

Package ‘rgoogleads’

July 23, 2025

Title Loading Data from 'Google Ads API'

Version 0.12.0

Description Interface for loading data from 'Google Ads API',
see <<https://developers.google.com/google-ads/api/docs/start>>.
Package provide function for authorization and loading reports.

License MIT + file LICENSE

BugReports <https://github.com/selesnow/rgoogleads/issues>

URL <https://selesnow.github.io/rgoogleads/>,
<https://selesnow.github.io/rgoogleads/docs/>,
<https://github.com/selesnow/rgoogleads>

Encoding UTF-8

Imports gargle (>= 1.2.0), httr, stringr, rlang, dplyr (>= 1.0.0),
tidyr (>= 1.0.0), jsonlite, snakecase, cli (>= 3.0.0), pbapply,
purrr, withr, rlist, rvest (>= 1.0.0), memoise, cachem,
rappdirs, utils, lifecycle

RoxygenNote 7.2.3

Suggests rmarkdown, knitr, DT

VignetteBuilder knitr

NeedsCompilation no

Author Alexey Seleznev [aut, cre] (ORCID:
<<https://orcid.org/0000-0003-0410-7385>>),
Netpeak [cph]

Maintainer Alexey Seleznev <selesnow@gmail.com>

Repository CRAN

Date/Publication 2025-01-10 16:30:02 UTC

Contents

rgoogleads-package	2
gads_auth	4
gads_auth_configure	6
gads_check_errors	8
gads_customer	8
gads_customer_id_from_env	9
gads_customer_id_to_env	9
gads_deauth	10
gads_fix_names	10
gads_get_accessible_customers	11
gads_get_account_hierarchy	11
gads_get_ads	12
gads_get_ad_groups	15
gads_get_ad_group_criteria	17
gads_get_campaigns	20
gads_get_fields	22
gads_get_geo_targets	22
gads_get_keywords	23
gads_get_metadata	24
gads_get_report	25
gads_has_token	28
gads_keyword_plan_forecast_metrics	29
gads_keyword_plan_forecast_timeseries	30
gads_keyword_plan_historical_metrics	31
gads_last_request_ids	32
gads_set_customer_id	33
gads_set_login_customer_id	33
gads_token	34
gads_user	34
Index	35

rgoogleads-package	<i>Loading Data From 'Google Ads API'</i>
--------------------	---

Description

Interface for loading data from 'Google Ads API', see <https://developers.google.com/google-ads/api/docs/start>. Package provide function for authorization and loading reports.

Capabilities of rgoogleads:

- Authorization in the Google Ads API
- Loading a list of top-level accounts
- Loading the entire hierarchy of accounts from manager accounts
- Loading list of Google Ads client account objects: campaigns, ad groups, ads, etc.

- Loading statistics from Google Ads client account
- Loading resource metadata, resource fields, segments and metrics
- Loading forecast and historical metrics from Keyword Planning.

Author(s)

Alexey Seleznev

See Also

- [YouTube lessons playlist](#)
- [Official Google Ads API documantation](#)
- [Google Ads Query Builder](#)
- [rgoogleads home page](#)

Examples

```
## Not run:
library(rgoogleads)

# set own oauth app
gads_auth_configure(path = 'C:/auth/app.json')
# set your developer token if needed, or use default developer token
gads_auth(email = 'me@gmail.com', developer_token = "own developer token")

# get list of accessible accounts
my_accounts <- gads_get_accessible_customers()

# set manager account id
gads_set_login_customer_id('xxx-xxx-xxxx')

# set client account id
gads_set_customer_id('xxx-xxx-xxxx')

# load report data
ad_group_report <- gads_get_report(
  resource = "ad_group",
  fields = c("ad_group.campaign",
             "ad_group.id",
             "ad_group.name",
             "ad_group.status",
             "metrics.clicks",
             "metrics.cost_micros"),
  date_from = "2021-06-10",
  date_to = "2021-06-17",
  where = "ad_group.status = 'ENABLED'",
  order_by = c("metrics.clicks DESC", "metrics.cost_micros")
)

## End(Not run)
```

gads_auth

*Authorization in Google Ads API***Description**

Authorize rgoogleads to view and manage your Google Ads Account. This function is a wrapper around `gargle::token_fetch()`.

