Google API eXtension Documentation

Release 0.16.0

Tim Emiola

Feb 28, 2018

Google APIs Extensions

This project has been deprecated and will no longer be developed. It has been wholly replaced by google-api-core.

CHAPTER 1

Google API Extensions for Python

This is the API documentation for Google API Extensions for Python (gax-python), a set of libraries which aids the development of APIs for clients and servers based on GRPC and Google APIs conventions.

google.gax	Google API Extensions
google.gax.api_callable	Provides function wrappers that implement page streaming and retrying.
google.gax.bundling	Provides behavior that supports request bundling.
google.gax.config	Runtime configuration shared by gax modules.
google.gax.errors	Provides GAX exceptions.
google.gax.grpc	Adapts the grpc surface.
google.gax.path_template	Implements a utility for parsing and formatting path templates.

1.1 google.gax

Google API Extensions

Classes

BackoffSettings	Parameters to the exponential backoff algorithm for retrying.
BundleDescriptor	Describes the structure of bundled call.
BundleOptions	Holds values used to configure bundling.
CallOptions([timeout, retry, page_token, ...])	Encapsulates the overridable settings for a particular API call.
PageDescriptor	Describes the structure of a page-streaming call.
PageIterator(api_call, page_descriptor, ...)	An iterator over the pages of a page streaming API call.

Continued on next page

Table 1.2 – continued from previous page

ResourceIterator(page_iterator)	An iterator over resources of the page iterator.
RetryOptions	Per-call configurable settings for retrying upon transient failure.

class BackoffSettings

Parameters to the exponential backoff algorithm for retrying.

initial_retry_delay_millis
the initial delay time, in milliseconds, between the completion of the first failed request and the initiation of the first retrying request.

retry_delay_multiplier
the multiplier by which to increase the delay time between the completion of failed requests, and the initiation of the subsequent retrying request.

max_retry_delay_millis
the maximum delay time, in milliseconds, between requests. When this value is reached, `retry_delay_multiplier` will no longer be used to increase delay time.

initial_rpc_timeout_millis
the initial timeout parameter to the request.

rpc_timeout_multiplier
the multiplier by which to increase the timeout parameter between failed requests.

max_rpc_timeout_millis
the maximum timeout parameter, in milliseconds, for a request. When this value is reached, `rpc_timeout_multiplier` will no longer be used to increase the timeout.

total_timeout_millis
the total time, in milliseconds, starting from when the initial request is sent, after which an error will be returned, regardless of the retrying attempts made meanwhile.

Create new instance of BackoffSettings(initial_retry_delay_millis, retry_delay_multiplier, max_retry_delay_millis, initial_rpc_timeout_millis, rpc_timeout_multiplier, max_rpc_timeout_millis, total_timeout_millis)

class BundleDescriptor

Describes the structure of bundled call.

request_discriminator_fields may include '.' as a separator, which is used to indicate object traversal. This allows fields in nested objects to be used to determine what requests to bundle.

bundled_field
the repeated field in the request message that will have its elements aggregated by bundling

request_discriminator_fields
a list of fields in the target request message class that are used to determine which messages should be bundled together.

subresponse_field
an optional field, when present it indicates the field in the response message that should be used to demultiplex the response into multiple response messages.

class BundleOptions

Holds values used to configure bundling.

The xxx_threshold attributes are used to configure when the bundled request should be made.

element_count_threshold
the bundled request will be sent once the count of outstanding elements in the repeated field reaches this value.

element_count_limit
represents a hard limit on the number of elements in the repeated field of the bundle; if adding a request to a bundle would exceed this value, the bundle is sent and the new request is added to a fresh bundle. It is invalid for a single request to exceed this limit.

request_byte_threshold
the bundled request will be sent once the count of bytes in the request reaches this value. Note that this value is pessimistically approximated by summing the bytesizes of the elements in the repeated field, and therefore may be an under-approximation.

request_byte_limit
represents a hard limit on the size of the bundled request; if adding a request to a bundle would exceed this value, the bundle is sent and the new request is added to a fresh bundle. It is invalid for a single request to exceed this limit. Note that this value is pessimistically approximated by summing the bytesizes of the elements in the repeated field, with a buffer applied to correspond to the resulting under-approximation.

delay_threshold
the bundled request will be sent this amount of time after the first element in the bundle was added to it.

