Package ‘googlePubsubR’

March 3, 2023

Title R Interface for Google 'Cloud Pub/Sub' REST API
Version 0.0.4
Description Provides an easy to use interface to the 'Google Pub/Sub' REST API <https://cloud.google.com/pubsub/docs/reference/rest>.
URL https://github.com/andodet/googlePubsubR
BugReports https://github.com/andodet/googlePubsubR/issues
License MIT + file LICENSE
Imports googleAuthR (>= 0.3), cli, magrittr
Suggests testthat (>= 3.0.0), jsonlite, base64enc, knitr, rmarkdown, shiny, future, promises
Encoding UTF-8
RoxygenNote 7.2.3
VignetteBuilder knitr
Config/testthat/edition 3
Config/testthat/parallel false
NeedsCompilation no
Author Andrea Dodet [aut, cre]
Maintainer Andrea Dodet <andodet@gmail.com>
Repository CRAN
Date/Publication 2023-03-03 01:00:02 UTC

R topics documented:

DeadLetterPolicy ............................................................ 3
DlqPolicy ................................................................. 3
ExpirationPolicy ........................................................... 4
MessageStoragePolicy ..................................................... 5
msg_decode ................................................................. 5
msg_encode ............................................................... 6
ps_project_get ........................................... 7
ps_project_set ......................................... 7
PubsubMessage .......................................... 8
pubsub_auth ............................................ 9
PushConfig ............................................. 9
RetryPolicy ............................................... 10
Schema ................................................... 11
SchemaSettings ......................................... 11
schemas_create ......................................... 12
schemas_delete ......................................... 13
schemas_exists ......................................... 13
schemas_get ............................................ 14
schemas_list ........................................... 14
schemas_validate ....................................... 15
schemas_validate_message .............................. 16
Snapshot .................................................. 16
snapshots_create ........................................ 17
snapshots_delete ....................................... 18
snapshots_exists ....................................... 18
snapshots_get .......................................... 19
snapshots_list ......................................... 19
snapshots_patch ....................................... 20
Subscription ............................................. 21
subscriptions_ack ...................................... 22
subscriptions_create ................................... 23
subscriptions_delete ................................... 24
subscriptions_detach ................................... 25
subscriptions_exists ................................... 25
subscriptions_get ...................................... 26
subscriptions_list ..................................... 27
subscriptions_modify_ack_deadline .................... 27
subscriptions_modify_pushconf ....................... 28
subscriptions_patch ................................... 29
subscriptions_pull ..................................... 30
subscriptions_seek ..................................... 31
Topic ..................................................... 31
topics_create .......................................... 32
topics_delete .......................................... 33
topics_exists .......................................... 34
topics_get ............................................. 34
topics_list ............................................ 35
topics_list_subscriptions ............................. 35
topics_patch .......................................... 36
topics_publish ........................................ 37

Index 38
**DeadLetterPolicy**

*Builds a DeadLetterPolicy Object*

**Description**

Builds a DeadLetterPolicy Object

**Usage**

`DeadLetterPolicy(max_delivery_attempts = NULL, dead_letter_topic = NULL)`

**Arguments**

- `max_delivery_attempts` numeric The maximum number of delivery attempts for any message
- `dead_letter_topic` character, Topic The name of the topic to which dead letter messages should be published

**Value**

DeadLetterPolicy object

**See Also**

Other Object functions: `DlqPolicy()`, `ExpirationPolicy()`, `MessageStoragePolicy()`, `PubsubMessage()`, `PushConfig()`, `RetryPolicy()`, `SchemaSettings()`, `Schema()`, `Snapshot()`, `Subscription()`, `Topic()`

---

**DlqPolicy**

*Builds a DlqPolicy object*

**Description**

Builds a DlqPolicy object

**Usage**

`DlqPolicy(dlq_topic, max_delivery_attempts)`

**Arguments**

- `dlq_topic` character, Topic Required, topic name or instance of a topic object
- `max_delivery_attempts` numeric Number of delivery attempts for any message. The value must be between 5 and 100.
ExpirationPolicy

Value

DlqPolicy object

See Also

Other Object functions: DeadLetterPolicy(), ExpirationPolicy(), MessageStoragePolicy(), PubsubMessage(), PushConfig(), RetryPolicy(), SchemaSettings(), Schema(), Snapshot(), Subscription(), Topic()

ExpirationPolicy  Builds a ExpirationPolicy Object

Description

Builds a ExpirationPolicy Object

Usage

ExpirationPolicy(ttl = NULL)

Arguments

ttl numeric Specifies the 'time-to-live' duration (in seconds, can be float) for an associated resource

Details

Autogenerated via gar_create_api_objects A policy that specifies the conditions for resource expiration (i.e., automatic resource deletion).

