_v1.types), [350](#)
[CropHintsAnnotation](#) (class in google.cloud.vision_v1.types), [350](#)
[CropHintsParams](#) (class in google.cloud.vision_v1.types), [350](#)
[current\(\)](#) (google.cloud.datastore.Batch method), [117](#)
[current\(\)](#) (google.cloud.datastore.batch.Batch method), [113](#)
[current\(\)](#) (google.cloud.datastore.Transaction method), [129](#)
[current\(\)](#) (google.cloud.datastore.transaction.Transaction method), [111](#)
[current\(\)](#) (google.cloud.storage.batch.Batch method), [335](#)
[current_batch](#) (google.cloud.datastore.Client attribute), [118](#)
[current_batch](#) (google.cloud.datastore.client.Client attribute), [100](#)
[current_batch](#) (google.cloud.storage.client.Client attribute), [336](#)
[current_transaction](#) (google.cloud.datastore.Client attribute), [119](#)
[current_transaction](#) (google.cloud.datastore.client.Client attribute), [100](#)

D

[data](#) (google.cloud.pubsub_v1.subscriber.message.Message attribute), [187](#)

- data (google.cloud.pubsub_v1.types.PubsubMessage attribute), 191
- data (Message attribute), 187
- DATA_API_HOST (in module google.cloud.bigtable.client), 62
- DATA_SCOPE (in module google.cloud.bigtable.client), 62
- Database (class in google.cloud.spanner.database), 230
- database() (google.cloud.spanner.instance.Instance method), 228
- database_admin_api (google.cloud.spanner.client.Client attribute), 225
- Dataset (class in google.cloud.bigquery.dataset), 19
- dataset() (google.cloud.bigquery.client.Client method), 16
- dataset_id (google.cloud.bigquery.dataset.Dataset attribute), 19
- dataset_name (google.cloud.bigquery.table.Table attribute), 42
- ddl_statements (google.cloud.spanner.database.Database attribute), 231
- default_dataset (google.cloud.bigquery.job.QueryJob attribute), 35
- default_dataset (google.cloud.bigquery.query.QueryResults attribute), 39
- default_object_acl (google.cloud.storage.bucket.Bucket attribute), 323
- DEFAULT_RETRY_TIMEOUT_SECS (in module google.cloud.spanner.session), 233
- DEFAULT_SERVE_NODES (in module google.cloud.bigtable.cluster), 64
- default_table_expiration_ms (google.cloud.bigquery.dataset.Dataset attribute), 20
- DefaultObjectACL (class in google.cloud.storage.acl), 334
- delete() (google.cloud.bigquery.dataset.Dataset method), 20
- delete() (google.cloud.bigquery.table.Table method), 43
- delete() (google.cloud.bigtable.cluster.Cluster method), 63
- delete() (google.cloud.bigtable.column_family.ColumnFamily method), 75
- delete() (google.cloud.bigtable.instance.Instance method), 65
- delete() (google.cloud.bigtable.row.ConditionalRow method), 79
- delete() (google.cloud.bigtable.row.DirectRow method), 81
- delete() (google.cloud.bigtable.table.Table method), 70
- delete() (google.cloud.datastore.Batch method), 117
- delete() (google.cloud.datastore.batch.Batch method), 113
- delete() (google.cloud.datastore.Client method), 119
- delete() (google.cloud.datastore.client.Client method), 100
- delete() (google.cloud.datastore.transaction.Transaction method), 111
- delete() (google.cloud.dns.zone.ManagedZone method), 133
- delete() (google.cloud.logging.logger.Logger method), 293
- delete() (google.cloud.logging.metric.Metric method), 297
- delete() (google.cloud.logging.sink.Sink method), 298
- delete() (google.cloud.monitoring.group.Group method), 275
- delete() (google.cloud.monitoring.metric.MetricDescriptor method), 273
- delete() (google.cloud.resource_manager.project.Project method), 199
- delete() (google.cloud.spanner.instance.Instance method), 228
- delete() (google.cloud.spanner.session.Session method), 234
- delete() (google.cloud.storage.blob.Blob method), 313
- delete() (google.cloud.storage.bucket.Bucket method), 323
- delete_blob() (google.cloud.storage.bucket.Bucket method), 323
- delete_blobs() (google.cloud.storage.bucket.Bucket method), 324
- delete_cell() (google.cloud.bigtable.row.ConditionalRow method), 79
- delete_cell() (google.cloud.bigtable.row.DirectRow method), 82
- delete_cells() (google.cloud.bigtable.row.ConditionalRow method), 80
- delete_cells() (google.cloud.bigtable.row.DirectRow method), 82
- delete_multi() (google.cloud.datastore.Client method), 119
- delete_multi() (google.cloud.datastore.client.Client method), 100
- delete_record_set() (google.cloud.dns.changes.Changes method), 136
- delete_snapshot() (google.cloud.pubsub_v1.subscriber.client.Client method), 177
- delete_subscription() (google.cloud.pubsub_v1.subscriber.client.Client method), 177
- delete_topic() (google.cloud.pubsub_v1.publisher.client.Client method), 167
- DeleteSnapshotRequest (class in google.cloud.pubsub_v1.types), 188
- DeleteSubscriptionRequest (class in google.cloud.pubsub_v1.types), 189
- DeleteTopicRequest (class in google.cloud.pubsub_v1.types), 189

[deletions](#) (google.cloud.dns.changes.Changes attribute), [136](#)
[dependency_edge](#) (google.cloud.language_v1.types.Token attribute), [152](#)
[dependency_edge](#) (google.cloud.language_v1beta2.types.Token attribute), [162](#)
[DependencyEdge](#) (class in google.cloud.language_v1.types), [150](#)
[DependencyEdge](#) (class in google.cloud.language_v1beta2.types), [159](#)
[description](#) (google.cloud.bigquery.dataset.Dataset attribute), [20](#)
[description](#) (google.cloud.bigquery.schema.SchemaField attribute), [41](#)
[description](#) (google.cloud.bigquery.table.Table attribute), [43](#)
[description](#) (google.cloud.dns.zone.ManagedZone attribute), [133](#)
[description](#) (google.cloud.runtimeconfig.config.Config attribute), [204](#)
[description](#) (google.cloud.vision_v1.types.EntityAnnotation attribute), [351](#)
[description](#) (google.cloud.vision_v1.types.WebDetection attribute), [356](#)
[destination](#) (google.cloud.bigquery.job.QueryJob attribute), [35](#)
[destination_format](#) (google.cloud.bigquery.job.ExtractTableToStorageJob attribute), [27](#)
[DestinationFormat](#) (class in google.cloud.bigquery.job), [26](#)
[detect_language\(\)](#) (google.cloud.translate_v2.client.Client method), [339](#)
[detected_break](#) (google.cloud.vision_v1.types.TextAnnotation attribute), [355](#)
[detected_languages](#) (google.cloud.vision_v1.types.TextAnnotation attribute), [355](#)
[detection_confidence](#) (google.cloud.vision_v1.types.FaceAnnotation attribute), [352](#)
[DirectRow](#) (class in google.cloud.bigtable.row), [81](#)
[DISABLE_GRPC](#) (in module google.cloud.environment_vars), [11](#)
[disable_logging\(\)](#) (google.cloud.storage.bucket.Bucket method), [324](#)
[disable_website\(\)](#) (google.cloud.storage.bucket.Bucket method), [324](#)
[distinct_on](#) (google.cloud.datastore.Query attribute), [126](#)
[distinct_on](#) (google.cloud.datastore.query.Query attribute), [108](#)
[Document](#) (class in google.cloud.language_v1.types), [150](#)
[Document](#) (class in google.cloud.language_v1beta2.types), [160](#)
[document](#) (google.cloud.language_v1.types.AnalyzeEntitiesRequest attribute), [147](#)
[document](#) (google.cloud.language_v1.types.AnalyzeEntitySentimentRequest attribute), [148](#)
[document](#) (google.cloud.language_v1.types.AnalyzeSentimentRequest attribute), [148](#)
[document](#) (google.cloud.language_v1.types.AnalyzeSyntaxRequest attribute), [148](#)
[document](#) (google.cloud.language_v1.types.AnnotateTextRequest attribute), [149](#)
[document](#) (google.cloud.language_v1beta2.types.AnalyzeEntitiesRequest attribute), [157](#)
[document](#) (google.cloud.language_v1beta2.types.AnalyzeEntitySentimentRequest attribute), [157](#)
[document](#) (google.cloud.language_v1beta2.types.AnalyzeSentimentRequest attribute), [157](#)
[document](#) (google.cloud.language_v1beta2.types.AnalyzeSyntaxRequest attribute), [158](#)
[document](#) (google.cloud.language_v1beta2.types.AnnotateTextRequest attribute), [158](#)
[document](#) (google.cloud.language_v1beta2.types.ClassifyTextRequest attribute), [159](#)
[document_sentiment](#) (google.cloud.language_v1.types.AnalyzeSentimentRequest attribute), [148](#)
[document_sentiment](#) (google.cloud.language_v1.types.AnnotateTextResponse attribute), [149](#)
[document_sentiment](#) (google.cloud.language_v1beta2.types.AnalyzeSentimentRequest attribute), [158](#)
[document_sentiment](#) (google.cloud.language_v1beta2.types.AnnotateTextRequest attribute), [159](#)
[document_text_detection\(\)](#) (google.cloud.vision_v1.ImageAnnotatorClient method), [346](#)
[domain\(\)](#) (google.cloud.iam.Policy static method), [12](#)
[domain\(\)](#) (google.cloud.storage.acl.ACL method), [332](#)
[dominant_colors](#) (google.cloud.vision_v1.types.ImageProperties attribute), [353](#)
[DominantColorsAnnotation](#) (class in google.cloud.vision_v1.types), [351](#)
[done\(\)](#) (google.cloud.bigquery.job.CopyJob method), [23](#)
[done\(\)](#) (google.cloud.bigquery.job.ExtractTableToStorageJob method), [27](#)
[done\(\)](#) (google.cloud.bigquery.job.LoadTableFromStorageJob method), [30](#)
[done\(\)](#) (google.cloud.bigquery.job.QueryJob method), [35](#)
[download_as_string\(\)](#) (google.cloud.storage.blob.Blob method), [314](#)
[download_to_file\(\)](#) (google.cloud.storage.blob.Blob method), [314](#)
[download_to_filename\(\)](#) (google.cloud.storage.blob.Blob method), [314](#)
[drop\(\)](#) (google.cloud.spanner.database.Database method), [231](#)
[dry_run](#) (google.cloud.bigquery.job.QueryJob attribute), [35](#)
[dry_run](#) (google.cloud.bigquery.query.QueryResults attribute), [35](#)

tribute), 39

E

EDITOR_ROLE (in module google.cloud.iam), 11

editors (google.cloud.iam.Policy attribute), 12

emit() (google.cloud.logging.handlers.handlers.CloudLoggingHandler attribute), 159
method), 301

enable_logging() (google.cloud.storage.bucket.Bucket attribute), 324
method), 324

enable_word_time_offsets
(google.cloud.speech_v1.types.RecognitionConfig attribute), 254

Encoding (class in google.cloud.bigquery.job), 26

encoding (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 31

encoding (google.cloud.speech_v1.types.RecognitionConfig attribute), 254

encoding_type (google.cloud.language_v1.types.AnalyzeEntitiesRequest attribute), 147

encoding_type (google.cloud.language_v1.types.AnalyzeEntitySentimentRequest attribute), 148

encoding_type (google.cloud.language_v1.types.AnalyzeSentimentRequest attribute), 148

encoding_type (google.cloud.language_v1.types.AnalyzeSyntaxRequest attribute), 148

encoding_type (google.cloud.language_v1.types.AnnotateTextRequest attribute), 149

encoding_type (google.cloud.language_v1beta2.types.AnalyzeEntitiesRequest attribute), 157

encoding_type (google.cloud.language_v1beta2.types.AnalyzeEntitySentimentRequest attribute), 157

encoding_type (google.cloud.language_v1beta2.types.AnalyzeSentimentRequest attribute), 157

encoding_type (google.cloud.language_v1beta2.types.AnalyzeSyntaxRequest attribute), 158

encoding_type (google.cloud.language_v1beta2.types.AnnotateTextRequest attribute), 158

end_time (google.cloud.speech_v1.types.WordInfo attribute), 257

ended (google.cloud.bigquery.job.CopyJob attribute), 23

ended (google.cloud.bigquery.job.ExtractTableToStorageJob attribute), 27

ended (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 31

ended (google.cloud.bigquery.job.QueryJob attribute), 35

ENGLISH_ISO_639 (in module google.cloud.translate_v2.client), 341

entities (google.cloud.language_v1.types.AnalyzeEntitiesResponse attribute), 148

entities (google.cloud.language_v1.types.AnalyzeEntitySentimentResponse attribute), 148

entities (google.cloud.language_v1.types.AnnotateTextResponse attribute), 149

entities (google.cloud.language_v1beta2.types.AnalyzeEntitiesResponse attribute), 157

entities (google.cloud.language_v1beta2.types.AnalyzeEntitySentimentResponse attribute), 157

entities (google.cloud.language_v1beta2.types.AnnotateTextResponse attribute), 159