Invokes the base constructor with default values.

The default values are zero for all attributes and it's necessary to specify at least one valid threshold value during construction.

Parameters

- **element_count_threshold** (*int*) – the bundled request will be sent once the count of outstanding elements in the repeated field reaches this value.
- **element_count_limit** (*int*) – represents a hard limit on the number of elements in the repeated field of the bundle; if adding a request to a bundle would exceed this value, the bundle is sent and the new request is added to a fresh bundle. It is invalid for a single request to exceed this limit.
- **request_byte_threshold** (*int*) – the bundled request will be sent once the count of bytes in the request reaches this value. Note that this value is pessimistically approximated by summing the bytesizes of the elements in the repeated field, with a buffer applied to compensate for the corresponding under-approximation.
- **request_byte_limit** (*int*) – represents a hard limit on the size of the bundled request; if adding a request to a bundle would exceed this value, the bundle is sent and the new request is added to a fresh bundle. It is invalid for a single request to exceed this limit. Note that this value is pessimistically approximated by summing the bytesizes of the elements in the repeated field, with a buffer applied to correspond to the resulting under-approximation.
- **delay_threshold** (*int*) – the bundled request will be sent this amount of time after the first element in the bundle was added to it.

Returns the constructed object.

Return type *BundleOptions*

static __new__(*element_count_threshold=0*, *element_count_limit=0*, *request_byte_threshold=0*, *request_byte_limit=0*, *delay_threshold=0*)
Invokes the base constructor with default values.

The default values are zero for all attributes and it's necessary to specify at least one valid threshold value during construction.

Parameters

- **element_count_threshold** (*int*) – the bundled request will be sent once the count of outstanding elements in the repeated field reaches this value.
- **element_count_limit** (*int*) – represents a hard limit on the number of elements in the repeated field of the bundle; if adding a request to a bundle would exceed this value, the bundle is sent and the new request is added to a fresh bundle. It is invalid for a single request to exceed this limit.
- **request_byte_threshold** (*int*) – the bundled request will be sent once the count of bytes in the request reaches this value. Note that this value is pessimistically approximated by summing the bytesizes of the elements in the repeated field, with a buffer applied to compensate for the corresponding under-approximation.
- **request_byte_limit** (*int*) – represents a hard limit on the size of the bundled request; if adding a request to a bundle would exceed this value, the bundle is sent and the new request is added to a fresh bundle. It is invalid for a single request to exceed this limit. Note that this value is pessimistically approximated by summing the bytesizes of the elements in the repeated field, with a buffer applied to correspond to the resulting under-approximation.
- **delay_threshold** (*int*) – the bundled request will be sent this amount of time after the first element in the bundle was added to it.

Returns the constructed object.

Return type *BundleOptions*

class CallOptions(*timeout=<object object>*, *retry=<object object>*, *page_token=<object object>*, *is_bundling=False*, ***kwargs*)

Encapsulates the overridable settings for a particular API call.

`CallOptions` is an optional arg for all GAX API calls. It is used to configure the settings of a specific API call.

When provided, its values override the GAX service defaults for that particular call.

Example

```
>>> # change an api call's timeout
>>> o1 = CallOptions(timeout=30)  # make the timeout 30 seconds
>>>
>>> # set page streaming to be per-page on a call where it is
>>> # normally per-resource
>>> o2 = CallOptions(page_token=INITIAL_PAGE)
>>>
>>> # disable retrying on an api call that normally retries
>>> o3 = CallOptions(retry=None)
>>>
>>> # enable bundling on a call that supports it
>>> o4 = CallOptions(is_bundling=True)
```

Parameters

- **timeout** (*int*) – The client-side timeout for non-retrying API calls.
- **retry** (`RetryOptions`) – determines whether and how to retry on transient errors. When set to None, the call will not retry.