Value

ExpirationPolicy object

See Also

Other Object functions: DeadLetterPolicy(), DlqPolicy(), MessageStoragePolicy(), PubsubMessage(), PushConfig(), RetryPolicy(), SchemaSettings(), Schema(), Snapshot(), Subscription(), Topic()
**MessageStoragePolicy**

*Builds a MessageStoragePolicy object*

**Description**

Builds a MessageStoragePolicy object

**Usage**

`MessageStoragePolicy(regions)`

**Arguments**

regions character A list of IDs of GCP regions

**Value**

MessageStoragePolicy object

**See Also**

Other Object functions: `DeadLetterPolicy()`, `DlqPolicy()`, `ExpirationPolicy()`, `PubsubMessage()`, `PushConfig()`, `RetryPolicy()`, `SchemaSettings()`, `Schema()`, `Snapshot()`, `Subscription()`, `Topic()`

---

**msg_decode**

*Decode Pub/Sub message*

**Description**

Converts a Pub/Sub message into an object

**Usage**

`msg_decode(x)`

**Arguments**

x A base64 encoded string

**Value**

A deserialized object
Examples

```r
## Not run:
library(jsonlite)

pulled_msgs$receivedMessages$messages$data %>%
  msg_decode() %>%
  fromJSON()

## End(Not run)
```

---

### msg_encode

**Encode Pub/Sub message**

**Description**

Converts an object into a base64 string

**Usage**

```r
msg_encode(x)
```

**Arguments**

- `x` A serializeable object

**Value**

character a base64 encoded string

**Examples**

```r
## Not run:
library(jsonlite)

mtcars %>%
  toJSON(auto_unbox = TRUE) %>%
  msg_encode() %>%
  PubsubMessage()

## End(Not run)
```
ps_project_get  

Get GCP projectId

Description
Get GCP projectId

Usage
ps_project_get()

Value
character  A valid GCP projectId, defaults to GCP_PROJECT env var

See Also
Other Auth functions: ps_project_set(), pubsub_auth()

ps_project_set  

Set GCP projectId

Description
Set GCP projectId

Usage
ps_project_set(project_id)

Arguments
project_id  character  A valid GCP projectId

Value
character  ProjectId string

See Also
Other Auth functions: ps_project_get(), pubsub_auth()
Examples

```r
## Not run:
ps_project_set("my-new-project")
# Do whatever...
# Jump back on the default project
ps_project_set(Sys.getenv("GCP_PROJECT"))

## End(Not run)
```

---

**PubsubMessage**  
*Builds a PubsubMessage Object*

## Description

Builds a PubsubMessage Object

## Usage

```r
PubsubMessage(
  data = NULL,
  message_id = NULL,
  ordering_key = NULL,
  attributes = NULL,
  publish_time = NULL
)
```

## Arguments

- **data** character The message data field as a base64 encoded string
- **message_id** character ID of this message, assigned by the server when the message is published
- **ordering_key** character If non-empty, identifies related messages for which publish order should be respected
- **attributes** list Key-value pairs attributes for this message
- **publish_time** character The time at which the message was published, populated by the server when it receives the Publish call

## Value

PubsubMessage object

## See Also

Other Object functions: `DeadLetterPolicy()`, `DlqPolicy()`, `ExpirationPolicy()`, `MessageStoragePolicy()`, `PushConfig()`, `RetryPolicy()`, `SchemaSettings()`, `Schema()`, `Snapshot()`, `Subscription()`, `Topic()`
pubsub_auth

Authenticate a Pub/Sub client

Description

Authenticate a Pub/Sub client

Usage

```r
pubsub_auth(
    json_file = Sys.getenv("GCP_AUTH_FILE"),
    token = NULL,
    email = NULL
)
```

Arguments

- `json_file` character: Path of the JSON file containing credentials for a GCP service account
- `token` character: An existing authentication token
- `email` character: The email to default authentication to

Value

None, called for side effects

See Also

Other Auth functions: `ps_project_get()`, `ps_project_set()`

PushConfig

Builds a PushConfig Object

Description

Builds a PushConfig Object

Usage

```r
PushConfig(attributes = NULL, push_endpoint = NULL, oidcToken = NULL)
```

Arguments

- `attributes` list: The attributes object or list of objects
- `push_endpoint` character: A URL locating the endpoint to which messages should be pushed
- `oidcToken` character: If specified, Pub/Sub will generate and attach an OIDC JWT token as an Authorization header in the HTTP request for every pushed message
RetryPolicy

Value

PushConfig object

See Also

Other Object functions: DeadLetterPolicy(), DlqPolicy(), ExpirationPolicy(), MessageStoragePolicy(), PubsubMessage(), RetryPolicy(), SchemaSettings(), Schema(), Snapshot(), Subscription(), Topic()

---

| RetryPolicy | Builds a retry policy object |

Description

More on this here

Usage

RetryPolicy(min_backoff = 600, max_backoff = 600)

Arguments

- **min_backoff** numeric The minimum delay between consecutive deliveries of a given message
- **max_backoff** numeric The maximum delay between consecutive deliveries of a given message

Value

RetryPolicy object

See Also

Other Object functions: DeadLetterPolicy(), DlqPolicy(), ExpirationPolicy(), MessageStoragePolicy(), PubsubMessage(), PushConfig(), SchemaSettings(), Schema(), Snapshot(), Subscription(), Topic()
**Schema**