Entity (class in google.cloud.datastore), 121

Entity (class in google.cloud.datastore.entity), 102

Entity (class in google.cloud.language_v1.types), 150

Entity (class in google.cloud.language_v1beta2.types), 160

entity() (google.cloud.storage.acl.ACL method), 332

entity_from_dict() (google.cloud.storage.acl.ACL method), 333

entity_from_protobuf() (in module google.cloud.datastore.helpers), 115

entity_id (google.cloud.vision_v1.types.WebDetection attribute), 356

entity_to_protobuf() (in module google.cloud.datastore.helpers), 115

ENTITY_TYPES (google.cloud.bigquery.dataset.AccessGrant attribute), 19

EntityAnnotation (class in google.cloud.vision_v1.types), 351

EntityMention (class in google.cloud.language_v1.types), 151

EntityMention (class in google.cloud.language_v1beta2.types), 160

enums (google.cloud.language_v1.LanguageServiceClient attribute), 147

enums (google.cloud.language_v1beta2.LanguageServiceClient attribute), 157

enums (google.cloud.speech_v1.SpeechClient attribute), 250

enums (google.cloud.vision_v1.ImageAnnotatorClient attribute), 347

environment variable

G_CLOUD_PROJECT, 209

GOOGLE_APPLICATION_CREDENTIALS, 48

GOOGLE_CLOUD_DISABLE_GRPC, 305

GOOGLE_CLOUD_PROJECT, 48, 59, 141, 163, 262, 284, 305, 343

error (google.cloud.operation.Operation attribute), 7

error (google.cloud.speech_v1.types.StreamingRecognizeResponse attribute), 257

error (google.cloud.vision_v1.types.AnnotateImageResponse attribute), 349

error_result (google.cloud.bigquery.job.CopyJob attribute), 23

error_result (google.cloud.bigquery.job.ExtractTableToStorageJob attribute), 27

error_result (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 31

error_result (google.cloud.bigquery.job.QueryJob attribute), 35

- tribute), 35
- errors (google.cloud.bigquery.job.CopyJob attribute), 24
- errors (google.cloud.bigquery.job.ExtractTableToStorageJob attribute), 27
- errors (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 31
- errors (google.cloud.bigquery.job.QueryJob attribute), 35
- errors (google.cloud.bigquery.query.QueryResults attribute), 39
- etag (google.cloud.bigquery.dataset.Dataset attribute), 20
- etag (google.cloud.bigquery.job.CopyJob attribute), 24
- etag (google.cloud.bigquery.job.ExtractTableToStorageJob attribute), 27
- etag (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 31
- etag (google.cloud.bigquery.job.QueryJob attribute), 35
- etag (google.cloud.bigquery.table.Table attribute), 43
- etag (google.cloud.storage.blob.Blob attribute), 314
- etag (google.cloud.storage.bucket.Bucket attribute), 325
- exception() (google.cloud.bigquery.job.CopyJob method), 24
- exception() (google.cloud.bigquery.job.ExtractTableToStorageJob method), 27
- exception() (google.cloud.bigquery.job.LoadTableFromStorageJob method), 31
- exception() (google.cloud.bigquery.job.QueryJob method), 35
- exclude_from_indexes (google.cloud.datastore.entity.Entity attribute), 103
- execute_sql() (google.cloud.spanner.session.Session method), 234
- exists() (google.cloud.bigquery.dataset.Dataset method), 20
- exists() (google.cloud.bigquery.job.CopyJob method), 24
- exists() (google.cloud.bigquery.job.ExtractTableToStorageJob method), 28
- exists() (google.cloud.bigquery.job.LoadTableFromStorageJob method), 31
- exists() (google.cloud.bigquery.job.QueryJob method), 36
- exists() (google.cloud.bigquery.table.Table method), 43
- exists() (google.cloud.dns.changes.Changes method), 136
- exists() (google.cloud.dns.zone.ManagedZone method), 133
- exists() (google.cloud.logging.metric.Metric method), 297
- exists() (google.cloud.logging.sink.Sink method), 299
- exists() (google.cloud.monitoring.group.Group method), 275
- exists() (google.cloud.resource_manager.project.Project method), 200
- exists() (google.cloud.runtimeconfig.config.Config method), 204
- exists() (google.cloud.runtimeconfig.variable.Variable method), 206
- exists() (google.cloud.spanner.database.Database method), 231
- exists() (google.cloud.spanner.instance.Instance method), 228
- exists() (google.cloud.spanner.session.Session method), 235
- exists() (google.cloud.storage.blob.Blob method), 315
- exists() (google.cloud.storage.bucket.Bucket method), 325
- expire_time (google.cloud.pubsub_v1.types.Snapshot attribute), 193
- expires (google.cloud.bigquery.table.Table attribute), 43
- extract_document_sentiment (google.cloud.language_v1.types.AnnotateTextRequest attribute), 149
- extract_document_sentiment (google.cloud.language_v1beta2.types.AnnotateTextRequest attribute), 158
- extract_entities (google.cloud.language_v1.types.AnnotateTextRequest attribute), 149
- extract_entities (google.cloud.language_v1beta2.types.AnnotateTextRequest attribute), 158
- extract_entity_sentiment (google.cloud.language_v1.types.AnnotateTextRequest attribute), 149
- extract_entity_sentiment (google.cloud.language_v1beta2.types.AnnotateTextRequest attribute), 158
- extract_syntax (google.cloud.language_v1.types.AnnotateTextRequest attribute), 149
- extract_syntax (google.cloud.language_v1beta2.types.AnnotateTextRequest attribute), 158
- extract_table_to_storage() (google.cloud.bigquery.client.Client method), 16
- ExtractTableToStorageJob (class in google.cloud.bigquery.job), 26
- face_annotations (google.cloud.vision_v1.types.AnnotateImageResponse attribute), 349
- face_detection() (google.cloud.vision_v1.ImageAnnotatorClient method), 347
- FaceAnnotation (class in google.cloud.vision_v1.types), 351
- FaceAnnotation.Landmark (class in google.cloud.vision_v1.types), 352
- FamilyNameRegexFilter (class in google.cloud.bigtable.row_filters), 90
- fd_bounding_poly (google.cloud.vision_v1.types.FaceAnnotation attribute), 352
- Feature (class in google.cloud.vision_v1.types), 352
- features (google.cloud.language_v1.types.AnnotateTextRequest attribute), 149
- features (google.cloud.language_v1beta2.types.AnnotateTextRequest attribute), 158

- features (google.cloud.vision_v1.types.AnnotateImageRequest attribute), 349
- fetch() (google.cloud.datastore.Query method), 126
- fetch() (google.cloud.datastore.query.Query method), 108
- fetch_data() (google.cloud.bigquery.query.QueryResults method), 39
- fetch_data() (google.cloud.bigquery.table.Table method), 43
- fetch_group() (google.cloud.monitoring.client.Client method), 265
- fetch_metric_descriptor() (google.cloud.monitoring.client.Client method), 266
- fetch_parent() (google.cloud.monitoring.group.Group method), 275
- fetch_project() (google.cloud.resource_manager.client.Client method), 197
- fetch_resource_descriptor() (google.cloud.monitoring.client.Client method), 266
- field_delimiter (google.cloud.bigquery.job.ExtractTableToStorageJob attribute), 28
- field_delimiter (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 31
- field_type (google.cloud.bigquery.schema.SchemaField attribute), 41
- fields (google.cloud.bigquery.schema.SchemaField attribute), 41
- fields (google.cloud.spanner.streamed.StreamedResultSet attribute), 243
- filter (google.cloud.monitoring.query.Query attribute), 279
- filters (google.cloud.datastore.Query attribute), 127
- filters (google.cloud.datastore.query.Query attribute), 109
- finish() (google.cloud.storage.batch.Batch method), 335
- FixedSizePool (class in google.cloud.spanner.pool), 237
- flat_path (google.cloud.datastore.Key attribute), 123
- flat_path (google.cloud.datastore.key.Key attribute), 105
- flatten_results (google.cloud.bigquery.job.QueryJob attribute), 36
- FlowControl (class in google.cloud.pubsub_v1.types), 189
- flush() (google.cloud.logging.handlers.transports.background_thread.BackgroundThreadTransport method), 304
- flush() (google.cloud.logging.handlers.transports.base.Transport method), 304
- form (google.cloud.language_v1.types.PartOfSpeech attribute), 151
- form (google.cloud.language_v1beta2.types.PartOfSpeech attribute), 161
- format() (google.cloud.logging.handlers.container_engine.ContainerEngineHandler method), 303
- friendly_name (google.cloud.bigquery.dataset.Dataset attribute), 20
- friendly_name (google.cloud.bigquery.table.Table attribute), 44
- from_api_repr() (google.cloud.bigquery.client.Project class method), 18
- from_api_repr() (google.cloud.bigquery.dataset.Dataset class method), 20
- from_api_repr() (google.cloud.bigquery.job.CopyJob class method), 24
- from_api_repr() (google.cloud.bigquery.job.ExtractTableToStorageJob class method), 28
- from_api_repr() (google.cloud.bigquery.job.LoadTableFromStorageJob class method), 31
- from_api_repr() (google.cloud.bigquery.job.QueryJob class method), 36
- from_api_repr() (google.cloud.bigquery.schema.SchemaField class method), 41
- from_api_repr() (google.cloud.bigquery.table.Table class method), 44
- from_api_repr() (google.cloud.dns.changes.Changes class method), 137
- from_api_repr() (google.cloud.dns.resource_record_set.ResourceRecordSet class method), 135
- from_api_repr() (google.cloud.dns.zone.ManagedZone class method), 133
- from_api_repr() (google.cloud.iam.Policy class method), 12
- from_api_repr() (google.cloud.logging.metric.Metric class method), 297
- from_api_repr() (google.cloud.logging.sink.Sink class method), 299
- from_api_repr() (google.cloud.resource_manager.project.Project class method), 200
- from_api_repr() (google.cloud.runtimeconfig.variable.Variable class method), 207
- from_dict() (google.cloud.operation.Operation class method), 7
- from_legacy_urlsafes() (google.cloud.datastore.Key class method), 123
- from_legacy_urlsafes() (google.cloud.datastore.key.Key class method), 105
- from_pb() (google.cloud.bigtable.cluster.Cluster class method), 63
- from_pb() (google.cloud.bigtable.instance.Instance class method), 66
- from_pb() (google.cloud.bigtable.row_data.Cell class method), 84
- from_pb() (google.cloud.operation.Operation class method), 8
- from_pb() (google.cloud.spanner.client.InstanceConfig class method), 227
- from_pb() (google.cloud.spanner.database.Database class method), 231
- from_pb() (google.cloud.spanner.instance.Instance class method), 229

[from_query_job\(\)](#) (google.cloud.bigquery.query.QueryResults class method), 39
[from_service_account_json\(\)](#) (google.cloud.client.Client method), 9
[from_service_account_json\(\)](#) (google.cloud.client.ClientWithProject method), 10
[full_matching_images](#) (google.cloud.vision_v1.types.WebDetection attribute), 356
[full_name](#) (google.cloud.logging.logger.Logger attribute), 294
[full_name](#) (google.cloud.logging.metric.Metric attribute), 297
[full_name](#) (google.cloud.logging.sink.Sink attribute), 299
[full_name](#) (google.cloud.resource_manager.project.Project attribute), 200
[full_name](#) (google.cloud.runtimeconfig.config.Config attribute), 204
[full_name](#) (google.cloud.runtimeconfig.variable.Variable attribute), 207
[full_text_annotation](#) (google.cloud.vision_v1.types.AnnotatedImage attribute), 349