By default, you are directed to a web browser, asked to sign in to your Google account, and to grant rgoogleads permission to operate on your behalf with Google Ads. By default, with your permission, these user credentials are cached in a folder below your home directory, from where they can be automatically refreshed, as necessary. Storage at the user level means the same token can be used across multiple projects and tokens are less likely to be synced to the cloud by accident.

Usage

```
gads_auth(
  email = gargle::gargle_oauth_email(),
  path = NULL,
  cache = gargle::gargle_oauth_cache(),
  use_oob = gargle::gargle_oob_default(),
  developer_token = getOption("gads.developer.token"),
  token = NULL
)
```

Arguments

email	Optional. Allows user to target a specific Google identity.
path	Path to JSON file with identifying the service account
cache	Specifies the OAuth token cache.
use_oob	Whether to prefer "out of band" authentication.
developer_token	Your Google Ads Developer Token.
token	A token with class <code>Token2.0</code> or an object of

Details

Most users, most of the time, do not need to call `gads_auth()` explicitly – it is triggered by the first action that requires authorization. Even when called, the default arguments often suffice.

However, when necessary, `gads_auth()` allows the user to explicitly:

- Declare which Google identity to use, via an email specification.
- Use a service account token or workload identity federation via path.
- Bring your own token.
- Customize scopes.

- Use a non-default cache folder or turn caching off.
- Explicitly request out-of-band (OOB) auth via `use_oob`.

If you are interacting with R within a browser (applies to RStudio Server, Posit Workbench, Posit Cloud, and Google Colaboratory), you need OOB auth or the pseudo-OOB variant. If this does not happen automatically, you can request it explicitly with `use_oob = TRUE` or, more persistently, by setting an option via `options(gargle_oob_default = TRUE)`.

The choice between conventional OOB or pseudo-OOB auth is determined by the type of OAuth client. If the client is of the "installed" type, `use_oob = TRUE` results in conventional OOB auth. If the client is of the "web" type, `use_oob = TRUE` results in pseudo-OOB auth. Packages that provide a built-in OAuth client can usually detect which type of client to use. But if you need to set this explicitly, use the "gargle_oauth_client_type" option:

```
options(gargle_oauth_client_type = "web")      # pseudo-OOB
# or, alternatively
options(gargle_oauth_client_type = "installed") # conventional OOB
```

For details on the many ways to find a token, see [gargle::token_fetch\(\)](#). For deeper control over auth, use [gads_auth_configure\(\)](#) to bring your own OAuth client or API key. To learn more about gargle options, see [gargle::gargle_options](#).

Value

[Token2.0](#)

See Also

Other auth functions: [gads_auth_configure\(\)](#), [gads_deauth\(\)](#)

Examples

```
## Not run:
## load/refresh existing credentials, if available
## otherwise, go to browser for authentication and authorization
gads_auth()

## force use of a token associated with a specific email
gads_auth(email = "yourname@example.com")

## force a menu where you can choose from existing tokens or
## choose to get a new one
gads_auth(email = NA)

## -----
## use own developer token
gads_auth(
  email = "yourname@example.com",
  developer_token = "your developer token"
)
```

```
## -----  
## use own OAuth client app  
gads_auth_configure(  
  path = "path/to/your/oauth_client.json"  
)  
  
gads_auth(email = "yourname@example.com")  
  
## End(Not run)
```

gads_auth_configure *Edit and view auth configuration*

Description

These functions give more control over and visibility into the auth configuration than [gads_auth\(\)](#) does. `gads_auth_configure()` lets the user specify their own:

- OAuth client, which is used when obtaining a user token.
- API key. If `rgoogleads` is de-authorized via [gads_deauth\(\)](#), all requests are sent with an API key in lieu of a token.

See the vignette("get-api-credentials", package = "gargle") for more. If the user does not configure these settings, internal defaults are used.

`gads_oauth_client()` and `gads_api_key()` retrieve the currently configured OAuth client and API key, respectively.