- **page_token** (*str*) – If set and the call is configured for page streaming, page streaming is performed per-page, starting with this page_token. Use `INITIAL_PAGE` for the first request. If unset and the call is configured for page streaming, page streaming is performed per-resource.
- **is_bundling** (*bool*) – If set and the call is configured for bundling, bundling is performed. Bundling is always disabled by default.
- **kwargs** – Additional arguments passed through to the API call.

Raises `ValueError` – if incompatible options are specified.

INITIAL_PAGE = <object object>
A placeholder for the page token passed into an initial paginated request.

OPTION_INHERIT = <object object>
Global constant.

If a CallOptions field is set to OPTION_INHERIT, the call to which that CallOptions belongs will attempt to inherit that field from its default settings.

class PageDescriptor
Describes the structure of a page-streaming call.

Create new instance of PageDescriptor(request_page_token_field, response_page_token_field, resource_field)

class PageIterator(*api_call*, *page_descriptor*, *page_token*, *request*, ***kwargs*)
An iterator over the pages of a page streaming API call.

Provides access to the individual pages of the call, as well as the page token.

response
The full response message for the call most recently made, or None if a call has not yet been made.

page_token
The page token to be passed in the request for the next call to be made.

Parameters

- **api_call** (`Callable` [`req`, `resp`]) – an API call that is page streaming.
- **page_descriptor** (`PageDescriptor`) – indicates the structure of page streaming to be performed.
- **page_token** (*str*) – The page token to be passed to API call request. If no page token has yet been acquired, this field should be set to `INITIAL_PAGE`.
- **request** (*object*) – The request to be passed to the API call. The page token field of the request is overwritten by the `page_token` passed to the constructor, unless `page_token` is `INITIAL_PAGE`.
- **kwargs** – Arbitrary keyword arguments to be passed to the API call.

__next__()
Retrieves the next page.

next()
For Python 2.7 compatibility; see __next__.

class ResourceIterator(*page_iterator*)
An iterator over resources of the page iterator.

Constructor.

Parameters **page_iterator** (PageIterator) – the base iterator of getting pages.

__next__()
Retrieves the next resource.

next()
For Python 2.7 compatibility; see __next__.

class RetryOptions
Per-call configurable settings for retrying upon transient failure.

retry_codes
list[string] – a list of Google API canonical error codes upon which a retry should be attempted.

backoff_settings
BackoffSettings – configures the retry exponential backoff algorithm.

Create new instance of RetryOptions(retry_codes, backoff_settings)

1.2 google.gax.api_callable

Provides function wrappers that implement page streaming and retrying.

Functions

construct_settings(service_name, . . . [, . . .])	Constructs a dictionary mapping method names to _CallSettings.
create_api_call(func, settings)	Converts an rpc call into an API call governed by the settings.

construct_settings(*service_name*, *client_config*, *config_override*, *retry_names*, *bundle_descriptors=None*, *page_descriptors=None*, *metrics_headers=()*, *kwargs=None*)
Constructs a dictionary mapping method names to _CallSettings.

The `client_config` parameter is parsed from a client configuration JSON file of the form:

```
{
  "interfaces": {
    "google.fake.v1.ServiceName": {
      "retry_codes": {
        "idempotent": ["UNAVAILABLE", "DEADLINE_EXCEEDED"],
        "non_idempotent": []
      },
      "retry_params": {
        "default": {
          "initial_retry_delay_millis": 100,
          "retry_delay_multiplier": 1.2,
          "max_retry_delay_millis": 1000,
          "initial_rpc_timeout_millis": 2000,
          "rpc_timeout_multiplier": 1.5,
          "max_rpc_timeout_millis": 30000,
          "total_timeout_millis": 45000
        }
      },
      "methods": {
```

```
        "CreateFoo": {
          "retry_codes_name": "idempotent",
          "retry_params_name": "default",
          "timeout_millis": 30000
        },
        "Publish": {
          "retry_codes_name": "non_idempotent",
          "retry_params_name": "default",
          "bundling": {
            "element_count_threshold": 40,
            "element_count_limit": 200,
            "request_byte_threshold": 90000,
            "request_byte_limit": 100000,
            "delay_threshold_millis": 100
          }
        }
      }
    }
  }
}
```