G

[GarbageCollectionRule](#) (class in google.cloud.bigtable.column_family), 76
[GCD_DATASET](#) (in module google.cloud.environment_vars), 11
[GCD_HOST](#) (in module google.cloud.environment_vars), 11
[GCLOUD_PROJECT](#), 209
[GCRuleIntersection](#) (class in google.cloud.bigtable.column_family), 75
[GCRuleUnion](#) (class in google.cloud.bigtable.column_family), 75
[gcs_content_uri](#) (google.cloud.language_v1.types.Document attribute), 150
[gcs_content_uri](#) (google.cloud.language_v1beta2.types.Document attribute), 160
[gcs_image_uri](#) (google.cloud.vision_v1.types.ImageSource attribute), 353
[gender](#) (google.cloud.language_v1.types.PartOfSpeech attribute), 151
[gender](#) (google.cloud.language_v1beta2.types.PartOfSpeech attribute), 161
[generate_signed_url\(\)](#) (google.cloud.storage.blob.Blob method), 315
[generate_upload_policy\(\)](#) (google.cloud.storage.bucket.Bucket method), 325
[generation](#) (google.cloud.storage.blob.Blob attribute), 315
[GeoPoint](#) (class in google.cloud.datastore.helpers), 114
[get\(\)](#) (google.cloud.datastore.Client method), 119
[get\(\)](#) (google.cloud.datastore.client.Client method), 100
[get\(\)](#) (google.cloud.spanner.pool.AbstractSessionPool method), 236
[get\(\)](#) (google.cloud.spanner.pool.BurstyPool method), 237
[get\(\)](#) (google.cloud.spanner.pool.FixedSizePool method), 238
[get\(\)](#) (google.cloud.spanner.pool.PingingPool method), 239
[get_blob\(\)](#) (google.cloud.storage.bucket.Bucket method), 326
[get_bucket\(\)](#) (google.cloud.storage.client.Client method), 336
[get_default_handler\(\)](#) (google.cloud.logging.client.Client method), 289
[get_entities\(\)](#) (google.cloud.storage.acl.ACL method), 333
[get_entity\(\)](#) (google.cloud.storage.acl.ACL method), 333
[get_gae_labels\(\)](#) (google.cloud.logging.handlers.app_engine.AppEngineHandler method), 302
[get_image_response\(\)](#) (google.cloud.logging.handlers.app_engine.AppEngineHandler method), 302
[get_iam_policy\(\)](#) (google.cloud.pubsub_v1.publisher.client.Client method), 168
[get_iam_policy\(\)](#) (google.cloud.pubsub_v1.subscriber.client.Client method), 178
[get_iam_policy\(\)](#) (google.cloud.storage.blob.Blob method), 316
[get_iam_policy\(\)](#) (google.cloud.storage.bucket.Bucket method), 326
[get_languages\(\)](#) (google.cloud.translate_v2.client.Client method), 340
[get_logging\(\)](#) (google.cloud.storage.bucket.Bucket method), 326
[get_multi\(\)](#) (google.cloud.datastore.Client method), 119
[get_multi\(\)](#) (google.cloud.datastore.client.Client method), 101
[get_subscription\(\)](#) (google.cloud.pubsub_v1.subscriber.client.Client method), 178
[get_topic\(\)](#) (google.cloud.pubsub_v1.publisher.client.Client method), 168
[get_variable\(\)](#) (google.cloud.runtimeconfig.config.Config method), 204
[GetSubscriptionRequest](#) (class in google.cloud.pubsub_v1.types), 189
[GetTopicRequest](#) (class in google.cloud.pubsub_v1.types), 189
[google.cloud.bigquery.client](#) (module), 15
[google.cloud.bigquery.dataset](#) (module), 18
[google.cloud.bigquery.job](#) (module), 22
[google.cloud.bigquery.query](#) (module), 38
[google.cloud.bigquery.schema](#) (module), 41
[google.cloud.bigquery.table](#) (module), 42
[google.cloud.bigtable.client](#) (module), 60

[google.cloud.bigtable.cluster \(module\)](#), 62
[google.cloud.bigtable.column_family \(module\)](#), 74
[google.cloud.bigtable.instance \(module\)](#), 64
[google.cloud.bigtable.row \(module\)](#), 76
[google.cloud.bigtable.row_data \(module\)](#), 84
[google.cloud.bigtable.row_filters \(module\)](#), 87
[google.cloud.bigtable.table \(module\)](#), 69
[google.cloud.client \(module\)](#), 9
[google.cloud.datastore \(module\)](#), 115
[google.cloud.datastore.batch \(module\)](#), 112
[google.cloud.datastore.client \(module\)](#), 99
[google.cloud.datastore.entity \(module\)](#), 102
[google.cloud.datastore.helpers \(module\)](#), 114
[google.cloud.datastore.key \(module\)](#), 104
[google.cloud.datastore.query \(module\)](#), 107
[google.cloud.datastore.transaction \(module\)](#), 110
[google.cloud.dns.changes \(module\)](#), 136
[google.cloud.dns.client \(module\)](#), 131
[google.cloud.dns.resource_record_set \(module\)](#), 135
[google.cloud.dns.zone \(module\)](#), 132
[google.cloud.environment_vars \(module\)](#), 11
[google.cloud.error_reporting.client \(module\)](#), 259
[google.cloud.error_reporting.util \(module\)](#), 261
[google.cloud.exceptions \(module\)](#), 10
[google.cloud.iam \(module\)](#), 11
[google.cloud.language_v1 \(module\)](#), 144
[google.cloud.language_v1.types \(module\)](#), 147
[google.cloud.language_v1beta2 \(module\)](#), 153
[google.cloud.language_v1beta2.types \(module\)](#), 157
[google.cloud.logging.client \(module\)](#), 289
[google.cloud.logging.entries \(module\)](#), 295
[google.cloud.logging.handlers.app_engine \(module\)](#), 302
[google.cloud.logging.handlers.container_engine \(module\)](#), 303
[google.cloud.logging.handlers.handlers \(module\)](#), 300
[google.cloud.logging.handlers.transports.background_thread \(module\)](#), 304
[google.cloud.logging.handlers.transports.base \(module\)](#), 304
[google.cloud.logging.handlers.transports.sync \(module\)](#), 303
[google.cloud.logging.logger \(module\)](#), 292
[google.cloud.logging.metric \(module\)](#), 296
[google.cloud.logging.sink \(module\)](#), 298
[google.cloud.monitoring.client \(module\)](#), 265
[google.cloud.monitoring.group \(module\)](#), 274
[google.cloud.monitoring.label \(module\)](#), 283
[google.cloud.monitoring.metric \(module\)](#), 271
[google.cloud.monitoring.query \(module\)](#), 277
[google.cloud.monitoring.resource \(module\)](#), 273
[google.cloud.monitoring.timeseries \(module\)](#), 282
[google.cloud.operation \(module\)](#), 7
[google.cloud.pubsub_v1.publisher.client \(module\)](#), 166
[google.cloud.pubsub_v1.subscriber.client \(module\)](#), 174
[google.cloud.pubsub_v1.types \(module\)](#), 188
[google.cloud.resource_manager.client \(module\)](#), 197
[google.cloud.resource_manager.project \(module\)](#), 199
[google.cloud.runtimeconfig \(module\)](#), 208
[google.cloud.runtimeconfig.client \(module\)](#), 203
[google.cloud.runtimeconfig.config \(module\)](#), 204
[google.cloud.runtimeconfig.variable \(module\)](#), 206
[google.cloud.spanner.batch \(module\)](#), 241
[google.cloud.spanner.client \(module\)](#), 224
[google.cloud.spanner.database \(module\)](#), 230
[google.cloud.spanner.instance \(module\)](#), 227
[google.cloud.spanner.keyset \(module\)](#), 240
[google.cloud.spanner.pool \(module\)](#), 236
[google.cloud.spanner.session \(module\)](#), 233
[google.cloud.spanner.snapshot \(module\)](#), 241
[google.cloud.spanner.streamed \(module\)](#), 242
[google.cloud.spanner.transaction \(module\)](#), 242
[google.cloud.speech_v1 \(module\)](#), 250
[google.cloud.speech_v1.types \(module\)](#), 253
[google.cloud.storage.acl \(module\)](#), 331
[google.cloud.storage.batch \(module\)](#), 335
[google.cloud.storage.blob \(module\)](#), 311
[google.cloud.storage.bucket \(module\)](#), 321
[google.cloud.storage.client \(module\)](#), 335
[google.cloud.translate_v2.client \(module\)](#), 339
[google.cloud.vision_v1 \(module\)](#), 345
[google.cloud.vision_v1.types \(module\)](#), 349
[GOOGLE_APPLICATION_CREDENTIALS](#), 48
[GOOGLE_CLOUD_DISABLE_GRPC](#), 305
[GOOGLE_CLOUD_PROJECT](#), 48, 59, 141, 163, 262, 284, 305, 343
[Group \(class in google.cloud.monitoring.group\)](#), 274
[group\(\) \(google.cloud.iam.Policy static method\)](#), 12
[group\(\) \(google.cloud.monitoring.client.Client method\)](#), 266
[group\(\) \(google.cloud.storage.acl.ACL method\)](#), 333
[GrpcRendezvous \(in module google.cloud.exceptions\)](#), 10

H

[has_entity\(\) \(google.cloud.storage.acl.ACL method\)](#), 333
[head_token_index \(google.cloud.language_v1.types.DependencyEdge attribute\)](#), 150
[head_token_index \(google.cloud.language_v1beta2.types.DependencyEdge attribute\)](#), 159
[header\(\) \(google.cloud.monitoring.timeseries.TimeSeries method\)](#), 283
[headwear_likelihood \(google.cloud.vision_v1.types.FaceAnnotation attribute\)](#), 352
[height \(google.cloud.vision_v1.types.Page attribute\)](#), 354
[HTTPContext \(class in google.cloud.error_reporting.client\)](#), 261

I

- `id` (google.cloud.datastore.Key attribute), 124
 - `id` (google.cloud.datastore.key.Key attribute), 105
 - `id` (google.cloud.datastore.Transaction attribute), 129
 - `id` (google.cloud.datastore.transaction.Transaction attribute), 111
 - `id` (google.cloud.monitoring.group.Group attribute), 275
 - `id` (google.cloud.storage.blob.Blob attribute), 316
 - `id` (google.cloud.storage.bucket.Bucket attribute), 327
 - `id_or_name` (google.cloud.datastore.Key attribute), 124
 - `id_or_name` (google.cloud.datastore.key.Key attribute), 105
 - `ignore_unknown_values` (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 32
 - `Image` (class in google.cloud.vision_v1.types), 353
 - `image` (google.cloud.vision_v1.types.AnnotateImageRequest attribute), 349
 - `image_context` (google.cloud.vision_v1.types.AnnotateImageRequest attribute), 349
 - `image_properties()` (google.cloud.vision_v1.ImageAnnotatorClient method), 347
 - `image_properties_annotation` (google.cloud.vision_v1.types.AnnotateImageResponse attribute), 349
 - `image_uri` (google.cloud.vision_v1.types.ImageSource attribute), 353
 - `ImageAnnotatorClient` (class in google.cloud.vision_v1), 345
 - `ImageContext` (class in google.cloud.vision_v1.types), 353
 - `ImageProperties` (class in google.cloud.vision_v1.types), 353
 - `ImageSource` (class in google.cloud.vision_v1.types), 353
 - `importance_fraction` (google.cloud.vision_v1.types.CropHints attribute), 350
 - `increment_cell_value()` (google.cloud.bigtable.row.AppendRow method), 78
 - `input_file_bytes` (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 32
 - `input_files` (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 32
 - `insert_data()` (google.cloud.bigquery.table.Table method), 44
 - `Instance` (class in google.cloud.bigtable.instance), 64
 - `Instance` (class in google.cloud.spanner.instance), 227
 - `instance()` (google.cloud.bigtable.client.Client method), 61
 - `instance()` (google.cloud.spanner.client.Client method), 225
 - `instance_admin_api` (google.cloud.spanner.client.Client attribute), 226
 - `INSTANCE_ADMIN_HOST` (in module google.cloud.bigtable.client), 62
 - `InstanceConfig` (class in google.cloud.spanner.client), 226
 - `interim_results` (google.cloud.speech_v1.types.StreamingRecognitionConfig attribute), 255
 - `InvalidChunk`, 84
 - `InvalidReadRowsResponse`, 84
 - `is_final` (google.cloud.speech_v1.types.StreamingRecognitionResult attribute), 255
 - `is_nullable` (google.cloud.bigquery.schema.SchemaField attribute), 42
 - `is_partial` (google.cloud.datastore.Key attribute), 124
 - `is_partial` (google.cloud.datastore.key.Key attribute), 105
 - `is_prefix` (google.cloud.vision_v1.types.TextAnnotation attribute), 355
 - `iter()` (google.cloud.monitoring.query.Query method), 279
 - `Iterator` (class in google.cloud.datastore.query), 107
- ## J
- `job` (google.cloud.bigquery.query.QueryResults attribute), 39
 - `job_from_resource()` (google.cloud.bigquery.client.Client method), 16
 - `job_type` (google.cloud.bigquery.job.CopyJob attribute), 24
 - `job_type` (google.cloud.bigquery.job.ExtractTableToStorageJob attribute), 28
 - `job_type` (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 32
 - `job_type` (google.cloud.bigquery.job.QueryJob attribute), 36
 - `joy_likelihood` (google.cloud.vision_v1.types.FaceAnnotation attribute), 352
- ## K
- `Key` (class in google.cloud.datastore), 122
 - `Key` (class in google.cloud.datastore.key), 104
 - `key()` (google.cloud.datastore.Client method), 120
 - `key()` (google.cloud.datastore.client.Client method), 101
 - `key_from_job` (google.cloud.datastore.Query method), 127
 - `key_filter()` (google.cloud.datastore.query.Query method), 109
 - `key_from_protobuf()` (in module google.cloud.datastore.helpers), 115
 - `KeyRange` (class in google.cloud.spanner.keyset), 240
 - `keys_only()` (google.cloud.datastore.Query method), 127
 - `keys_only()` (google.cloud.datastore.query.Query method), 109
 - `KeySet` (class in google.cloud.spanner.keyset), 240
 - `kind` (google.cloud.datastore.Entity attribute), 122
 - `kind` (google.cloud.datastore.entity.Entity attribute), 104
 - `kind` (google.cloud.datastore.Key attribute), 124
 - `kind` (google.cloud.datastore.key.Key attribute), 105
 - `kind` (google.cloud.datastore.Query attribute), 127
 - `kind` (google.cloud.datastore.query.Query attribute), 109