Usage

```
gads_auth_configure(  
  client,  
  path,  
  api_key,  
  developer_token,  
  app = lifecycle::deprecated()  
)  
  
gads_auth_cache_path()  
  
gads_open_auth_cache_folder()  
  
gads_api_key()  
  
gads_developer_token()  
  
gads_oauth_app()
```

Arguments

client	A Google OAuth client, presumably constructed via <code>gargle::gargle_oauth_client_from_json()</code> . Note, however, that it is preferred to specify the client with JSON, using the path argument.
path	JSON downloaded from Google Cloud Console , containing a client id and secret, in one of the forms supported for the txt argument of <code>jsonlite::fromJSON()</code> (typically, a file path or JSON string).
api_key	API key.
developer_token	Your Google Ads Developer Token.
app	[Deprecated] Replaced by the client argument.

Value

- `gads_auth_configure()`: An object of R6 class `gargle::AuthState`, invisibly.
- `gads_oauth_client()`: the current user-configured OAuth client.
- `gads_api_key()`: the current user-configured API key.

See Also

Other auth functions: `gads_auth()`, `gads_deauth()`

Examples

```
## Not run:
# see and store the current user-configured OAuth app (probably `NULL`)
(original_app <- gads_oauth_app())

# see and store the current user-configured API key (probably `NULL`)
(original_api_key <- gads_api_key())

if (require(httr)) {
  # bring your own app via client id (aka key) and secret
  google_app <- httr::oauth_app(
    "my-awesome-google-api-wrapping-package",
    key = "YOUR_CLIENT_ID_GOES_HERE",
    secret = "YOUR_SECRET_GOES_HERE"
  )
  google_key <- "YOUR_API_KEY"
  gads_auth_configure(app = google_app, api_key = google_key)

  # confirm the changes
  gads_oauth_app()
  gads_api_key()

  # bring your own app via JSON downloaded from Google Developers Console
  # this file has the same structure as the JSON from Google
  gads_auth_configure(path = app_path)
```

```

    # confirm the changes
    gads_oauth_app()

    # use own developer token
    gads_auth_configure(developer_token = 'Your developer token')

}

# restore original auth config
gs4_auth_configure(app = original_app, api_key = original_api_key)

## End(Not run)

```

gads_check_errors	<i>Helper function for check api answer on error</i>
-------------------	--

Description

Helper function for check api answer on error

Usage

```
gads_check_errors(out, client_id = NULL, verbose = FALSE, request_id)
```

Arguments

out	API answer
client_id	Google Ads Customer id
verbose	Console output
request_id	Api request id

Value

stop the function when api request faild

gads_customer	<i>Get all information about Google Ads Customer</i>
---------------	--

Description

Get all information about Google Ads Customer

Usage

```
gads_customer(customer_id = getOption("gads.customer.id"), verbose = TRUE)
```

Arguments

<code>customer_id</code>	Google Ads customer id
<code>verbose</code>	Processing log output into console

Value

Google Ads customer data

See Also

[Method: SearchStream documentation](#)

`gads_customer_id_from_env`

Get customer id for error message

Description

Get customer id for error message

Usage

`gads_customer_id_from_env()`

Value

only set customer id into env

`gads_customer_id_to_env`

Write customer id for error message

Description

Write customer id for error message

Usage

`gads_customer_id_to_env(customer_id)`

Arguments

<code>customer_id</code>	Your client customer id
--------------------------	-------------------------

Value

only set customer id into env

<code>gads_deauth</code>	<i>Suspend authorization</i>
--------------------------	------------------------------

Description

Put rgoogleads into a de-authorized state. Instead of sending a token, rgoogleads will send an API key. This can be used to access public resources for which no Google sign-in is required. This is handy for using rgoogleads in a non-interactive setting to make requests that do not require a token. It will prevent the attempt to obtain a token interactively in the browser. The user can configure their own API key via `gads_auth_configure()` and retrieve that key via `gads_api_key()`. In the absence of a user-configured key, a built-in default key is used.