**Builds a Schema Object**

**Description**

Builds a Schema Object

**Usage**

```r
Schema(type = NULL, definition = NULL, name = NULL)
```

**Arguments**

- **type** character  The type of the schema definition
- **definition** character  The definition of the schema
- **name** character  The schema name

**Value**

Schema object

**See Also**

Other Object functions: DeadLetterPolicy(), DlqPolicy(), ExpirationPolicy(), MessageStoragePolicy(), PubsubMessage(), PushConfig(), RetryPolicy(), SchemaSettings(), Snapshot(), Subscription(), Topic()

---

**SchemaSettings**

**SchemaSettings Object**

**Description**

SchemaSettings Object

**Usage**

```r
SchemaSettings(encoding = NULL, schema = NULL)
```

**Arguments**

- **encoding** character  The encoding of messages validated against schema
- **schema** Schema, character  Required, schema object or schema name

**Value**

SchemaSettings object
schemas_create

See Also


| schemas_create | Creates a schema |

Description

Creates a schema

Usage

schemas_create(
  name,
  type = c("AVRO", "PROTOCOL_BUFFER", "TYPE_UNSPECIFIED"),
  definition,
  project = ps_project_get()
)

Arguments

name character, Schema Required, schema name or instance of a schema object

| type character Type of the schema definition |

| definition character Required, the definition of the schema |

| project character GCP project id |

Value

| a Schema object |

See Also

Other Schema functions: schemas_delete(), schemas_exists(), schemas_get(), schemas_list(), schemas_validate_message(), schemas_validate()
schemas_delete

Deletes a schema

Description

Deletes a schema

Usage

schemas_delete(name)

Arguments

name character, Schema Schema name or instance of a schema object

Value

None, called for side effects

See Also

Other Schema functions: schemas_create(), schemas_exists(), schemas_get(), schemas_list(), schemas_validate_message(), schemas_validate()

schemas_exists

Check if a schema exists

Description

Check if a schema exists

Usage

schemas_exists(schema)

Arguments

schema character, Schema Required, schema name or an instance of a Schema object

Value

logical TRUE if the schema exists

See Also

Other Schema functions: schemas_create(), schemas_delete(), schemas_get(), schemas_list(), schemas_validate_message(), schemas_validate()
### schemas_get

**Description**

Gets a schema

**Usage**

```r
schemas_get(schema, view = c("SCHEMA_VIEW_UNSPECIFIED", "BASIC", "FULL"))
```

**Arguments**

- `schema` character, Schema Required, schema name or an instance of a Schema object
- `view` character The set of fields to return in the response

**Value**

A Schema object

**See Also**

Other Schema functions: `schemas_create()`, `schemas_delete()`, `schemas_exists()`, `schemas_list()`, `schemas_validate_message()`, `schemas_validate()`

### schemas_list

**Description**

Lists all schemas in a project

**Usage**

```r
schemas_list(
  project = ps_project_get(),
  pageSize = NULL,
  view = c("SCHEMA_VIEW_UNSPECIFIED", "BASIC", "FULL"),
  pageToken = NULL
)
```
schemas_validate

Arguments

- **project**: character GCP project id
- **pageSize**: numeric Maximum number of schemas to return
- **view**: list The set of Schema fields to return in the response
- **pageToken**: character The value returned by the last ListSchemasResponse; indicates that this is a continuation of a prior ListSchemas call, and that the system should return the next page of data

Value

A data.frame containing all schema objects and properties

See Also

Other Schema functions: `schemas_create()`, `schemas_delete()`, `schemas_exists()`, `schemas_get()`, `schemas_validate_message()`, `schemas_validate()`

 schemas_validate  Validates a schema

Description

Validates a schema

Usage

```r
schemas_validate(schema, project = ps_project_get())
```

Arguments

- **schema**: Schema Required, an instance of a Schema object
- **project**: character GCP project id

Value

logical TRUE if successfully validated

See Also

Other Schema functions: `schemas_create()`, `schemas_delete()`, `schemas_exists()`, `schemas_get()`, `schemas_list()`, `schemas_validate_message()`
schemas_validate_message

Validates a message against a schema

Description

Validates a message against a schema

Usage

schemas_validate_message(
  schema,
  message,
  encoding = c("ENCODING_UNSPECIFIED", "JSON", "BINARY"),
  project = ps_project_get()
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>schema</td>
<td>character, Schema Required, schema name or instance of a Schema object</td>
</tr>
<tr>
<td>message</td>
<td>PubsubMessage Required, an instance of a PubsubMessage, can be created using PubsubMessage</td>
</tr>
<tr>
<td>encoding</td>
<td>character The encoding of the message</td>
</tr>
<tr>
<td>project</td>
<td>character A GCP project id</td>
</tr>
</tbody>
</table>

Value

logical TRUE if successfully validated

See Also

Other Schema functions: schemas_create(), schemas_delete(), schemas_exists(), schemas_get(), schemas_list(), schemas_validate()