L

- label (google.cloud.language_v1.types.DependencyEdge attribute), 150
- label (google.cloud.language_v1beta2.types.DependencyEdge attribute), 160
- label_annotations (google.cloud.vision_v1.types.AnnotateImageResponse attribute), 349
- label_detection() (google.cloud.vision_v1.ImageAnnotatorClient method), 347
- LabelDescriptor (class in google.cloud.monitoring.label), 283
- labels (google.cloud.monitoring.timeseries.TimeSeries attribute), 283
- labels (google.cloud.pubsub_v1.types.Snapshot attribute), 193
- labels (google.cloud.pubsub_v1.types.Subscription attribute), 195
- labels (google.cloud.pubsub_v1.types.Topic attribute), 195
- labels (google.cloud.storage.bucket.Bucket attribute), 327
- LabelValueType (class in google.cloud.monitoring.label), 284
- landmark_annotations (google.cloud.vision_v1.types.AnnotateImageResponse attribute), 349
- landmark_detection() (google.cloud.vision_v1.ImageAnnotatorClient method), 347
- landmarking_confidence (google.cloud.vision_v1.types.FaceAnnotation attribute), 352
- landmarks (google.cloud.vision_v1.types.FaceAnnotation attribute), 352
- language (google.cloud.language_v1.types.AnalyzeEntitiesResponse attribute), 148
- language (google.cloud.language_v1.types.AnalyzeEntitySentimentResponse attribute), 148
- language (google.cloud.language_v1.types.AnalyzeSentimentResponse attribute), 148
- language (google.cloud.language_v1.types.AnalyzeSyntaxResponse attribute), 149
- language (google.cloud.language_v1.types.AnnotateTextResponse attribute), 150
- language (google.cloud.language_v1.types.Document attribute), 150
- language (google.cloud.language_v1beta2.types.AnalyzeEntitiesResponse attribute), 157
- language (google.cloud.language_v1beta2.types.AnalyzeEntitySentimentResponse attribute), 157
- language (google.cloud.language_v1beta2.types.AnalyzeSentimentResponse attribute), 158
- language (google.cloud.language_v1beta2.types.AnalyzeSyntaxResponse attribute), 158
- language (google.cloud.language_v1beta2.types.AnnotateTextResponse attribute), 159
- language (google.cloud.language_v1beta2.types.Document attribute), 160
- language_code (google.cloud.speech_v1.types.RecognitionConfig attribute), 254
- language_code (google.cloud.vision_v1.types.TextAnnotation attribute), 355
- language_hints (google.cloud.vision_v1.types.ImageContext attribute), 353
- LanguageServiceClient (class in google.cloud.language_v1), 144
- LanguageServiceClient (class in google.cloud.language_v1beta2), 153
- last_update_time (google.cloud.speech_v1.types.LongRunningRecognizeResponse attribute), 253
- lat_lng (google.cloud.vision_v1.types.LocationInfo attribute), 354
- lat_long_rect (google.cloud.vision_v1.types.ImageContext attribute), 353
- LatLngRect (class in google.cloud.vision_v1.types), 353
- lemma (google.cloud.language_v1.types.Token attribute), 152
- lemma (google.cloud.language_v1beta2.types.Token attribute), 162
- lifecycle_rules (google.cloud.storage.bucket.Bucket attribute), 327
- list_ancestors() (google.cloud.monitoring.group.Group method), 275
- list_blobs() (google.cloud.storage.bucket.Bucket method), 327
- list_buckets() (google.cloud.storage.client.Client method), 337
- list_changes() (google.cloud.dns.zone.ManagedZone method), 133
- list_children() (google.cloud.monitoring.group.Group method), 275
- list_clusters() (google.cloud.bigtable.instance.Instance method), 66
- list_column_families() (google.cloud.bigtable.table.Table method), 70
- list_databases() (google.cloud.spanner.instance.Instance method), 229
- list_datasets() (google.cloud.bigquery.client.Client method), 16
- list_descendants() (google.cloud.monitoring.group.Group method), 275
- list_entries() (google.cloud.logging.client.Client method), 290
- list_entries() (google.cloud.logging.logger.Logger method), 294
- list_groups() (google.cloud.monitoring.client.Client method), 267
- list_instance_configs() (google.cloud.spanner.client.Client method), 226
- list_instances() (google.cloud.bigtable.client.Client method), 61
- list_instances() (google.cloud.spanner.client.Client method), 61

- method), 226
- list_jobs() (google.cloud.bigquery.client.Client method), 17
- list_members() (google.cloud.monitoring.group.Group method), 276
- list_metric_descriptors() (google.cloud.monitoring.client.Client method), 267
- list_metrics() (google.cloud.logging.client.Client method), 290
- list_partitions() (google.cloud.bigquery.table.Table method), 44
- list_projects() (google.cloud.bigquery.client.Client method), 17
- list_projects() (google.cloud.resource_manager.client.Client method), 198
- list_resource_descriptors() (google.cloud.monitoring.client.Client method), 267
- list_resource_record_sets() (google.cloud.dns.zone.ManagedZone method), 134
- list_sinks() (google.cloud.logging.client.Client method), 290
- list_snapshots() (google.cloud.pubsub_v1.subscriber.client.Client method), 179
- list_subscriptions() (google.cloud.pubsub_v1.subscriber.client.Client method), 180
- list_tables() (google.cloud.bigquery.dataset.Dataset method), 20
- list_tables() (google.cloud.bigtable.instance.Instance method), 66
- list_topic_subscriptions() (google.cloud.pubsub_v1.publisher.client.Client method), 169
- list_topics() (google.cloud.pubsub_v1.publisher.client.Client method), 169
- list_variables() (google.cloud.runtimeconfig.config.Config method), 205
- list_zones() (google.cloud.dns.client.Client method), 131
- ListSnapshotsRequest (class in google.cloud.pubsub_v1.types), 189
- ListSnapshotsResponse (class in google.cloud.pubsub_v1.types), 189
- ListSubscriptionsRequest (class in google.cloud.pubsub_v1.types), 189
- ListSubscriptionsResponse (class in google.cloud.pubsub_v1.types), 190
- ListTopicsRequest (class in google.cloud.pubsub_v1.types), 190
- ListTopicsResponse (class in google.cloud.pubsub_v1.types), 190
- ListTopicSubscriptionsRequest (class in google.cloud.pubsub_v1.types), 190
- ListTopicSubscriptionsResponse (class in google.cloud.pubsub_v1.types), 190
- load_table_from_storage() (google.cloud.bigquery.client.Client method), 17
- LoadTableFromStorageJob (class in google.cloud.bigquery.job), 29
- locale (google.cloud.vision_v1.types.EntityAnnotation attribute), 351
- location (google.cloud.bigquery.dataset.Dataset attribute), 21
- location (google.cloud.bigquery.table.Table attribute), 45
- location (google.cloud.storage.bucket.Bucket attribute), 328
- LocationInfo (class in google.cloud.vision_v1.types), 354
- locations (google.cloud.vision_v1.types.EntityAnnotation attribute), 351
- log_proto() (google.cloud.logging.logger.Batch method), 292
- log_proto() (google.cloud.logging.logger.Logger method), 294
- log_struct() (google.cloud.logging.logger.Batch method), 292
- log_struct() (google.cloud.logging.logger.Logger method), 294
- log_text() (google.cloud.logging.logger.Batch method), 293
- log_text() (google.cloud.logging.logger.Logger method), 295
- Logger (class in google.cloud.logging.logger), 293
- logger() (google.cloud.logging.client.Client method), 290
- logger_name_from_path() (in module google.cloud.logging.entries), 296
- logging_api (google.cloud.logging.client.Client attribute), 291
- logo_annotations (google.cloud.vision_v1.types.AnnotateImageResponse attribute), 349
- logo_detection() (google.cloud.vision_v1.ImageAnnotatorClient method), 348
- long_running_recognize() (google.cloud.speech_v1.SpeechClient method), 250
- LongRunningRecognizeMetadata (class in google.cloud.speech_v1.types), 253
- LongRunningRecognizeRequest (class in google.cloud.speech_v1.types), 253
- LongRunningRecognizeResponse (class in google.cloud.speech_v1.types), 253
- lookup_bucket() (google.cloud.storage.client.Client method), 337

M

magnitude (google.cloud.language_v1beta2.types.Sentiment attribute), 162
 make_public() (google.cloud.storage.blob.Blob method), 316
 make_public() (google.cloud.storage.bucket.Bucket method), 328
 ManagedZone (class in google.cloud.dns.zone), 132
 match_project_from_project_name() (google.cloud.pubsub_v1.publisher.client.Client method), 170
 match_project_from_project_name() (google.cloud.pubsub_v1.subscriber.client.Client method), 181
 match_project_from_snapshot_name() (google.cloud.pubsub_v1.subscriber.client.Client method), 181
 match_project_from_subscription_name() (google.cloud.pubsub_v1.subscriber.client.Client method), 181
 match_project_from_topic_name() (google.cloud.pubsub_v1.publisher.client.Client method), 170
 match_project_from_topic_name() (google.cloud.pubsub_v1.subscriber.client.Client method), 181
 match_snapshot_from_snapshot_name() (google.cloud.pubsub_v1.subscriber.client.Client method), 181
 match_subscription_from_subscription_name() (google.cloud.pubsub_v1.subscriber.client.Client method), 181
 match_topic_from_topic_name() (google.cloud.pubsub_v1.publisher.client.Client method), 170
 match_topic_from_topic_name() (google.cloud.pubsub_v1.subscriber.client.Client method), 181
 max_alternatives (google.cloud.speech_v1.types.RecognitionConfig attribute), 254
 max_bad_records (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 32
 max_bytes (google.cloud.pubsub_v1.types.BatchSettings attribute), 188
 max_bytes (google.cloud.pubsub_v1.types.FlowControl attribute), 189
 max_lat_lng (google.cloud.vision_v1.types.LatLongRect attribute), 353
 max_latency (google.cloud.pubsub_v1.types.BatchSettings attribute), 188
 max_messages (google.cloud.pubsub_v1.types.BatchSettings attribute), 188
 max_messages (google.cloud.pubsub_v1.types.FlowControl attribute), 189
 max_messages (google.cloud.pubsub_v1.types.PullRequest attribute), 192
 MAX_MUTATIONS (in google.cloud.bigtable.row module), 83
 max_results (google.cloud.bigquery.query.QueryResults attribute), 39
 max_results (google.cloud.vision_v1.types.Feature attribute), 353
 MaxAgeGCRule (class in google.cloud.bigtable.column_family), 76
 maximum_billing_tier (google.cloud.bigquery.job.QueryJob attribute), 36
 maximum_bytes_billed (google.cloud.bigquery.job.QueryJob attribute), 36
 MaxVersionsGCRule (class in google.cloud.bigtable.column_family), 76
 md5_hash (google.cloud.storage.blob.Blob attribute), 316
 media_link (google.cloud.storage.blob.Blob attribute), 316
 medical (google.cloud.vision_v1.types.SafeSearchAnnotation attribute), 355
 mentions (google.cloud.language_v1.types.Entity attribute), 151
 mentions (google.cloud.language_v1beta2.types.Entity attribute), 160
 Message (class in google.cloud.pubsub_v1.subscriber.message), 187
 message (google.cloud.pubsub_v1.types.ReceivedMessage attribute), 192
 message_id (google.cloud.pubsub_v1.types.PubsubMessage attribute), 191
 message_id (Message attribute), 187
 message_ids (google.cloud.pubsub_v1.types.PublishResponse attribute), 191
 message_retention_duration (google.cloud.pubsub_v1.types.Subscription attribute), 194
 messages (google.cloud.pubsub_v1.types.PublishRequest attribute), 191
 metadata (google.cloud.language_v1.types.Entity attribute), 150
 metadata (google.cloud.language_v1beta2.types.Entity attribute), 160
 metadata (google.cloud.operation.Operation attribute), 8
 metadata (google.cloud.spanner.streamed.StreamedResultSet attribute), 243
 metadata (google.cloud.storage.blob.Blob attribute), 316
 metageneration (google.cloud.storage.blob.Blob attribute), 316
 metageneration (google.cloud.storage.bucket.Bucket attribute), 328
 Metric (class in google.cloud.logging.metric), 296
 Metric (class in google.cloud.monitoring.metric), 271
 metric() (google.cloud.logging.client.Client method), 291
 metric() (google.cloud.monitoring.client.Client static

method), 268

metric_descriptor() (google.cloud.monitoring.client.Client method), 268

METRIC_KIND_UNSPECIFIED (google.cloud.monitoring.metric.MetricKind attribute), 273

metric_type (google.cloud.monitoring.query.Query attribute), 279

MetricDescriptor (class in google.cloud.monitoring.metric), 272

MetricKind (class in google.cloud.monitoring.metric), 273

metrics_api (google.cloud.logging.client.Client attribute), 291

mid (google.cloud.vision_v1.types.EntityAnnotation attribute), 351

MIMEApplicationHTTP (class in google.cloud.storage.batch), 335

min_lat_lng (google.cloud.vision_v1.types.LatLongRect attribute), 353

mode (google.cloud.bigquery.schema.SchemaField attribute), 42

modified (google.cloud.bigquery.dataset.Dataset attribute), 21

modified (google.cloud.bigquery.table.Table attribute), 45

modify_ack_deadline() (google.cloud.pubsub_v1.subscriber.client.Client method), 181

modify_deadline_ack_ids (google.cloud.pubsub_v1.types.StreamingPullRequest attribute), 194

modify_deadline_seconds (google.cloud.pubsub_v1.types.StreamingPullRequest attribute), 193

modify_push_config() (google.cloud.pubsub_v1.subscriber.client.Client method), 182