Usage

`gads_deauth()`

Value

only suspend authorization

See Also

Other auth functions: `gads_auth_configure()`, `gads_auth()`

<code>gads_fix_names</code>	<i>function for fix names in get_report</i>
-----------------------------	---

Description

function for fix names in `get_report`

Usage

`gads_fix_names(x)`

Arguments

`x` character, column names

Value

new columns names

`gads_get_accessible_customers`

Get all data of customers directly accessible by the user authenticating the call.

Description

Get all data of customers directly accessible by the user authenticating the call.

Usage

```
gads_get_accessible_customers()
```

Value

List of your accessible accounts from top level

See Also

[Method: customers.listAccessibleCustomers documentation](#)

Examples

```
## Not run:
accounts <- gads_get_accessible_customers()

## End(Not run)
```

`gads_get_account_hierarchy`

Get Google Ads Manager Account Hierarchy

Description

Get Google Ads Manager Account Hierarchy

Usage

```
gads_get_account_hierarchy(
  manager_customer_id = getOption("gads.login.customer.id"),
  include_drafts = FALSE,
  login_customer_id = getOption("gads.login.customer.id")
)
```

Arguments

manager_customer_id
ID of the manager account whose hierarchy you want to get.

include_drafts logical, Including drafts child account.

login_customer_id
Ypor top-level manager account id.

Value

tibble with data of all the child accounts

See Also

[Get Account Hierarchy API documentation](#)

Examples

```
## Not run:
acc_hier <- gads_get_account_hierarchy(
  manager_customer_id = '111-111-1111',
  login_customer_id   = '000-000-0000')

## End(Not run)
```