Snapshot

Builds a Snapshot Object

Description

Builds a Snapshot Object

Usage

Snapshot(topic = NULL, expire_time = NULL, name = NULL, labels = NULL)
Snapshots are used in Seek operations, which allow you to manage message acknowledgments in bulk. That is, you can set the acknowledgment state of messages in an existing subscription to the state captured by a snapshot. If the snapshot already exists, returns ALREADY_EXISTS. If the requested subscription doesn’t exist, returns NOT_FOUND. If the backlog in the subscription is too old – and the resulting snapshot would expire in less than 1 hour – then FAILED_PRECONDITION is returned. See also the Snapshot.expire_time field. 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

**Usage**

```
snapshots_create(name, subscription, labels = NULL)
```

**Arguments**

- `name`: Snapshot, character
  - Required, an instance of a Snapshot object or a snapshot name
- `subscription`: Subscription, character
  - Required, an instance of a Subscription object or a subscription name
- `labels`: list
  - Key-value pairs for snapshot labels
snapshots_exists

Value

An instance of a Snapshot object

See Also

Other Snapshot functions: snapshots_delete(), snapshots_exists(), snapshots_list(), snapshots_patch()
snapshots_get

Value

logical TRUE if snapshot exists

See Also

Other Snapshot functions: snapshots_create(), snapshots_delete(), snapshots_list(), snapshots_patch()

snapshots_get Gets the configuration details of a snapshot

Description

Gets the configuration details of a snapshot

Usage

snapshots_get(snapshot)

Arguments

snapshot Snapshot, character Required, an instance of a Snapshot object or a snapshot name

Value

An instance of a Snapshot object

snapshots_list Lists the existing snapshots

Description

Lists the existing snapshots

Usage

snapshots_list(project = ps_project_get(), pageSize = NULL, pageToken = NULL)

Arguments

project character a GCP project ID
pageSize numeric Maximum number of snapshots to return
pageToken character 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
Value

A `data.frame` containing all snapshots

See Also

Other Snapshot functions: `snapshots_create()`, `snapshots_delete()`, `snapshots_exists()`, `snapshots_patch()`

---

`snapshots_patch` 

*Updates an existing snapshot*

Description

Updates an existing snapshot

Usage

`snapshots_patch(snapshot, topic = NULL, expire_time = NULL, labels = NULL)`

Arguments

- `snapshot` : `Snapshot`, `character`
  Required, an instance of a Snapshot object or a snapshot name
- `topic` : `character`, `Topic`
  Topic name or instance of a topic object
- `expire_time` : `string`
  The snapshot is guaranteed to exist up until this time. Must be formatted in RFC3339 UTC "Zulu" format
- `labels` : `list`
  Key-value pairs for topic labels

Value

An instance the patched Snapshot object

See Also

Other Snapshot functions: `snapshots_create()`, `snapshots_delete()`, `snapshots_exists()`, `snapshots_list()`
Description

Builds a Subscription Object

Usage

Subscription(
    dead_letter_policy = NULL,
    msg_retention_duration = NULL,
    labels = NULL,
    retry_policy = NULL,
    push_config = NULL,
    ack_deadline = NULL,
    expiration_policy = NULL,
    filter = NULL,
    detached = NULL,
    retain_acked_msgs = NULL,
    topic = NULL,
    name = NULL,
    enable_msg_ordering = NULL,
    topic_msg_retention = NULL
)

Arguments

dead_letter_policy
    DeadLetterPolicy A policy that specifies the conditions for dead lettering messages in this subscription

msg_retention_duration
    character How long to retain unacknowledged messages in the subscription’s backlog, from the moment a message is published

labels
    list See Creating and managing labels

retry_policy
    RetryPolicy A policy that specifies how Pub/Sub retries message delivery for this subscription

push_config
    PushConfig If push delivery is used with this subscription, this field is used to configure it

ack_deadline
    character The approximate amount of time (on a best-effort basis) Pub/Sub waits for the subscriber to acknowledge receipt before resending the message

expiration_policy
    ExpirationPolicy A policy that specifies the conditions for this subscription’s expiration

filter
    character An expression written in the Pub/Sub filter language
detached logical Indicates whether the subscription is detached from its topic
retain_acked_msgs logical Indicates whether to retain acknowledged messages
topic character, Topic A Topic object or topic name
name character A name for the subscription
enable_msg_ordering logical If true, messages published with the same ordering_key in PubsubMessage will be delivered to the subscribers in the order in which they are received by the Pub/Sub system
topic_msg_retention character minimum duration for which a message is retained after it is published to the subscription’s topic

Value
Subscription object

See Also

subscriptions_ack Acknowledges the messages

Description
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.