Parameters

- **service_name** (*str*) – The fully-qualified name of this service, used as a key into the client config file (in the example above, this value would be `google.fake.v1.ServiceName`).
- **client_config** (*dict*) – A dictionary parsed from the standard API client config file.
- **bundle_descriptors** (`Mapping` [`str`, `BundleDescriptor`]) – A dictionary of method names to BundleDescriptor objects for methods that are bundling-enabled.
- **page_descriptors** (`Mapping` [`str`, `PageDescriptor`]) – A dictionary of method names to PageDescriptor objects for methods that are page streaming-enabled.
- **config_override** (*str*) – A dictionary in the same structure of client_config to override the settings. Usually client_config is supplied from the default config and config_override will be specified by users.
- **retry_names** (`Mapping` [`str`, `object`]) – A dictionary mapping the strings referring to response status codes to the Python objects representing those codes.
- **metrics_headers** (`Mapping` [`str`, `str`]) – Dictionary of headers to be passed for analytics. Sent as a dictionary; eventually becomes a space-separated string (e.g. 'foo/1.0.0 bar/3.14.1').
- **kwargs** (*dict*) – The keyword arguments to be passed to the API calls.

Returns A dictionary mapping method names to _CallSettings.

Return type dict

Raises `KeyError` – If the configuration for the service in question cannot be located in the provided `client_config`.

create_api_call(*func*, *settings*)

Converts an rpc call into an API call governed by the settings.

In typical usage, `func` will be a callable used to make an rpc request. This will mostly likely be a bound method from a request stub used to make an rpc call.

The result is created by applying a series of function decorators defined in this module to `func`. `settings` is used to determine which function decorators to apply.

The result is another callable which for most values of `settings` has has the same signature as the original. Only when `settings` configures bundling does the signature change.

Parameters

- **func** (`Callable [ Sequence [ object ] , object ]`) – is used to make a bare rpc call.
- **settings** (*_CallSettings*) – provides the settings for this call

Returns

a bound method on a request stub used to make an rpc call

Return type `Callable [ Sequence [ object ] , object ]`

Raises `ValueError` – if `settings` has incompatible values, e.g, if bundling and page_streaming are both configured

1.3 google.gax.bundling

Provides behavior that supports request bundling.

compute_bundle_id() is used generate ids linking API requests to the appropriate bundles.

Event is the result of scheduling a bundled api call. It is a decorated `threading.Event`; its `wait` and `is_set` methods are used wait for the bundle request to complete or determine if it has been completed respectively.

Task manages the sending of all the requests in a specific bundle.

Executor has a `schedule` method that is used add bundled api calls to a new or existing *Task*.

Functions

compute_bundle_id(obj, discriminator_fields)	Computes a bundle id from the discriminator fields of *obj*.

Classes

Event()	Wraps a threading.Event, adding, canceller and result attributes.
Executor(options)	Organizes bundling for an api service that requires it.
Task(api_call, bundle_id, bundled_field, ...)	Coordinates the execution of a single bundle.

class Event

Wraps a threading.Event, adding, canceller and result attributes.

Constructor.

cancel()

Invokes the cancellation function provided on construction.

clear()

Calls `clear` on the decorated `threading.Event`.

Also resets the result if one has been set.

is_set()
Calls is_set on the decorated threading.Event.

set()
Calls set on the decorated threading.Event.

wait(*timeout=None*)
Calls wait on the decorated threading.Event.

class Executor(*options*)
Organizes bundling for an api service that requires it.

Constructor.