gads_get_ads

Get Ads Dictionary From Google Ads Client Account

Description

Get Ads Dictionary From Google Ads Client Account

Usage

```
gads_get_ads(
  fields = c("ad_group_ad.ad.id", "ad_group_ad.ad.name",
    "ad_group_ad.ad.added_by_google_ads", "ad_group_ad.ad.app_ad.descriptions",
    "ad_group_ad.ad.app_ad.headlines", "ad_group_ad.ad.app_ad.html5_media_bundles",
    "ad_group_ad.ad.app_ad.images", "ad_group_ad.ad.app_ad.mandatory_ad_text",
    "ad_group_ad.ad.call_ad.business_name", "ad_group_ad.ad.call_ad.call_tracked",
    "ad_group_ad.ad.call_ad.conversion_action",
    "ad_group_ad.ad.app_engagement_ad.videos",
    "ad_group_ad.ad.call_ad.conversion_reporting_state",
    "ad_group_ad.ad.call_ad.country_code",
    "ad_group_ad.ad.call_ad.description1",
    "ad_group_ad.ad.call_ad.description2",
    "ad_group_ad.ad.call_ad.disable_call_conversion", "ad_group_ad.ad.call_ad.headline1",
    "ad_group_ad.ad.call_ad.headline2", "ad_group_ad.ad.call_ad.path1",
```

```
"ad_group_ad.ad.call_ad.path2", "ad_group_ad.ad.call_ad.phone_number",
"ad_group_ad.ad.call_ad.phone_number_verification_url",
"ad_group_ad.ad.device_preference",
"ad_group_ad.ad.display_upload_ad.display_upload_product_type",
"ad_group_ad.ad.display_upload_ad.media_bundle",
"ad_group_ad.ad.display_url",
"ad_group_ad.ad.expanded_dynamic_search_ad.description",
"ad_group_ad.ad.expanded_dynamic_search_ad.description2",
"ad_group_ad.ad.expanded_text_ad.description",
"ad_group_ad.ad.expanded_text_ad.description2",
"ad_group_ad.ad.expanded_text_ad.headline_part1",
"ad_group_ad.ad.expanded_text_ad.headline_part2",
"ad_group_ad.ad.expanded_text_ad.headline_part3",
"ad_group_ad.ad.expanded_text_ad.path1", "ad_group_ad.ad.expanded_text_ad.path2",
"ad_group_ad.ad.final_url_suffix",
"ad_group_ad.ad.final_urls",
"ad_group_ad.ad.final_mobile_urls", "ad_group_ad.ad.hotel_ad",
"ad_group_ad.ad.image_ad.image_url", "ad_group_ad.ad.image_ad.mime_type",
"ad_group_ad.ad.image_ad.name", "ad_group_ad.ad.image_ad.pixel_height",
"ad_group_ad.ad.image_ad.pixel_width", "ad_group_ad.ad.image_ad.preview_image_url",
"ad_group_ad.ad.image_ad.preview_pixel_height",
"ad_group_ad.ad.image_ad.preview_pixel_width",
"ad_group_ad.ad.legacy_app_install_ad",
"ad_group_ad.ad.legacy_responsive_display_ad.accent_color",

"ad_group_ad.ad.legacy_responsive_display_ad.allow_flexible_color",
"ad_group_ad.ad.legacy_responsive_display_ad.business_name",
"ad_group_ad.ad.legacy_responsive_display_ad.description",
"ad_group_ad.ad.legacy_responsive_display_ad.call_to_action_text",
"ad_group_ad.ad.legacy_responsive_display_ad.format_setting",
"ad_group_ad.ad.legacy_responsive_display_ad.logo_image",
"ad_group_ad.ad.legacy_responsive_display_ad.long_headline",
"ad_group_ad.ad.legacy_responsive_display_ad.main_color",
"ad_group_ad.ad.legacy_responsive_display_ad.marketing_image",

"ad_group_ad.ad.legacy_responsive_display_ad.price_prefix",
"ad_group_ad.ad.legacy_responsive_display_ad.promo_text",
"ad_group_ad.ad.legacy_responsive_display_ad.short_headline",
"ad_group_ad.ad.legacy_responsive_display_ad.square_logo_image",
"ad_group_ad.ad.legacy_responsive_display_ad.square_marketing_image",
"ad_group_ad.ad.local_ad.call_to_actions", "ad_group_ad.ad.local_ad.descriptions",
"ad_group_ad.ad.local_ad.headlines", "ad_group_ad.ad.local_ad.logo_images",
"ad_group_ad.ad.local_ad.marketing_images",
"ad_group_ad.ad.local_ad.path1",
"ad_group_ad.ad.local_ad.path2", "ad_group_ad.ad.resource_name",
"ad_group_ad.ad.responsive_display_ad.accent_color",
"ad_group_ad.ad.responsive_display_ad.allow_flexible_color",
"ad_group_ad.ad.responsive_display_ad.business_name",
```

```

    "ad_group_ad.ad.responsive_display_ad.call_to_action_text",
    "ad_group_ad.ad.responsive_display_ad.control_spec.enable_asset_enhancements",
    "ad_group_ad.ad.responsive_display_ad.control_spec.enable_autogen_video",
    "ad_group_ad.ad.responsive_display_ad.format_setting",

    "ad_group_ad.ad.responsive_display_ad.headlines",
    "ad_group_ad.ad.responsive_display_ad.long_headline",
    "ad_group_ad.ad.responsive_display_ad.main_color",
    "ad_group_ad.ad.responsive_display_ad.price_prefix",
    "ad_group_ad.ad.responsive_display_ad.promo_text",
    "ad_group_ad.ad.responsive_display_ad.square_marketing_images",
    "customer.descriptive_name", "customer.id"),
  where = NULL,
  order_by = NULL,
  limit = NULL,
  parameters = NULL,
  customer_id = getOption("gads.customer.id"),
  login_customer_id = getOption("gads.login.customer.id"),
  include_resource_name = FALSE,
  cl = NULL,
  verbose = TRUE
)

```

Arguments

fields	character vector, list pf report fields, all report has own fields list, for example see field list of ads report .
where	Filter, for example you can filter campaigns by status where = "campaign.status = 'ENABLED' ".
order_by	Sorting, character vectors of fields and sorting directions, for example order_by = c("campaign.name DESC", "metrics.clicks").
limit	Maximun rows in report
parameters	Query parameters, for example parameters = "include_drafts=true".
customer_id	Google Ads client customer id, supports a single account id: "xxx-xxx-xxxx" or a vector of ids from the same Google Ads MCC: c("xxx-xxx-xxxx", "xxx-xxx-xxxx")
login_customer_id	Google Ads manager customer id
include_resource_name	Get resource names fields in report
cl	A cluster object created by makeCluster , or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).
verbose	Console log output