Usage
subscriptions_ack(ack_ids, subscription)

Arguments
ack_ids character A vector containing one or more message ackIDs
subscription character, Subscription Required, the subscription whose messages are being acknowledged

Value
logical TRUE if message(s) was successfully acknowledged
subscriptions_create

See Also

Other Subscription functions: subscriptions_create(), subscriptions_delete(), subscriptions_detach(), subscriptions_exists(), subscriptions_get(), subscriptions_list(), subscriptions_modify_ack_deadline(), subscriptions_modify_pushconf(), subscriptions_patch(), subscriptions_pull(), subscriptions_seek()

subscriptions_create  Creates a subscription to a given topic

Description

Creates a subscription to a given topic

Usage

subscriptions_create(
    name,
    topic,
    dead_letter_policy = NULL,
    msg_retention_duration = NULL,
    labels = NULL,
    retry_policy = NULL,
    push_config = NULL,
    ack_deadline = NULL,
    expiration_policy = NULL,
    filter = NULL,
    detached = NULL,
    retain_acked_messages = NULL,
    enable_msg_ordering = NULL
)

Arguments

    name  character  Required, name of the subscription to be created
    topic  Topic, character  Required, an instance of a Topic object or a topic name
    dead_letter_policy  DeadLetterPolicy  A policy object that specifies the conditions for dead lettering messages in this subscription
    msg_retention_duration  string  How long to retain unacknowledged messages in the subscription’s backlog in seconds
    labels  list  Key-value pairs for snapshot labels
    retry_policy  RetryPolicy  A RetryPolicy object that specifies how Pub/Sub retries message delivery for this subscription
    push_config  PushConfig  A PushConfig object
subscriptions_delete

**Description**

All messages retained in the subscription will be 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.

**Usage**

```
subscriptions_delete(subscription)
```

**Arguments**

- `subscription` character, Subscription Required, subscription name or instance of a Subscription object

**Value**

None, called for side effects

---

**Value**

A Subscription object

**See Also**

Other Subscription functions: `subscriptions_ack()`, `subscriptions_delete()`, `subscriptions_detach()`, `subscriptions_exists()`, `subscriptions_get()`, `subscriptions_list()`, `subscriptions_modify_ack_deadline()`, `subscriptions_modify_pushconf()`, `subscriptions_patch()`, `subscriptions_pull()`, `subscriptions_seek()`

---

**Description**

`subscriptions_delete` Deletes an existing subscription.

**Usage**

```
subscriptions_delete(subscription)
```

**Arguments**

- `subscription` character, Subscription Required, subscription name or instance of a Subscription object

**Value**

None, called for side effects
See Also

Other Subscription functions: subscriptions_ack(), subscriptions_create(), subscriptions_detach(), subscriptions_exists(), subscriptions_get(), subscriptions_list(), subscriptions_modify_ack_deadline(), subscriptions_modify_pushconf(), subscriptions_patch(), subscriptions_pull(), subscriptions_seek()

subscriptions_detach

Detaches a subscription from a topic.

Description

Detaches a subscription from a topic.

Usage

subscriptions_detach(subscription)

Arguments

subscription character, Subscription Required, subscription name or instance of a Subscription object

Value

logical, TRUE if successfully detached

See Also

Other Subscription functions: subscriptions_ack(), subscriptions_create(), subscriptions_delete(), subscriptions_exists(), subscriptions_get(), subscriptions_list(), subscriptions_modify_ack_deadline(), subscriptions_modify_pushconf(), subscriptions_patch(), subscriptions_pull(), subscriptions_seek()

subscriptions_exists

Check if a subscription exists

Description

Check if a subscription exists

Usage

subscriptions_exists(subscription)

Arguments

subscription character, Subscription Required, subscription name or instance of a Subscription object
submissions_get

Value

logical TRUE if the subscription exist

See Also

Other Subscription functions: submissions_ack(), submissions_create(), submissions_delete(), submissions_detach(), submissions_get(), submissions_list(), submissions_modify_ack_deadline(), submissions_modify_pushconf(), submissions_patch(), submissions_pull(), submissions_seek()

submissions_get

Gets the configuration details of a subscription.

Description

Gets the configuration details of a subscription.

Usage

submissions_get(subscription)

Arguments

subscription character, Subscription Required, subscription name or instance of a Subscription object

Value

A Subscription object

See Also

Other Subscription functions: submissions_ack(), submissions_create(), submissions_delete(), submissions_detach(), submissions_exists(), submissions_list(), submissions_modify_ack_deadline(), submissions_modify_pushconf(), submissions_patch(), submissions_pull(), submissions_seek()
subscriptions_list  List subscriptions

Description

List subscriptions

Usage

submissions_list(
    project = Sys.getenv("GCP_PROJECT"),
    pageSize = NULL,
    pageToken = NULL
)

Arguments

project character Required, GCP project id
pageSize numeric Maximum number of subscriptions to return
pageToken character The value returned by the last submissions_list; indicates that this is a continuation of a prior submissions_list call

Value

list A list containing all subscriptions

See Also

Other Subscription functions: subscriptions_ack(), subscriptions_create(), subscriptions_delete(), submissions_detach(), subscriptions_exists(), subscriptions_get(), subscriptions_modify_ack_deadline(), subscriptions_modify_pushconf(), subscriptions_patch(), subscriptions_pull(), subscriptions_seek()

submissions_modify_ack_deadline

Modify the ack deadline for a subscription

Description

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.