Parameters **options** (gax.BundleOptions) – configures strategy this instance uses when executing bundled functions.

schedule(*api_call*, *bundle_id*, *bundle_desc*, *bundling_request*, *kwargs=None*)
Schedules bundle_desc of bundling_request as part of bundle_id.

The returned value an *Event* that

- has a result attribute that will eventually be set to the result the api call
- will be used to wait for the response
- holds the canceller function for canceling this part of the bundle

Parameters

- **api_call** (*callable[[object], object]*) – the scheduled API call.
- **bundle_id** (*str*) – identifies the bundle on which the API call should be made.
- **bundle_desc** (gax.BundleDescriptor) – describes the structure of the bundled call.
- **bundling_request** (*object*) – the request instance to use in the API call.
- **kwargs** (*dict*) – optional, the keyword arguments passed to the API call.

Returns the scheduled event.

Return type *Event*

TIMER_FACTORY
A class with an interface similar to threading.Timer.

Defaults to threading.Timer. This makes it easy to plug-in alternate timer implementations.

alias of Timer

class Task(*api_call*, *bundle_id*, *bundled_field*, *bundling_request*, *kwargs*, *subresponse_field=None*)
Coordinates the execution of a single bundle.

Parameters

- **api_call** (Callable [Sequence [object] , object]) – the func that is this tasks's API call.
- **bundle_id** (Tuple [str]) – the id of this bundle.
- **bundled_field** (*str*) – the field used to create the bundled request.
- **bundling_request** (*object*) – the request to pass as the arg to api_call.
- **kwargs** (*dict*) – keyword arguments passed to api_call.

- **subresponse_field** (*str*) – optional field used to demultiplex responses.

element_count
The number of bundled elements in the repeated field.

extend(*elts*)
Adds elts to the tasks.

Parameters **elts** (*Sequence*) – a iterable of elements that can be appended to the task's bundle_field.

Returns an event that can be used to wait on the response.

Return type *Event*

request_bytesize
The size of in bytes of the bundled field elements.

run()
Call the task's func.

The task's func will be called with the bundling requests func

compute_bundle_id(*obj*, *discriminator_fields*)
Computes a bundle id from the discriminator fields of *obj*.

discriminator_fields may include '.' as a separator, which is used to indicate object traversal. This is meant to allow fields in the computed bundle_id.

the id is a tuple computed by going through the discriminator fields in order and obtaining the str(value) object field (or nested object field)

if any discriminator field cannot be found, ValueError is raised.

Parameters

- **obj** (*object*) – an object.
- **discriminator_fields** (Sequence [str]) – a list of discriminator fields in the order to be to be used in the id.

Returns computed as described above.

Return type Tuple [str]

Raises AttributeError – if any discriminator fields attribute does not exist.

1.4 google.gax.config

Runtime configuration shared by gax modules.

API_ERRORS = (<class 'grpc.RpcError'>,)
Errors that indicate that an RPC was aborted.

NAME_STATUS_CODES = {<StatusCode.FAILED_PRECONDITION: (9, 'failed precondition')>: 'FAILE
Inverse map for STATUS_CODE_NAMES

STATUS_CODE_NAMES = {'FAILED_PRECONDITION': <StatusCode.FAILED_PRECONDITION: (9, 'failed p
Maps strings used in client config to the status codes they represent.

This is necessary for google.gax.api_callable.construct_settings to translate the client constants configuration for retrying into the correct gRPC objects.

create_stub(*generated_create_stub*, *channel=None*, *service_path=None*, *service_port=None*, *credentials=None*, *scopes=None*, *ssl_credentials=None*)
The function to use to create stubs.

exc_to_code(*exc*)
A function that takes an exception and returns a status code.

May return None if the exception is not associated with a status code.

1.5 google.gax.errors

Provides GAX exceptions.

Functions

create_error(msg[, cause])	Creates a `GaxError` or subclass.