Value

tibble with ads dicrionary

See Also

[Google Ads Query Builder](#)

Examples

```
## Not run:
# set client customer id
gads_set_login_customer_id('xxx-xxx-xxxx')

# set manager id if you work under MCC
gads_set_customer_id('xxx-xxx-xxxx')

# load ads list
myads <- gads_get_ads(
  fields = c("ad_group_ad.ad.id",
             "customer.descriptive_name",
             "ad_group_ad.ad.call_ad.description1",
             "ad_group_ad.ad.call_ad.description2"),
  where = 'ad_group_ad.status = "ENABLED"'
)

## End(Not run)
```

gads_get_ad_groups	<i>Get Ad Groups Dictionary From Google Ads Client Account</i>
--------------------	--

Description

Get Ad Groups Dictionary From Google Ads Client Account

Usage

```
gads_get_ad_groups(
  customer_id = getOption("gads.customer.id"),
  fields = c("ad_group.id", "ad_group.name", "ad_group.status",
             "ad_group.ad_rotation_mode", "ad_group.base_ad_group", "ad_group.campaign",
             "campaign.id", "ad_group.display_custom_bid_dimension",
             "ad_group.effective_target_cpa_source", "ad_group.effective_target_roas",
             "ad_group.effective_target_roas_source", "ad_group.final_url_suffix",
             "ad_group.target_roas", "ad_group.type", "ad_group.url_custom_parameters",
             "ad_group.tracking_url_template", "customer.id", "customer.descriptive_name"),
  where = NULL,
  order_by = NULL,
  limit = NULL,
  parameters = NULL,
  login_customer_id = getOption("gads.login.customer.id"),
  include_resource_name = FALSE,
```

```

    cl = NULL,
    verbose = TRUE
  )

```

Arguments

customer_id	Google Ads client customer id, supports a single account id: "xxx-xxx-xxxx" or a vector of ids from the same Google Ads MCC: c("xxx-xxx-xxxx", "xxx-xxx-xxxx")
fields	character vector, list of report fields, all report has own fields list, for example see field list of ad group report .
where	Filter, for example you can filter campaigns by status where = "campaign.status = 'ENABLED'".
order_by	Sorting, character vectors of fields and sorting directions, for example order_by = c("campaign.name DESC", "metrics.clicks").
limit	Maximun rows in report
parameters	Query parameters, for example parameters = "include_drafts=true".
login_customer_id	Google Ads manager customer id
include_resource_name	Get resource names fields in report
cl	A cluster object created by makeCluster , or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).
verbose	Console log output

Value

tibble with ad group dicrionary

See Also

[Google Ads Query Builder](#)

Examples

```

## Not run:
# set client customer id
gads_set_login_customer_id('xxx-xxx-xxxx')

# set manager id if you work under MCC
gads_set_customer_id('xxx-xxx-xxxx')

# load ad groups list
adgroups <- gads_get_ad_groups(
  where = 'ad_group.status = "ENABLED"'
)