Usage

submissions_modify_ack_deadline(subscription, ack_ids, ack_deadline)
**subscriptions_modify_pushconf**

**Modify PushConfig for a subscription**

### Description

Modify PushConfig for a subscription

### Usage

```r
subscriptions_modify_pushconf(subscription, push_config)
```

### Arguments

- **subscription** character, Subscription Required, a subscription name or a Subscription object
- **push_config** PushConfig New PushConfig object, can be built using `PushConfig`

### Value

**logical**, TRUE if successfully modified

### See Also

Other Subscription functions: `subscriptions_ack()`, `subscriptions_create()`, `subscriptions_delete()`, `subscriptions_detach()`, `subscriptions_exists()`, `subscriptions_get()`, `subscriptions_list()`, `subscriptions_modify_ack_deadline()`, `subscriptions_patch()`, `subscriptions_pull()`, `subscriptions_seek()`
**subscriptions_patch**

Updates an existing subscription.

**Description**

Certain properties of a subscription, such as its topic, are not modifiable.

**Usage**

```r
subscriptions_patch(
    subscription,
    topic,
    labels = NULL,
    dead_letter_policy = NULL,
    msg_retention_duration = NULL,
    retry_policy = NULL,
    push_config = NULL,
    ack_deadline = NULL,
    expiration_policy = NULL,
    filter = NULL,
    detached = NULL,
    retain_acked_msgs = NULL,
    enable_ordering = NULL
)
```

**Arguments**

- **subscription** character, Subscription Required, a subscription name or a Subscription object
- **topic** character, Topic Required, a topic name or a Topic object
- **labels** labels Key value pairs
- **dead_letter_policy** DeadLetterPolicy A DeadLetterPolicy object
- **msg_retention_duration** numeric How long to retain unacknowledged messages (in seconds)
- **retry_policy** RetryPolicy policy that specifies how Pub/Sub retries message delivery for this subscription, can be built with RetryPolicy
- **push_config** PushConfig Can be built with PushConfig
- **ack_deadline** numeric amount of time (in seconds) Pub/Sub waits for the subscriber to acknowledge receipt before resending the message
- **expiration_policy** ExpirationPolicy specifies the conditions for this subscription’s expiration. Can be built with ExpirationPolicy
- **filter** character An expression written in the Pub/Sub filter language
detach logical Indicates whether the subscription is detached from its topic
retain_acked_msgs logical Indicates whether to retain acknowledged messages
enable Ordering logical messages published with the same orderingKey in PubsubMessage will be delivered to the subscribers in the order in which they are received by the Pub/Sub system

Value

An updated Subscription object

See Also

Other Subscription functions: `subscriptions_ack()`, `subscriptions_create()`, `subscriptions_delete()`, `subscriptions_detach()`, `subscriptions_exists()`, `subscriptions_get()`, `subscriptions_list()`, `subscriptions_modify_ack_deadline()`, `subscriptions_modify_pushconf()`, `subscriptions_pull()`, `subscriptions_seek()`

---

**subscriptions_pull**

Pulls messages from the server.

**Description**

Pulls messages from the server.

**Usage**

```
subscriptions_pull(subscription, max_messages = 100)
```

**Arguments**

- `subscription` character, Subscription Required, subscription where to pull messages from
- `max_messages` numeric Maximum number of messages to return

**Value**

A named list with pulled messages

**See Also**

Other Subscription functions: `subscriptions_ack()`, `subscriptions_create()`, `subscriptions_delete()`, `subscriptions_detach()`, `subscriptions_exists()`, `subscriptions_get()`, `subscriptions_list()`, `subscriptions_modify_ack_deadline()`, `subscriptions_modify_pushconf()`, `subscriptions_patch()`, `subscriptions_seek()`
**subscriptions_seek**

*Seek a subscription to a point in time*

**Description**

A subscription can be sought to a point in time or to a given snapshot.

**Usage**

```
subscriptions_seek(subscription, time = NULL, snapshot = NULL)
```

**Arguments**

- `subscription` character, Subscription Required, a snapshot name or a Snapshot object
- `time` character A timestamp in RFC3339 UTC "Zulu" format
- `snapshot` character, Snapshot A Snapshot name or a Snapshot object

**Value**

logical TRUE when successful sought

**See Also**

Other Subscription functions: `subscriptions_ack()`, `subscriptions_create()`, `subscriptions_delete()`, `subscriptions_detach()`, `subscriptions_exists()`, `subscriptions_get()`, `subscriptions_list()`, `subscriptions_modify_ack_deadline()`, `subscriptions_modify_pushconf()`, `subscriptions_patch()`, `subscriptions_pull()`

---

**Topic**

*Builds a Topic Object*

**Description**

Builds a Topic Object

**Usage**

```
Topic(
  labels = NULL,
  name = NULL,
  kms_key_name = NULL,
  satisfies_pzs = NULL,
  message_storage_policy = NULL,
  schema_settings = NULL,
  message_retention_duration = NULL
)
```
topics_create