ModifyAckDeadlineRequest (class in google.cloud.pubsub_v1.types), 191

ModifyPushConfigRequest (class in google.cloud.pubsub_v1.types), 191

mood (google.cloud.language_v1.types.PartOfSpeech attribute), 151

mood (google.cloud.language_v1beta2.types.PartOfSpeech attribute), 161

mutate_rows() (google.cloud.bigtable.table.Table method), 70

mutations (google.cloud.datastore.Batch attribute), 117

mutations (google.cloud.datastore.batch.Batch attribute), 113

mutations (google.cloud.datastore.transaction.Transaction attribute), 111

name (google.cloud.bigquery.query.QueryResults attribute), 40

name (google.cloud.bigquery.schema.SchemaField attribute), 42

name (google.cloud.bigtable.cluster.Cluster attribute), 63

name (google.cloud.bigtable.column_family.ColumnFamily attribute), 75

name (google.cloud.bigtable.instance.Instance attribute), 66

name (google.cloud.bigtable.table.Table attribute), 70

name (google.cloud.datastore.Key attribute), 124

name (google.cloud.datastore.key.Key attribute), 105

name (google.cloud.dns.changes.Changes attribute), 137

name (google.cloud.language_v1.types.Entity attribute), 150

name (google.cloud.language_v1beta2.types.ClassificationCategory attribute), 159

name (google.cloud.language_v1beta2.types.Entity attribute), 160

name (google.cloud.monitoring.group.Group attribute), 276

name (google.cloud.pubsub_v1.types.CreateSnapshotRequest attribute), 188

name (google.cloud.pubsub_v1.types.Snapshot attribute), 193

name (google.cloud.pubsub_v1.types.Subscription attribute), 194

name (google.cloud.pubsub_v1.types.Topic attribute), 195

name (google.cloud.spanner.database.Database attribute), 232

name (google.cloud.spanner.instance.Instance attribute), 229

name (google.cloud.spanner.session.Session attribute), 235

name (google.cloud.vision_v1.types.Property attribute), 354

name_server_set (google.cloud.dns.zone.ManagedZone attribute), 134

name_servers (google.cloud.dns.zone.ManagedZone attribute), 134

namespace (google.cloud.datastore.Batch attribute), 117

namespace (google.cloud.datastore.batch.Batch attribute), 114

namespace (google.cloud.datastore.Key attribute), 124

namespace (google.cloud.datastore.key.Key attribute), 106

namespace (google.cloud.datastore.Query attribute), 127

namespace (google.cloud.datastore.query.Query attribute), 109

namespace (google.cloud.datastore.transaction.Transaction attribute), 112

new_project() (google.cloud.resource_manager.client.Client method), 198

N

nack() (google.cloud.pubsub_v1.subscriber.message.Message method), 188

- next_page_token (google.cloud.pubsub_v1.types.ListSnapshotsRequest attribute), 189
- next_page_token (google.cloud.pubsub_v1.types.ListTopicSubscriptionsRequest attribute), 190
- next_page_token (google.cloud.pubsub_v1.types.ListTopicsRequest attribute), 191
- next_page_token (google.cloud.pubsub_v1.types.ListTopicSubscriptionsRequest attribute), 190
- NMT (in module google.cloud.translate_v2.client), 341
- null_marker (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 32
- num_bytes (google.cloud.bigquery.table.Table attribute), 45
- num_dml_affected_rows (google.cloud.bigquery.query.QueryResults attribute), 40
- num_rows (google.cloud.bigquery.table.Table attribute), 45
- number (google.cloud.language_v1.types.PartOfSpeech attribute), 151
- number (google.cloud.language_v1beta2.types.PartOfSpeech attribute), 161
- O**
- ObjectACL (class in google.cloud.storage.acl), 334
- one() (google.cloud.spanner.streamed.StreamedResultSet method), 243
- one_or_none() (google.cloud.spanner.streamed.StreamedResultSet method), 243
- open() (google.cloud.pubsub_v1.subscriber.policy.thread.Policy method), 186
- Operation (class in google.cloud.operation), 7
- OPERATORS (google.cloud.datastore.query.Query attribute), 107
- order (google.cloud.datastore.Query attribute), 127
- order (google.cloud.datastore.query.Query attribute), 109
- output_bytes (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 32
- output_rows (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 32
- owner (google.cloud.storage.blob.Blob attribute), 317
- owner (google.cloud.storage.bucket.Bucket attribute), 328
- OWNER_ROLE (in module google.cloud.iam), 11
- owners (google.cloud.iam.Policy attribute), 12
- P**
- Page (class in google.cloud.vision_v1.types), 354
- page_size (google.cloud.pubsub_v1.types.ListSnapshotsRequest attribute), 189
- page_size (google.cloud.pubsub_v1.types.ListSubscriptionsRequest attribute), 190
- page_size (google.cloud.pubsub_v1.types.ListTopicsRequest attribute), 190
- page_token (google.cloud.pubsub_v1.types.ListTopicsRequest attribute), 190
- page_token (google.cloud.pubsub_v1.types.ListTopicSubscriptionsRequest attribute), 190
- pages (google.cloud.vision_v1.types.TextAnnotation attribute), 355
- pages_with_matching_images (google.cloud.vision_v1.types.WebDetection attribute), 356
- pan_angle (google.cloud.vision_v1.types.FaceAnnotation attribute), 352
- Paragraph (class in google.cloud.vision_v1.types), 354
- paragraphs (google.cloud.vision_v1.types.Block attribute), 350
- parent (google.cloud.datastore.Key attribute), 124
- parent (google.cloud.datastore.key.Key attribute), 106
- parent_name (google.cloud.monitoring.group.Group attribute), 276
- parse_message() (google.cloud.logging.entries.ProtoBufEntry method), 296
- part_of_speech (google.cloud.language_v1.types.Token attribute), 152
- part_of_speech (google.cloud.language_v1beta2.types.Token attribute), 162
- partial_matching_images (google.cloud.vision_v1.types.WebDetection attribute), 356
- PartialCellData (class in google.cloud.bigtable.row_data), 84
- PartialRowData (class in google.cloud.bigtable.row_data), 84
- PartialRowsData (class in google.cloud.bigtable.row_data), 85
- partition_expiration (google.cloud.bigquery.table.Table attribute), 45
- partitioning_type (google.cloud.bigquery.table.Table attribute), 45
- PartOfSpeech (class in google.cloud.language_v1.types), 151
- PartOfSpeech (class in google.cloud.language_v1beta2.types), 161
- PassAllFilter (class in google.cloud.bigtable.row_filters), 90
- patch() (google.cloud.bigquery.dataset.Dataset method), 21
- patch() (google.cloud.bigquery.table.Table method), 45

- patch() (google.cloud.storage.blob.Blob method), 317
- patch() (google.cloud.storage.bucket.Bucket method), 329
- path (google.cloud.bigquery.dataset.Dataset attribute), 21
- path (google.cloud.bigquery.job.CopyJob attribute), 24
- path (google.cloud.bigquery.job.ExtractTableToStorageJob attribute), 28
- path (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 32
- path (google.cloud.bigquery.job.QueryJob attribute), 36
- path (google.cloud.bigquery.table.Table attribute), 46
- path (google.cloud.datastore.Key attribute), 124
- path (google.cloud.datastore.key.Key attribute), 106
- path (google.cloud.dns.changes.Changes attribute), 137
- path (google.cloud.dns.zone.ManagedZone attribute), 134
- path (google.cloud.logging.logger.Logger attribute), 295
- path (google.cloud.logging.metric.Metric attribute), 297
- path (google.cloud.logging.sink.Sink attribute), 299
- path (google.cloud.monitoring.group.Group attribute), 276
- path (google.cloud.resource_manager.project.Project attribute), 200
- path (google.cloud.runtimeconfig.config.Config attribute), 205
- path (google.cloud.runtimeconfig.variable.Variable attribute), 207
- path (google.cloud.storage.blob.Blob attribute), 317
- path (google.cloud.storage.bucket.Bucket attribute), 329
- path_helper() (google.cloud.storage.blob.Blob static method), 317
- path_helper() (google.cloud.storage.bucket.Bucket static method), 329
- person (google.cloud.language_v1.types.PartOfSpeech attribute), 151
- person (google.cloud.language_v1beta2.types.PartOfSpeech attribute), 161
- phrases (google.cloud.speech_v1.types.SpeechContext attribute), 254
- ping() (google.cloud.spanner.pool.PingingPool method), 239
- PingingPool (class in google.cloud.spanner.pool), 238
- pixel_fraction (google.cloud.vision_v1.types.ColorInfo attribute), 350
- Point (class in google.cloud.monitoring.timeseries), 282
- Policy (class in google.cloud.iam), 11
- Policy (class in google.cloud.pubsub_v1.subscriber.policy.thread), 186
- poll() (google.cloud.operation.Operation method), 8
- Position (class in google.cloud.vision_v1.types), 354
- position (google.cloud.vision_v1.types.FaceAnnotation attribute), 351
- PREDEFINED_JSON_ACLS (google.cloud.storage.acl.ACL attribute), 332
- preserve_nulls (google.cloud.bigquery.query.QueryResults attribute), 40
- print_header (google.cloud.bigquery.job.ExtractTableToStorageJob attribute), 28
- priority (google.cloud.bigquery.job.QueryJob attribute), 36
- profanity_filter (google.cloud.speech_v1.types.RecognitionConfig attribute), 254
- progress_percent (google.cloud.speech_v1.types.LongRunningRecognizeM attribute), 253
- Project (class in google.cloud.bigquery.client), 18
- Project (class in google.cloud.resource_manager.project), 199
- project (google.cloud.bigquery.dataset.Dataset attribute), 21
- project (google.cloud.bigquery.job.CopyJob attribute), 24
- project (google.cloud.bigquery.job.ExtractTableToStorageJob attribute), 28
- project (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 32
- project (google.cloud.bigquery.job.QueryJob attribute), 36
- project (google.cloud.bigquery.query.QueryResults attribute), 40
- project (google.cloud.bigquery.table.Table attribute), 46
- project (google.cloud.datastore.Batch attribute), 117
- project (google.cloud.datastore.batch.Batch attribute), 114
- project (google.cloud.datastore.Key attribute), 124
- project (google.cloud.datastore.key.Key attribute), 106
- project (google.cloud.datastore.Query attribute), 127
- project (google.cloud.datastore.query.Query attribute), 109
- project (google.cloud.datastore.transaction.Transaction attribute), 112
- project (google.cloud.dns.zone.ManagedZone attribute), 134
- project (google.cloud.logging.logger.Logger attribute), 295
- project (google.cloud.logging.metric.Metric attribute), 298
- project (google.cloud.logging.sink.Sink attribute), 299
- project (google.cloud.pubsub_v1.types.ListSnapshotsRequest attribute), 189
- project (google.cloud.pubsub_v1.types.ListSubscriptionsRequest attribute), 190
- project (google.cloud.pubsub_v1.types.ListTopicsRequest attribute), 190
- project (google.cloud.runtimeconfig.config.Config attribute), 205
- project_name (google.cloud.bigtable.client.Client attribute), 61
- project_name (google.cloud.spanner.client.Client attribute), 332