Exceptions

GaxError(msg[, cause])	Common base class for exceptions raised by GAX.
InvalidArgumentError(msg[, cause])	GAX exception class for `INVALID_ARGUMENT` errors.
RetryError(msg[, cause])	Indicates an error during automatic GAX retrying.

exception GaxError(*msg*, *cause=None*)
Common base class for exceptions raised by GAX.

msg
string – describes the error that occurred.

cause
Exception, optional – the exception raised by a lower layer of the RPC stack (for example, gRPC) that caused this exception, or None if this exception originated in GAX.

exception InvalidArgumentError(*msg*, *cause=None*)
GAX exception class for `INVALID_ARGUMENT` errors.

msg
string – describes the error that occurred.

cause
Exception, optional – the exception raised by a lower layer of the RPC stack (for example, gRPC) that caused this exception, or None if this exception originated in GAX.

exception RetryError(*msg*, *cause=None*)
Indicates an error during automatic GAX retrying.

create_error(*msg*, *cause=None*)
Creates a `GaxError` or subclass.

msg
string – describes the error that occurred.

cause
Exception, optional – the exception raised by a lower layer of the RPC stack (for example, gRPC) that

caused this exception, or None if this exception originated in GAX.

Returns The exception that wraps `cause`.

Return type *GaxError*

1.6 google.gax.grpc

Adapts the grpc surface.

Functions

create_stub(generated_create_stub[, ...])	Creates a gRPC client stub.
exc_to_code(exc)	Retrieves the status code from an exception

API_ERRORS = (<class 'grpc.RpcError'>,)
gRPC exceptions that indicate that an RPC was aborted.

NAME_STATUS_CODES = {<StatusCode.FAILED_PRECONDITION: (9, 'failed precondition')>: 'FAILE
Inverse map for STATUS_CODE_NAMES

STATUS_CODE_NAMES = {'FAILED_PRECONDITION': <StatusCode.FAILED_PRECONDITION: (9, 'failed p
Maps strings used in client config to gRPC status codes.

create_stub(*generated_create_stub*, *channel=None*, *service_path=None*, *service_port=None*, *credentials=None*, *scopes=None*, *ssl_credentials=None*)
Creates a gRPC client stub.

Parameters

- **generated_create_stub** (*Callable*) – The generated gRPC method to create a stub.
- **channel** (*grpc.Channel*) – A Channel object through which to make calls. If None, a secure channel is constructed. If specified, all remaining arguments are ignored.
- **service_path** (*str*) – The domain name of the API remote host.
- **service_port** (*int*) – The port on which to connect to the remote host.
- **credentials** (*google.auth.credentials.Credentials*) – The authorization credentials to attach to requests. These credentials identify your application to the service.
- **scopes** (Sequence [str]) – The OAuth scopes for this service. This parameter is ignored if a credentials is specified.
- **ssl_credentials** (*grpc.ChannelCredentials*) – gRPC channel credentials used to create a secure gRPC channel. If not specified, SSL credentials will be created using default certificates.

Returns A gRPC client stub.

Return type grpc.Client

exc_to_code(*exc*)
Retrieves the status code from an exception

1.7 google.gax.path_template

Implements a utility for parsing and formatting path templates.

Classes

PathTemplate(data)	Represents a path template.

Exceptions

ValidationException	Represents a path template validation error.

class PathTemplate(*data*)
Represents a path template.

match(*path*)
Matches a fully qualified path template string.

Parameters **path** (*str*) – A fully qualified path template string.

Returns Var names to matched binding values.

Return type dict

Raises *ValidationException* – If path can't be matched to the template.

render(*bindings*)
Renders a string from a path template using the provided bindings.

Parameters **bindings** (*dict*) – A dictionary of var names to binding strings.

Returns The rendered instantiation of this path template.

Return type str

Raises *ValidationException* – If a key isn't provided or if a sub-template can't be parsed.

exception ValidationException
Represents a path template validation error.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

g

Index

Symbols

A

B

C

D

E

G

I

M

N

O

P

R

S

T

V

W