Arguments

- **labels**: list Key-value pairs for topic labels
- **name**: character Name of the topic
- **kms_key_name**: character The resource name of the Cloud KMS CryptoKey to be used to protect access to messages published on this topic
- **satisfies_pzs**: logical Reserved for future use
- **message_storage_policy**: MessageStoragePolicy Policy constraining the set of Google Cloud Platform regions where messages published to the topic may be stored
- **schema_settings**: SchemaSettings Settings for validating messages published against a schema
- **message_retention_duration**: character Indicates the minimum duration to retain a message after it is published to the topic

Value

Topic object

See Also

Other Object functions: DeadLetterPolicy(), DlqPolicy(), ExpirationPolicy(), MessageStoragePolicy(), PubsubMessage(), PushConfig(), RetryPolicy(), SchemaSettings(), Schema(), Snapshot(), Subscription()

topics_create  Creates a pub/sub topic

Description

Creates a pub/sub topic

Usage

```r
topics_create(
  name,
  labels = NULL,
  kms_key_name = NULL,
  satisfies_pzs = NULL,
  message_storage_policy = NULL,
  schema_settings = NULL,
  message_retention_duration = NULL
)
```
**topics_delete**

**Description**

Deletes a pub/sub topic

**Usage**

`topics_delete(topic)`

**Arguments**

name character, Topic Required, topic name or instance of a topic object

labels list Key-value pairs for topic labels

kms_key_name character The resource name of the Cloud KMS CryptoKey to be used to protect access to messages published on this topic.

satisfies_pzs logical Reserved for future use.

message_storage_policy MessageStorePolicy An instance of a MessageStorePolicy object Policy constraining the set of Google Cloud Platform regions where messages published to the topic may be stored

schema_settings SchemaSettings An instance of a SchemaSettings object

message_retention_duration numeric Indicates the minimum duration (in seconds) to retain a message after it is published to the topic

**Value**

A Topic object representing the freshly created topic

**See Also**

Other Topic functions: `topics_create()`, `topics_exists()`, `topics_get()`, `topics_list_subscriptions()`, `topics_list()`, `topics_patch()`, `topics_publish()`
topics_exists  
*Check if a topic exists*

**Description**
Check if a topic exists

**Usage**
```r
topics_exists(topic, project = ps_project_get())
```

**Arguments**
- **topic** character, Topic Required, topic name or instance of a topic object
- **project** character GCP project id

**Value**
logical, TRUE if topic exists, FALSE otherwise

**See Also**
Other Topic functions: `topics_create()`, `topics_delete()`, `topics_get()`, `topics_list_subscriptions()`, `topics_list()`, `topics_patch()`, `topics_publish()`

---

topics_get  
*Gets a topic configuration*

**Description**
Gets a topic configuration

**Usage**
```r
topics_get(topic)
```

**Arguments**
- **topic** character, Topic Required, topic name or instance of a Topic

**Value**
Topic, A Topic object

**See Also**
Other Topic functions: `topics_create()`, `topics_delete()`, `topics_exists()`, `topics_list_subscriptions()`, `topics_list()`, `topics_patch()`, `topics_publish()`
**topics_list**

*Lists topics from project*

**Description**

Lists topics from project

**Usage**

```
topics_list(project = ps_project_get(), pageSize = NULL, pageToken = NULL)
```

**Arguments**

- **project** character GCP project id
- **pageSize** numeric Maximum number of topics to return
- **pageToken** character 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.

**Value**

A list of topics

**See Also**

Other Topic functions: `topics_create()`, `topics_delete()`, `topics_exists()`, `topics_get()`, `topics_list_subscriptions()`, `topics_patch()`, `topics_publish()`

**topics_list_subscriptions**

*List attached subscriptions to a topic.*

**Description**

List attached subscriptions to a topic.

**Usage**

```
topics_list_subscriptions(topic, pageToken = NULL, pageSize = NULL)
```

**Arguments**

- **topic** Topic, character Required, an instance of a Topic object or a topic name
- **pageToken** character The value returned by the last response; indicates that this is a continuation of a prior `topics_list_subscriptions()` paged call, and that the system should return the next page of data
- **pageSize** numeric Maximum number of subscription names to return
topics_patch

Value

A character vector

See Also

Other Topic functions: topics_create(), topics_delete(), topics_exists(), topics_get(), topics_list(), topics_patch(), topics_publish()

topics_patch

Updates an existing topic

Description

Updates an existing topic

Usage

```r
topics_patch(
  topic,
  labels = NULL,
  message_storage_policy = NULL,
  kms_key_name = NULL,
  schema_settings = NULL,
  satisfies_pzs = NULL,
  message_retention_duration = NULL
)
```

Arguments

- **topic** character, Topic Required, topic name or instance of a Topic object
- **labels** list Key-value pairs for topic labels
- **message_storage_policy** MessageStoragePolicy Policy constraining the set of Google Cloud Platform regions where messages published to the topic may be stored.
- **kms_key_name** character The resource name of the Cloud KMS CryptoKey to be used to protect access to messages published on this topic.
- **schema_settings** SchemaSettings An instance of a SchemaSettings object
- **satisfies_pzs** logical Reserved for future use.
- **message_retention_duration** character Indicates the minimum duration to retain a message after it is published to the topic.