- tribute), 226
 - project_number (google.cloud.storage.bucket.Bucket attribute), 329
 - project_path() (google.cloud.pubsub_v1.publisher.client.Client method), 171
 - project_path() (google.cloud.pubsub_v1.subscriber.client.Client method), 183
 - projection (google.cloud.datastore.Query attribute), 127
 - projection (google.cloud.datastore.query.Query attribute), 109
 - proper (google.cloud.language_v1.types.PartOfSpeech attribute), 151
 - proper (google.cloud.language_v1beta2.types.PartOfSpeech attribute), 161
 - properties (google.cloud.vision_v1.types.EntityAnnotation attribute), 351
 - Property (class in google.cloud.vision_v1.types), 354
 - property (google.cloud.vision_v1.types.Block attribute), 350
 - property (google.cloud.vision_v1.types.Page attribute), 354
 - property (google.cloud.vision_v1.types.Paragraph attribute), 354
 - property (google.cloud.vision_v1.types.Symbol attribute), 355
 - property (google.cloud.vision_v1.types.Word attribute), 356
 - ProtobufEntry (class in google.cloud.logging.entries), 295
 - public_url (google.cloud.storage.blob.Blob attribute), 317
 - publish() (google.cloud.pubsub_v1.publisher.client.Client method), 171
 - publish_time (google.cloud.pubsub_v1.subscriber.message.Message attribute), 188
 - publish_time (google.cloud.pubsub_v1.types.PubsubMessage attribute), 192
 - publish_time (Message attribute), 187
 - PublishRequest (class in google.cloud.pubsub_v1.types), 191
 - PublishResponse (class in google.cloud.pubsub_v1.types), 191
 - PUBSUB_EMULATOR (in module google.cloud.environment_vars), 11
 - PubsubMessage (class in google.cloud.pubsub_v1.types), 191
 - PullRequest (class in google.cloud.pubsub_v1.types), 192
 - PullResponse (class in google.cloud.pubsub_v1.types), 192
 - push_config (google.cloud.pubsub_v1.types.ModifyPushConfig attribute), 191
 - push_config (google.cloud.pubsub_v1.types.Subscription attribute), 194
 - push_endpoint (google.cloud.pubsub_v1.types.PushConfig attribute), 192
 - PushConfig (class in google.cloud.pubsub_v1.types), 192
 - put() (google.cloud.datastore.Batch method), 117
 - put() (google.cloud.datastore.batch.Batch method), 114
 - put() (google.cloud.datastore.Client method), 120
 - put() (google.cloud.datastore.client.Client method), 101
 - put() (google.cloud.datastore.transaction.Transaction method), 112
 - put() (google.cloud.spanner.pool.AbstractSessionPool method), 237
 - put() (google.cloud.spanner.pool.BurstyPool method), 237
 - put() (google.cloud.spanner.pool.FixedSizePool method), 238
 - put() (google.cloud.spanner.pool.PingingPool method), 239
 - put() (google.cloud.spanner.pool.TransactionPingingPool method), 240
 - put_multi() (google.cloud.datastore.Client method), 120
 - put_multi() (google.cloud.datastore.client.Client method), 101
- ## Q
- Query (class in google.cloud.datastore), 125
 - Query (class in google.cloud.datastore.query), 107
 - Query (class in google.cloud.monitoring.query), 277
 - query() (google.cloud.datastore.Client method), 120
 - query() (google.cloud.datastore.client.Client method), 101
 - query() (google.cloud.monitoring.client.Client method), 269
 - query_results() (google.cloud.bigquery.job.QueryJob method), 36
 - QueryResults (class in google.cloud.bigquery.job), 34
 - QueryPriority (class in google.cloud.bigquery.job), 38
 - QueryResults (class in google.cloud.bigquery.query), 38
 - quotas() (google.cloud.dns.client.Client method), 132
 - quote_character (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 32
- ## R
- read() (google.cloud.spanner.session.Session method), 235
 - READ_ONLY_SCOPE (in module google.cloud.bigtable.client), 62
 - read_row() (google.cloud.bigtable.table.Table method), 70
 - read_rows() (google.cloud.bigtable.table.Table method), 71
 - received_messages (google.cloud.pubsub_v1.types.PullResponse attribute), 192
 - received_messages (google.cloud.pubsub_v1.types.StreamingPullResponse attribute), 194
 - ReceivedMessage (class in google.cloud.pubsub_v1.types), 192

- reciprocity (google.cloud.language_v1.types.PartOfSpeech attribute), 151
- reciprocity (google.cloud.language_v1beta2.types.PartOfSpeech attribute), 161
- RecognitionAudio (class in google.cloud.speech_v1.types), 253
- RecognitionConfig (class in google.cloud.speech_v1.types), 254
- recognize() (google.cloud.speech_v1.SpeechClient method), 251
- RecognizeRequest (class in google.cloud.speech_v1.types), 254
- RecognizeResponse (class in google.cloud.speech_v1.types), 254
- reduce() (google.cloud.monitoring.query.Query method), 279
- Reducer (class in google.cloud.monitoring.query), 282
- register_type() (in module google.cloud.operation), 8
- reload() (google.cloud.bigquery.dataset.Dataset method), 21
- reload() (google.cloud.bigquery.job.CopyJob method), 25
- reload() (google.cloud.bigquery.job.ExtractTableToStorageJob method), 28
- reload() (google.cloud.bigquery.job.LoadTableFromStorageJob method), 33
- reload() (google.cloud.bigquery.job.QueryJob method), 37
- reload() (google.cloud.bigquery.table.Table method), 46
- reload() (google.cloud.bigtable.cluster.Cluster method), 64
- reload() (google.cloud.bigtable.instance.Instance method), 67
- reload() (google.cloud.dns.changes.Changes method), 137
- reload() (google.cloud.dns.zone.ManagedZone method), 134
- reload() (google.cloud.logging.metric.Metric method), 298
- reload() (google.cloud.logging.sink.Sink method), 299
- reload() (google.cloud.monitoring.group.Group method), 277
- reload() (google.cloud.resource_manager.project.Project method), 200
- reload() (google.cloud.runtimeconfig.config.Config method), 205
- reload() (google.cloud.runtimeconfig.variable.Variable method), 207
- reload() (google.cloud.spanner.database.Database method), 232
- reload() (google.cloud.spanner.instance.Instance method), 229
- reload() (google.cloud.storage.acl.ACL method), 333
- reload() (google.cloud.storage.blob.Blob method), 317
- reload() (google.cloud.storage.bucket.Bucket method), 329
- reload_path (google.cloud.storage.acl.BucketACL attribute), 334
- reload_path (google.cloud.storage.acl.ObjectACL attribute), 334
- rename_blob() (google.cloud.storage.bucket.Bucket method), 329
- report() (google.cloud.error_reporting.client.Client method), 260
- report_errors_api (google.cloud.error_reporting.client.Client attribute), 260
- report_exception() (google.cloud.error_reporting.client.Client method), 260
- requests (google.cloud.vision_v1.types.BatchAnnotateImagesRequest attribute), 349
- reset() (google.cloud.storage.acl.ACL method), 333
- Resource (class in google.cloud.monitoring.resource), 273
- resource() (google.cloud.monitoring.client.Client static method), 269
- resource_record_set() (google.cloud.dns.zone.ManagedZone method), 135
- ResourceDescriptor (class in google.cloud.monitoring.resource), 273
- ResourceRecordSet (class in google.cloud.dns.resource_record_set), 135
- response (google.cloud.operation.Operation attribute), 8
- responses (google.cloud.vision_v1.types.BatchAnnotateImagesResponse attribute), 350
- result() (google.cloud.bigquery.job.CopyJob method), 25
- result() (google.cloud.bigquery.job.ExtractTableToStorageJob method), 28
- result() (google.cloud.bigquery.job.LoadTableFromStorageJob method), 33
- result() (google.cloud.bigquery.job.QueryJob method), 37
- results (google.cloud.speech_v1.types.LongRunningRecognizeResponse attribute), 253
- results (google.cloud.speech_v1.types.RecognizeResponse attribute), 254
- results (google.cloud.speech_v1.types.StreamingRecognizeResponse attribute), 257
- resume_threshold (google.cloud.pubsub_v1.types.FlowControl attribute), 189
- resume_token (google.cloud.spanner.streamed.StreamedResultSet attribute), 243
- retain_acked_messages (google.cloud.pubsub_v1.types.Subscription attribute), 194
- return_immediately (google.cloud.pubsub_v1.types.PullRequest attribute), 192
- rewrite() (google.cloud.storage.blob.Blob method), 317
- roll_angle (google.cloud.vision_v1.types.FaceAnnotation attribute), 352
- rollback() (google.cloud.datastore.Batch method), 118
- rollback() (google.cloud.datastore.batch.Batch method), 118

114

rollback() (google.cloud.datastore.Transaction method), 129

rollback() (google.cloud.datastore.transaction.Transaction method), 112

rollback() (google.cloud.spanner.transaction.Transaction method), 242

Row (class in google.cloud.bigtable.row), 83

row() (google.cloud.bigtable.table.Table method), 71

row_from_mapping() (google.cloud.bigquery.table.Table method), 46

row_key (google.cloud.bigtable.row.AppendRow attribute), 78

row_key (google.cloud.bigtable.row.ConditionalRow attribute), 80

row_key (google.cloud.bigtable.row.DirectRow attribute), 82

row_key (google.cloud.bigtable.row.Row attribute), 83

row_key (google.cloud.bigtable.row_data.PartialRowData attribute), 85

RowFilter (class in google.cloud.bigtable.row_filters), 90

RowFilterChain (class in google.cloud.bigtable.row_filters), 90

RowFilterUnion (class in google.cloud.bigtable.row_filters), 91

RowKeyRegexFilter (class in google.cloud.bigtable.row_filters), 91

rows (google.cloud.bigquery.query.QueryResults attribute), 40

rows (google.cloud.bigtable.row_data.PartialRowsData attribute), 85

rows (google.cloud.spanner.streamed.StreamedResultSet attribute), 243

RowSampleFilter (class in google.cloud.bigtable.row_filters), 91

run() (google.cloud.bigquery.query.QueryResults method), 40

run_async_query() (google.cloud.bigquery.client.Client method), 17

run_in_transaction() (google.cloud.spanner.database.Database method), 232

run_in_transaction() (google.cloud.spanner.session.Session method), 235

run_sync_query() (google.cloud.bigquery.client.Client method), 18

running() (google.cloud.bigquery.job.CopyJob method), 25

running() (google.cloud.bigquery.job.ExtractTableToStorageJob method), 29

running() (google.cloud.bigquery.job.LoadTableFromStorageJob method), 33

running() (google.cloud.bigquery.job.QueryJob method), 37

S

safe_search_annotation (google.cloud.vision_v1.types.AnnotateImageResponse attribute), 349

safe_search_detection() (google.cloud.vision_v1.ImageAnnotatorClient method), 348

SafeSearchAnnotation (class in google.cloud.vision_v1.types), 354

salience (google.cloud.language_v1.types.Entity attribute), 150

salience (google.cloud.language_v1beta2.types.Entity attribute), 160

sample_rate_hertz (google.cloud.speech_v1.types.RecognitionConfig attribute), 254