Value

An instance of the patched Topic
topics_publish

See Also
Other Topic functions: topics_create(), topics_delete(), topics_exists(), topics_get(), topics_list_subscriptions(), topics_list(), topics_publish()

---

topics_publish        Adds one or more messages to the topic

Description
Adds one or more messages to the topic

Usage

topics_publish(messages, topic)

Arguments

messages list Required, a list containing the messages to be published

topic Topic, character Required, an instance of a Topic object or a topic name

Value
A character vector containing message IDs

See Also
Other Topic functions: topics_create(), topics_delete(), topics_exists(), topics_get(), topics_list_subscriptions(), topics_list(), topics_patch()
Index

* Auth functions
  ps_project_get, 7
  ps_project_set, 7
  pubsub_auth, 9

* Object functions
  DeadLetterPolicy, 3
  DlqPolicy, 3
  ExpirationPolicy, 4
  MessageStoragePolicy, 5
  PubsubMessage, 8
  PushConfig, 9
  RetryPolicy, 10
  Schema, 11
  SchemaSettings, 11
  Snapshot, 16
  Subscription, 21
  Topic, 31

* Schema functions
  schemas_create, 12
  schemas_delete, 13
  schemas_exists, 13
  schemas_get, 14
  schemas_list, 14
  schemas_validate, 15
  schemas_validate_message, 16

* Snapshot functions
  snapshots_create, 17
  snapshots_delete, 18
  snapshots_exists, 18
  snapshots_list, 19
  snapshots_patch, 20

* Subscription functions
  subscriptions_ack, 22
  subscriptions_create, 23
  subscriptions_delete, 24
  subscriptions_detach, 25
  subscriptions_exists, 25
  subscriptions_get, 26
  subscriptions_list, 27
  subscriptions_modify_ack_deadline, 27
  subscriptions_modify_pushconf, 28
  subscriptions_patch, 29
  subscriptions_pull, 30
  subscriptions_seek, 31

* Topic functions
  topics_create, 32
  topics_delete, 33
  topics_exists, 34
  topics_get, 34
  topics_list, 35
  topics_list_subscriptions, 35
  topics_patch, 36
  topics_publish, 37

  DeadLetterPolicy, 3, 4, 5, 8, 10–12, 17, 22, 32
  DlqPolicy, 3, 3, 4, 5, 8, 10–12, 17, 22, 32
  ExpirationPolicy, 3, 4, 4, 5, 8, 10–12, 17, 22, 29, 32
  gar_create_api_objects, 4
  MessageStoragePolicy, 3, 4, 5, 8, 10–12, 17, 22, 32
  msg_decode, 5
  msg_encode, 6
  ps_project_get, 7, 7, 9
  ps_project_set, 7, 9
  pubsub_auth, 7, 9
  PubsubMessage, 3–5, 8, 10–12, 16, 17, 22, 32
  PushConfig, 3–5, 8, 9, 10–12, 17, 22, 28, 29, 32
  RetryPolicy, 3–5, 8, 10, 10, 11, 12, 17, 22, 29, 32
  Schema, 3–5, 8, 10, 11, 12, 17, 22, 32
INDEX

schemas_create, 12, 13–16
schemas_delete, 12, 13, 14–16
schemas_exists, 12, 13, 14–16
schemas_get, 12, 13, 14, 15, 16
schemas_list, 12–14, 14, 15, 16
schemas_validate, 12–15, 15, 16
schemas_validate_message, 12–15, 16
SchemaSettings, 3–5, 8, 10, 11, 11, 17, 22, 32
Snapshots, 3–5, 8, 10–12, 16, 22, 32
snapshots_create, 17, 18–20
snapshots_delete, 18, 18, 19, 20
snapshots_exists, 18, 18, 20
snapshots_get, 19
snapshots_list, 18, 19, 19, 20
snapshots_patch, 18–20, 20
Subscription, 3–5, 8, 10–12, 17, 21, 32
subscriptions_ack, 22, 24–28, 30, 31
subscriptions_create, 23, 23, 25–28, 30, 31
subscriptions_delete, 23, 24, 24, 25–28, 30, 31
subscriptions_detach, 23–25, 25, 26–28, 30, 31
subscriptions_exists, 23–25, 25, 26–28, 30, 31
subscriptions_get, 23–26, 26, 27, 28, 30, 31
subscriptions_list, 23–26, 27, 28, 30, 31
subscriptions_modify_ack_deadline, 23–27, 27, 28, 30, 31
subscriptions_modify_pushconf, 23–28, 28, 30, 31
subscriptions_patch, 23–28, 29, 30, 31
subscriptions_pull, 23–28, 30, 30, 31
subscriptions_seek, 23–28, 30, 31
Topic, 3–5, 8, 10–12, 17, 22, 31
topics_create, 32, 33–37
topics_delete, 33, 33, 34–37
topics_exists, 33, 34, 34, 35–37
topics_get, 33, 34, 34, 35–37
topics_list, 33, 34, 35, 36, 37
topics_list_subscriptions, 33–35, 35, 37
topics_patch, 33–36, 36, 37
topics_publish, 33–37, 37