sample_row_keys() (google.cloud.bigtable.table.Table method), 71

save() (google.cloud.storage.acl.ACL method), 333

save_path (google.cloud.storage.acl.BucketACL attribute), 334

save_path (google.cloud.storage.acl.ObjectACL attribute), 335

save_predefined() (google.cloud.storage.acl.ACL method), 334

schema (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 33

schema (google.cloud.bigquery.query.QueryResults attribute), 40

schema (google.cloud.bigquery.table.Table attribute), 46

SchemaField (class in google.cloud.bigquery.schema), 41

SCOPE (google.cloud.bigquery.client.Client attribute), 15

SCOPE (google.cloud.client.Client attribute), 9

SCOPE (google.cloud.datastore.client.Client attribute), 99

SCOPE (google.cloud.dns.client.Client attribute), 131

SCOPE (google.cloud.error_reporting.client.Client attribute), 260

SCOPE (google.cloud.logging.client.Client attribute), 289

SCOPE (google.cloud.monitoring.client.Client attribute), 265

SCOPE (google.cloud.resource_manager.client.Client attribute), 197

SCOPE (google.cloud.runtimeconfig.client.Client attribute), 203

SCOPE (google.cloud.spanner.client.Client attribute), 225

SCOPE (google.cloud.storage.client.Client attribute), 336

SCOPE (google.cloud.translate_v2.client.Client attribute), 339

score (google.cloud.language_v1.types.Sentiment attribute), 152

score (google.cloud.language_v1beta2.types.Sentiment attribute), 162

score (google.cloud.vision_v1.types.ColorInfo attribute),

- 350
- score (google.cloud.vision_v1.types.EntityAnnotation attribute), 351
- score (google.cloud.vision_v1.types.WebDetection attribute), 356
- seek() (google.cloud.pubsub_v1.subscriber.client.Client method), 183
- SeekRequest (class in google.cloud.pubsub_v1.types), 193
- select_group() (google.cloud.monitoring.query.Query method), 280
- select_interval() (google.cloud.monitoring.query.Query method), 280
- select_metrics() (google.cloud.monitoring.query.Query method), 280
- select_projects() (google.cloud.monitoring.query.Query method), 281
- select_resources() (google.cloud.monitoring.query.Query method), 281
- self_link (google.cloud.bigquery.dataset.Dataset attribute), 21
- self_link (google.cloud.bigquery.job.CopyJob attribute), 25
- self_link (google.cloud.bigquery.job.ExtractTableToStorageJob attribute), 29
- self_link (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 33
- self_link (google.cloud.bigquery.job.QueryJob attribute), 37
- self_link (google.cloud.bigquery.table.Table attribute), 46
- self_link (google.cloud.storage.blob.Blob attribute), 318
- self_link (google.cloud.storage.bucket.Bucket attribute), 329
- send() (google.cloud.logging.handlers.transports.background_thread.BackgroundThreadTransport method), 304
- send() (google.cloud.logging.handlers.transports.base.Transport method), 304
- send() (google.cloud.logging.handlers.transports.sync.SyncTransport method), 37
- Sentence (class in google.cloud.language_v1.types), 152
- Sentence (class in google.cloud.language_v1beta2.types), 161
- sentences (google.cloud.language_v1.types.AnalyzeSentimentResponse attribute), 148
- sentences (google.cloud.language_v1.types.AnalyzeSyntaxResponse attribute), 149
- sentences (google.cloud.language_v1.types.AnnotateTextResponse attribute), 149
- sentences (google.cloud.language_v1beta2.types.AnalyzeSentimentResponse attribute), 158
- sentences (google.cloud.language_v1beta2.types.AnalyzeSyntaxResponse attribute), 158
- sentences (google.cloud.language_v1beta2.types.AnnotateTextResponse attribute), 159
- Sentiment (class in google.cloud.language_v1.types), 152
- Sentiment (class in google.cloud.language_v1beta2.types), 161
- sentiment (google.cloud.language_v1.types.Entity attribute), 151
- sentiment (google.cloud.language_v1.types.EntityMention attribute), 151
- sentiment (google.cloud.language_v1.types.Sentence attribute), 152
- sentiment (google.cloud.language_v1beta2.types.Entity attribute), 160
- sentiment (google.cloud.language_v1beta2.types.EntityMention attribute), 161
- sentiment (google.cloud.language_v1beta2.types.Sentence attribute), 161
- service_account() (google.cloud.iam.Policy static method), 12
- Session (class in google.cloud.spanner.session), 233
- session() (google.cloud.spanner.database.Database method), 232
- session() (google.cloud.spanner.pool.AbstractSessionPool method), 237
- session_id (google.cloud.spanner.session.Session attribute), 236
- SessionCheckout (class in google.cloud.spanner.pool), 239
- set_cell() (google.cloud.bigtable.row.ConditionalRow method), 80
- set_cell() (google.cloud.bigtable.row.DirectRow method), 82
- set_exception() (google.cloud.bigquery.job.CopyJob method), 25
- set_exception() (google.cloud.bigquery.job.ExtractTableToStorageJob method), 33
- set_exception() (google.cloud.bigquery.job.LoadTableFromStorageJob method), 33
- set_exception() (google.cloud.bigquery.job.QueryJob method), 37
- set_iam_policy() (google.cloud.pubsub_v1.publisher.client.Client method), 171
- set_iam_policy() (google.cloud.pubsub_v1.subscriber.client.Client method), 183
- set_iam_policy() (google.cloud.storage.blob.Blob method), 318
- set_iam_policy() (google.cloud.storage.bucket.Bucket method), 330
- set_properties_from_api_repr() (google.cloud.resource_manager.project.Project method), 200
- set_result() (google.cloud.bigquery.job.CopyJob method), 25
- set_result() (google.cloud.bigquery.job.ExtractTableToStorageJob method), 29
- set_result() (google.cloud.bigquery.job.LoadTableFromStorageJob method), 29

- method), 33
- set_result() (google.cloud.bigquery.job.QueryJob method), 37
- setup_logging() (google.cloud.logging.client.Client method), 291
- setup_logging() (in module google.cloud.logging.handlers.handlers), 301
- single_utterance (google.cloud.speech_v1.types.StreamingRecognitionResult attribute), 255
- Sink (class in google.cloud.logging.sink), 298
- sink() (google.cloud.logging.client.Client method), 291
- SinkFilter (class in google.cloud.bigtable.row_filters), 91
- sinks_api (google.cloud.logging.client.Client attribute), 291
- size (google.cloud.storage.blob.Blob attribute), 318
- skip_leading_rows (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 33
- Snapshot (class in google.cloud.pubsub_v1.types), 193
- Snapshot (class in google.cloud.spanner.snapshot), 241
- snapshot (google.cloud.pubsub_v1.types.DeleteSnapshotRequest attribute), 188
- snapshot (google.cloud.pubsub_v1.types.SeekRequest attribute), 193
- snapshot (google.cloud.pubsub_v1.types.UpdateSnapshotRequest attribute), 195
- snapshot() (google.cloud.spanner.database.Database method), 233
- snapshot() (google.cloud.spanner.session.Session method), 236
- snapshot_path() (google.cloud.pubsub_v1.subscriber.client.Client method), 184
- SnapshotCheckout (class in google.cloud.spanner.database), 233
- snapshots (google.cloud.pubsub_v1.types.ListSnapshotsResponse attribute), 189
- sorrow_likelihood (google.cloud.vision_v1.types.FaceAnnotation attribute), 352
- source (google.cloud.language_v1.types.Document attribute), 150
- source (google.cloud.language_v1beta2.types.Document attribute), 160
- source (google.cloud.vision_v1.types.Image attribute), 353
- source_format (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 330
- SourceFormat (class in google.cloud.bigquery.job), 38
- spanner_api (google.cloud.spanner.database.Database attribute), 233
- speech_contexts (google.cloud.speech_v1.types.RecognitionConfig attribute), 254
- speech_event_type (google.cloud.speech_v1.types.StreamingRecognitionResult attribute), 257
- SpeechClient (class in google.cloud.speech_v1), 250
- SpeechContext (class in google.cloud.speech_v1.types), 254
- SpeechRecognitionAlternative (class in google.cloud.speech_v1.types), 255
- SpeechRecognitionResult (class in google.cloud.speech_v1.types), 255
- spoof (google.cloud.vision_v1.types.SafeSearchAnnotation attribute), 355
- stability (google.cloud.speech_v1.types.StreamingRecognitionResult attribute), 255
- start_time (google.cloud.speech_v1.types.LongRunningRecognizeMetadata attribute), 253
- start_time (google.cloud.speech_v1.types.WordInfo attribute), 257
- started (google.cloud.bigquery.job.CopyJob attribute), 25
- started (google.cloud.bigquery.job.ExtractTableToStorageJob attribute), 29
- started (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 33
- started (google.cloud.bigquery.job.QueryJob attribute), 37
- started (google.cloud.dns.changes.Changes attribute), 137
- state (google.cloud.bigquery.job.CopyJob attribute), 25
- state (google.cloud.bigquery.job.ExtractTableToStorageJob attribute), 29
- state (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 33
- state (google.cloud.bigquery.job.QueryJob attribute), 37
- state (google.cloud.bigtable.row_data.PartialRowsData attribute), 86
- State (google.cloud.runtimeconfig.variable.Variable attribute), 207
- STATE_DELETED (in module google.cloud.runtimeconfig.variable), 206
- STATE_UNSPECIFIED (in module google.cloud.runtimeconfig.variable), 206
- STATE_UPDATED (in module google.cloud.runtimeconfig.variable), 206
- stats (google.cloud.spanner.streamed.StreamedResultSet attribute), 243
- status (google.cloud.dns.changes.Changes attribute), 137
- storage_class (google.cloud.storage.blob.Blob attribute), 318
- storage_class (google.cloud.storage.bucket.Bucket attribute), 330
- stream_ack_deadline_seconds (google.cloud.pubsub_v1.types.StreamingPullRequest attribute), 194
- StreamedResultSet (class in google.cloud.spanner.streamed), 242
- streaming_config (google.cloud.speech_v1.types.StreamingRecognizeRequest attribute), 256
- streaming_recognize() (google.cloud.speech_v1.SpeechClient method), 252

streaming_request (google.cloud.speech_v1.types.StreamingRecognizeRequest attribute), 256

StreamingPullRequest (class in google.cloud.pubsub_v1.types), 193

StreamingPullResponse (class in google.cloud.pubsub_v1.types), 194

StreamingRecognitionConfig (class in google.cloud.speech_v1.types), 255

StreamingRecognitionResult (class in google.cloud.speech_v1.types), 255

StreamingRecognizeRequest (class in google.cloud.speech_v1.types), 256

StreamingRecognizeResponse (class in google.cloud.speech_v1.types), 256

StripValueTransformerFilter (class in google.cloud.bigtable.row_filters), 92

StructEntry (class in google.cloud.logging.entries), 296

subscribe() (google.cloud.pubsub_v1.subscriber.client.Client method), 184

Subscription (class in google.cloud.pubsub_v1.types), 194

subscription (google.cloud.pubsub_v1.types.AcknowledgeRequest attribute), 188

subscription (google.cloud.pubsub_v1.types.CreateSnapshotRequest attribute), 188

subscription (google.cloud.pubsub_v1.types.DeleteSubscriptionRequest attribute), 189

subscription (google.cloud.pubsub_v1.types.GetSubscriptionRequest attribute), 189

subscription (google.cloud.pubsub_v1.types.ModifyAckDeadlineRequest attribute), 191

subscription (google.cloud.pubsub_v1.types.ModifyPushConfigRequest attribute), 191

subscription (google.cloud.pubsub_v1.types.PullRequest attribute), 192

subscription (google.cloud.pubsub_v1.types.SeekRequest attribute), 193

subscription (google.cloud.pubsub_v1.types.StreamingPullRequest attribute), 193

subscription (google.cloud.pubsub_v1.types.UpdateSubscriptionRequest attribute), 195

subscription_path() (google.cloud.pubsub_v1.subscriber.client.Client method), 185

subscriptions (google.cloud.pubsub_v1.types.ListSubscriptionsResponse attribute), 190

subscriptions (google.cloud.pubsub_v1.types.ListTopicSubscriptionsResponse attribute), 190

surprise_likelihood (google.cloud.vision_v1.types.FaceAnnotation attribute), 352

Symbol (class in google.cloud.vision_v1.types), 355

symbols (google.cloud.vision_v1.types.Word attribute), 357

SyncTransport (class in google.cloud.logging.handlers.transports.sync),

T

Table (class in google.cloud.bigquery.table), 42

in Table (class in google.cloud.bigtable.table), 69

table (google.cloud.bigtable.row.AppendRow attribute), 78

table (google.cloud.bigtable.row.ConditionalRow attribute), 81

table (google.cloud.bigtable.row.DirectRow attribute), 83

in table (google.cloud.bigtable.row.Row attribute), 83

table() (google.cloud.bigquery.dataset.Dataset method), 22

table() (google.cloud.bigtable.instance.Instance method), 67

TABLE_ADMIN_HOST (in module google.cloud.bigtable.client), 62

table_id (google.cloud.bigquery.table.Table attribute), 46

table_type (google.cloud.bigquery.table.Table attribute), 46

TableMismatchError, 72

table_request (google.cloud.language_v1.types.PartOfSpeech attribute), 151

table_request (google.cloud.language_v1beta2.types.PartOfSpeech attribute), 161

table_request (google.cloud.operation.Operation attribute), 8

tense (google.cloud.language_v1.types.PartOfSpeech attribute), 151

tense (google.cloud.language_v1beta2.types.PartOfSpeech attribute), 161

test_iam_permissions() (google.cloud.pubsub_v1.publisher.client.Client method), 172

test_iam_permissions() (google.cloud.pubsub_v1.subscriber.client.Client method), 185

test_iam_permissions() (google.cloud.storage.blob.Blob method), 318

test_iam_permissions() (google.cloud.storage.bucket.Bucket method), 330

text (google.cloud.language_v1.types.EntityMention attribute), 151

text (google.cloud.language_v1.types.Sentence attribute), 152

text (google.cloud.language_v1.types.Token attribute), 160

text (google.cloud.language_v1beta2.types.EntityMention attribute), 161

text (google.cloud.language_v1beta2.types.Sentence attribute), 162

text (google.cloud.language_v1beta2.types.Token attribute), 162

text (google.cloud.vision_v1.types.Symbol attribute), 355

text (google.cloud.vision_v1.types.TextAnnotation attribute), 355

text_annotations (google.cloud.vision_v1.types.AnnotateImageResponse attribute), 76
 attribute), 349
 text_detection() (google.cloud.vision_v1.ImageAnnotatorClient method), 87
 method), 348
 TextAnnotation (class in google.cloud.vision_v1.types), 355
 TextAnnotation.DetectedBreak (class in google.cloud.vision_v1.types), 355
 TextAnnotation.DetectedLanguage (class in google.cloud.vision_v1.types), 356
 TextAnnotation.TextProperty (class in google.cloud.vision_v1.types), 356
 TextEntry (class in google.cloud.logging.entries), 296
 TextSpan (class in google.cloud.language_v1.types), 152
 TextSpan (class in google.cloud.language_v1beta2.types), 162
 tilt_angle (google.cloud.vision_v1.types.FaceAnnotation attribute), 352
 time (google.cloud.pubsub_v1.types.SeekRequest attribute), 193
 time_created (google.cloud.storage.blob.Blob attribute), 319
 time_created (google.cloud.storage.bucket.Bucket attribute), 330
 time_deleted (google.cloud.storage.blob.Blob attribute), 319
 time_series() (google.cloud.monitoring.client.Client static method), 270
 timeout_ms (google.cloud.bigquery.query.QueryResults attribute), 40
 TimeSeries (class in google.cloud.monitoring.timeseries), 283
 TimestampRange (class in google.cloud.bigtable.row_filters), 92
 TimestampRangeFilter (class in google.cloud.bigtable.row_filters), 92
 to_api_repr() (google.cloud.bigquery.schema.SchemaField method), 42
 to_api_repr() (google.cloud.iam.policy.Policy method), 12
 to_dict() (google.cloud.bigtable.row_data.PartialRowData method), 85
 to_legacy_urlsafe() (google.cloud.datastore.Key method), 124
 to_legacy_urlsafe() (google.cloud.datastore.key.Key method), 106
 to_pb() (google.cloud.bigtable.column_family.ColumnFamily method), 75
 to_pb() (google.cloud.bigtable.column_family.GCRuleIntersection method), 75
 to_pb() (google.cloud.bigtable.column_family.GCRuleUnion method), 75
 to_pb() (google.cloud.bigtable.column_family.MaxAgeGCRule method), 76
 to_pb() (google.cloud.bigtable.column_family.MaxVersionsGCRule method), 76
 to_pb() (google.cloud.bigtable.row_filters.ApplyLabelFilter method), 87
 to_pb() (google.cloud.bigtable.row_filters.BlockAllFilter method), 87
 to_pb() (google.cloud.bigtable.row_filters.CellsColumnLimitFilter method), 87
 to_pb() (google.cloud.bigtable.row_filters.CellsRowLimitFilter method), 88
 to_pb() (google.cloud.bigtable.row_filters.CellsRowOffsetFilter method), 88
 to_pb() (google.cloud.bigtable.row_filters.ColumnQualifierRegexFilter method), 88
 to_pb() (google.cloud.bigtable.row_filters.ColumnRangeFilter method), 89
 to_pb() (google.cloud.bigtable.row_filters.ConditionalRowFilter method), 90
 to_pb() (google.cloud.bigtable.row_filters.FamilyNameRegexFilter method), 90
 to_pb() (google.cloud.bigtable.row_filters.PassAllFilter method), 90
 to_pb() (google.cloud.bigtable.row_filters.RowFilterChain method), 90
 to_pb() (google.cloud.bigtable.row_filters.RowFilterUnion method), 91
 to_pb() (google.cloud.bigtable.row_filters.RowKeyRegexFilter method), 91
 to_pb() (google.cloud.bigtable.row_filters.RowSampleFilter method), 91
 to_pb() (google.cloud.bigtable.row_filters.SinkFilter method), 92
 to_pb() (google.cloud.bigtable.row_filters.StripValueTransformerFilter method), 92
 to_pb() (google.cloud.bigtable.row_filters.TimestampRange method), 92
 to_pb() (google.cloud.bigtable.row_filters.TimestampRangeFilter method), 92
 to_pb() (google.cloud.bigtable.row_filters.ValueRangeFilter method), 93
 to_pb() (google.cloud.bigtable.row_filters.ValueRegexFilter method), 93
 to_pb() (google.cloud.spanner.keyset.KeyRange method), 240
 to_pb() (google.cloud.spanner.keyset.KeySet method), 240
 to_protobuf() (google.cloud.datastore.helpers.GeoPoint method), 115
 to_protobuf() (google.cloud.datastore.Key method), 125
 to_protobuf() (google.cloud.datastore.key.Key method), 106
 Token (class in google.cloud.language_v1.types), 152
 Token (class in google.cloud.language_v1beta2.types), 162
 tokenize() (google.cloud.language_v1.types.AnalyzeSyntaxResponse method), 152

- attribute), 149
 - tokens (google.cloud.language_v1.types.AnnotateTextResponse attribute), 149
 - tokens (google.cloud.language_v1beta2.types.AnalyzeSyntaxResponse attribute), 158
 - tokens (google.cloud.language_v1beta2.types.AnnotateTextResponse attribute), 159
 - TooManyMutationsError, 72
 - Topic (class in google.cloud.pubsub_v1.types), 195
 - topic (google.cloud.pubsub_v1.types.DeleteTopicRequest attribute), 189
 - topic (google.cloud.pubsub_v1.types.GetTopicRequest attribute), 189
 - topic (google.cloud.pubsub_v1.types.ListTopicSubscriptionsRequest attribute), 190
 - topic (google.cloud.pubsub_v1.types.PublishRequest attribute), 191
 - topic (google.cloud.pubsub_v1.types.Snapshot attribute), 193
 - topic (google.cloud.pubsub_v1.types.Subscription attribute), 194
 - topic (google.cloud.pubsub_v1.types.UpdateTopicRequest attribute), 195
 - topic_path() (google.cloud.pubsub_v1.publisher.client.Client method), 172
 - topic_path() (google.cloud.pubsub_v1.subscriber.client.Client method), 185
 - topicality (google.cloud.vision_v1.types.EntityAnnotation attribute), 351
 - topics (google.cloud.pubsub_v1.types.ListTopicsResponse attribute), 190
 - total_bytes_processed (google.cloud.bigquery.query.QueryResults attribute), 40
 - total_rows (google.cloud.bigquery.query.QueryResults attribute), 41
 - Transaction (class in google.cloud.datastore), 127
 - Transaction (class in google.cloud.datastore.transaction), 110
 - Transaction (class in google.cloud.spanner.transaction), 242
 - transaction() (google.cloud.datastore.Client method), 121
 - transaction() (google.cloud.datastore.client.Client method), 102
 - transaction() (google.cloud.spanner.session.Session method), 236
 - TransactionPingingPool (class in google.cloud.spanner.pool), 239
 - transcript (google.cloud.speech_v1.types.SpeechRecognitionAlternative attribute), 255
 - translate() (google.cloud.translate_v2.client.Client method), 340
 - Transport (class in google.cloud.logging.handlers.transports.base), 304
 - type (google.cloud.language_v1.types.Document attribute), 150
 - type (google.cloud.language_v1.types.Entity attribute), 150
 - type (google.cloud.language_v1.types.EntityMention attribute), 151
 - type (google.cloud.language_v1beta2.types.Document attribute), 160
 - type (google.cloud.language_v1beta2.types.Entity attribute), 160
 - type (google.cloud.language_v1beta2.types.EntityMention attribute), 161
 - type (google.cloud.vision_v1.types.FaceAnnotation attribute), 351
 - type (google.cloud.vision_v1.types.Feature attribute), 353
- ## U
- undelimited() (google.cloud.resource_manager.project.Project method), 200
 - under_exposed_likelihood (google.cloud.vision_v1.types.FaceAnnotation attribute), 352
 - Unmergeable, 243
 - update() (google.cloud.bigquery.dataset.Dataset method), 22
 - update() (google.cloud.bigquery.table.Table method), 46
 - update() (google.cloud.bigtable.cluster.Cluster method), 64
 - update() (google.cloud.bigtable.column_family.ColumnFamily method), 75
 - update() (google.cloud.bigtable.instance.Instance method), 67
 - update() (google.cloud.logging.metric.Metric method), 298
 - update() (google.cloud.logging.sink.Sink method), 299
 - update() (google.cloud.monitoring.group.Group method), 277
 - update() (google.cloud.resource_manager.project.Project method), 201
 - update() (google.cloud.spanner.instance.Instance method), 229
 - update() (google.cloud.storage.blob.Blob method), 319
 - update() (google.cloud.storage.bucket.Bucket method), 330
 - update_ddl() (google.cloud.spanner.database.Database method), 233
 - update_mask (google.cloud.pubsub_v1.types.UpdateSnapshotRequest attribute), 195
 - update_mask (google.cloud.pubsub_v1.types.UpdateSubscriptionRequest attribute), 195
 - update_mask (google.cloud.pubsub_v1.types.UpdateTopicRequest attribute), 195
 - update_storage_class() (google.cloud.storage.blob.Blob method), 319

[update_subscription\(\)](#) (google.cloud.pubsub_v1.subscriber.Credentials class method), 185
[update_time](#) (google.cloud.runtimeconfig.variable.Variable attribute), 207
[updated](#) (google.cloud.storage.blob.Blob attribute), 319
[UpdateSnapshotRequest](#) (class in google.cloud.pubsub_v1.types), 195
[UpdateSubscriptionRequest](#) (class in google.cloud.pubsub_v1.types), 195
[UpdateTopicRequest](#) (class in google.cloud.pubsub_v1.types), 195
[upload_from_file\(\)](#) (google.cloud.bigquery.table.Table method), 47
[upload_from_file\(\)](#) (google.cloud.storage.blob.Blob method), 319
[upload_from_filename\(\)](#) (google.cloud.storage.blob.Blob method), 320
[upload_from_string\(\)](#) (google.cloud.storage.blob.Blob method), 321
[uri](#) (google.cloud.speech_v1.types.RecognitionAudio attribute), 253
[url](#) (google.cloud.vision_v1.types.WebDetection attribute), 356
[use_legacy_sql](#) (google.cloud.bigquery.job.QueryJob attribute), 37
[use_legacy_sql](#) (google.cloud.bigquery.query.QueryResults attribute), 41
[use_query_cache](#) (google.cloud.bigquery.job.QueryJob attribute), 37
[use_query_cache](#) (google.cloud.bigquery.query.QueryResults attribute), 41
[user\(\)](#) (google.cloud.iam.Policy static method), 12
[user\(\)](#) (google.cloud.storage.acl.ACL method), 334
[user_email](#) (google.cloud.bigquery.job.CopyJob attribute), 25
[user_email](#) (google.cloud.bigquery.job.ExtractTableToStorageJob attribute), 29
[user_email](#) (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 33
[user_email](#) (google.cloud.bigquery.job.QueryJob attribute), 37

V

[value](#) (google.cloud.runtimeconfig.variable.Variable attribute), 207
[value](#) (google.cloud.vision_v1.types.Property attribute), 354
[VALUE_TYPE_UNSPECIFIED](#) (google.cloud.monitoring.metric.ValueType attribute), 273
[ValueRangeFilter](#) (class in google.cloud.bigtable.row_filters), 93
[ValueRegexFilter](#) (class in google.cloud.bigtable.row_filters), 93
[ValueTip](#) (class in google.cloud.monitoring.metric), 273
[Variable](#) (class in google.cloud.runtimeconfig.variable), 206
[variable\(\)](#) (google.cloud.runtimeconfig.config.Config method), 206
[versioning_enabled](#) (google.cloud.storage.bucket.Bucket attribute), 331
[Vertex](#) (class in google.cloud.vision_v1.types), 356
[vertices](#) (google.cloud.vision_v1.types.BoundingPoly attribute), 350
[view_query](#) (google.cloud.bigquery.table.Table attribute), 48
[view_use_legacy_sql](#) (google.cloud.bigquery.table.Table attribute), 48
[VIEWER_ROLE](#) (in module google.cloud.iam), 12
[viewers](#) (google.cloud.iam.Policy attribute), 12
[violence](#) (google.cloud.vision_v1.types.SafeSearchAnnotation attribute), 355
[voice](#) (google.cloud.language_v1.types.PartOfSpeech attribute), 151
[voice](#) (google.cloud.language_v1beta2.types.PartOfSpeech attribute), 161

W

[web_detection](#) (google.cloud.vision_v1.types.AnnotateImageResponse attribute), 349
[web_detection\(\)](#) (google.cloud.vision_v1.ImageAnnotatorClient method), 348
[web_entities](#) (google.cloud.vision_v1.types.WebDetection attribute), 356
[WebDetection](#) (class in google.cloud.vision_v1.types), 356
[WebDetection.WebEntity](#) (class in google.cloud.vision_v1.types), 356
[WebDetection.WebImage](#) (class in google.cloud.vision_v1.types), 356
[WebDetection.WebPage](#) (class in google.cloud.vision_v1.types), 356
[width](#) (google.cloud.vision_v1.types.Page attribute), 354
[Word](#) (class in google.cloud.vision_v1.types), 356
[word](#) (google.cloud.speech_v1.types.WordInfo attribute), 257
[WordInfo](#) (class in google.cloud.speech_v1.types), 257
[words](#) (google.cloud.speech_v1.types.SpeechRecognitionAlternative attribute), 255
[words](#) (google.cloud.vision_v1.types.Paragraph attribute), 354
[write_disposition](#) (google.cloud.bigquery.job.CopyJob attribute), 25
[write_disposition](#) (google.cloud.bigquery.job.LoadTableFromStorageJob attribute), 34
[write_disposition](#) (google.cloud.bigquery.job.QueryJob attribute), 38

`write_point()` (google.cloud.monitoring.client.Client method), [270](#)
`write_time_series()` (google.cloud.monitoring.client.Client method), [271](#)
`WriteDisposition` (class in google.cloud.bigquery.job), [38](#)

X

`x` (google.cloud.vision_v1.types.Position attribute), [354](#)

Y

`y` (google.cloud.vision_v1.types.Position attribute), [354](#)
`y` (google.cloud.vision_v1.types.Vertex attribute), [356](#)

Z

`z` (google.cloud.vision_v1.types.Position attribute), [354](#)
`zone()` (google.cloud.dns.client.Client method), [132](#)
`zone_id` (google.cloud.dns.zone.ManagedZone attribute), [135](#)