```

```
## End(Not run)
```

```
gads_get_ad_group_criteria
```

Get Ad Group Criteria Dictionary From Google Ads Client Account

Description

Get Ad Group Criteria Dictionary From Google Ads Client Account

Usage

```
gads_get_ad_group_criteria(
    customer_id = getOption("gads.customer.id"),
    fields = c("ad_group_criteria.ad_group", "ad_group_criteria.age_range.type",
    "ad_group_criteria.app_payment_model.type", "ad_group_criteria.approval_status",
    "ad_group_criteria.bid_modifier",
    "ad_group_criteria.combined_audience.combined_audience",
    "ad_group_criteria.cpc_bid_micros", "ad_group_criteria.cpm_bid_micros",
    "ad_group_criteria.cpv_bid_micros", "ad_group.id", "customer.id",
    "customer.descriptive_name", "ad_group_criteria.criteria_id",
    "ad_group_criteria.custom_affinity.custom_Affinity",

    "ad_group_criteria.custom_audience.custom_audience",
    "ad_group_criteria.custom_intent.custom_intent",
    "ad_group_criteria.disapproval_reasons", "ad_group_criteria.display_name",
    "ad_group_criteria.effective_cpc_bid_micros",
    "ad_group_criteria.effective_cpc_bid_source",
    "ad_group_criteria.effective_cpm_bid_micros",
    "ad_group_criteria.effective_cpm_bid_source",
    "ad_group_criteria.effective_cpv_bid_micros",
    "ad_group_criteria.effective_cpv_bid_source",
    "ad_group_criteria.effective_percent_cpc_bid_micros",

    "ad_group_criteria.effective_percent_cpc_bid_source",
    "ad_group_criteria.final_mobile_urls", "ad_group_criteria.final_url_suffix",
    "ad_group_criteria.final_urls", "ad_group_criteria.gender.type",
    "ad_group_criteria.income_range.type", "ad_group_criteria.keyword.match_type",
    "ad_group_criteria.keyword.text", "ad_group_criteria.labels",
    "ad_group_criteria.listing_group.case_value.hotel_city.city_criteria",
    "ad_group_criteria.listing_group.case_value.hotel_class.value",
    "ad_group_criteria.listing_group.case_value.hotel_id.value",

    "ad_group_criteria.listing_group.case_value.hotel_state.state_criteria",
    "ad_group_criteria.listing_group.case_value.product_brand.value",
    "ad_group_criteria.listing_group.case_value.product_channel.channel",
```

```

    "ad_group_criterion.listing_group.case_value.product_condition.condition",
    "ad_group_criterion.listing_group.case_value.product_custom_attribute.index",
    "ad_group_criterion.listing_group.case_value.product_custom_attribute.value",
    "ad_group_criterion.listing_group.case_value.product_type.level",

    "ad_group_criterion.listing_group.case_value.product_item_id.value",
    "ad_group_criterion.listing_group.case_value.product_type.value",
    "ad_group_criterion.listing_group.parent_ad_group_criterion",
    "ad_group_criterion.listing_group.type",
    "ad_group_criterion.mobile_app_category.mobile_app_category_constant",
    "ad_group_criterion.mobile_application.app_id",
    "ad_group_criterion.mobile_application.name", "ad_group_criterion.negative",
    "ad_group_criterion.parental_status.type",
    "ad_group_criterion.percent_cpc_bid_micros",

    "ad_group_criterion.placement.url",
    "ad_group_criterion.position_estimates.estimated_add_cost_at_first_position_cpc",
    "ad_group_criterion.position_estimates.estimated_add_clicks_at_first_position_cpc",
    "ad_group_criterion.position_estimates.first_page_cpc_micros",
    "ad_group_criterion.position_estimates.first_position_cpc_micros",
    "ad_group_criterion.position_estimates.top_of_page_cpc_micros",
    "ad_group_criterion.quality_info.creative_quality_score",
    "ad_group_criterion.quality_info.post_click_quality_score",

    "ad_group_criterion.quality_info.quality_score",
    "ad_group_criterion.quality_info.search_predicted_ctr",
    "ad_group_criterion.resource_name", "ad_group_criterion.status",
    "ad_group_criterion.system_serving_status", "ad_group_criterion.topic.path",
    "ad_group_criterion.topic.topic_constant",
    "ad_group_criterion.tracking_url_template", "ad_group_criterion.type",
    "ad_group_criterion.url_custom_parameters",
    "ad_group_criterion.user_interest.user_interest_category",
    "ad_group_criterion.user_list.user_list",

    "ad_group_criterion.webpage.conditions",
    "ad_group_criterion.webpage.coverage_percentage",
    "ad_group_criterion.webpage.criterion_name",
    "ad_group_criterion.webpage.sample.sample_urls",
    "ad_group_criterion.youtube_channel.channel_id",
    "ad_group_criterion.youtube_video.video_id"),
where = NULL,
order_by = NULL,
limit = NULL,
parameters = NULL,
login_customer_id = getOption("gads.login.customer.id"),
include_resource_name = FALSE,
cl = NULL,
verbose = TRUE

```

)

Arguments

customer_id	Google Ads client customer id, supports a single account id: "xxx-xxx-xxxx" or a vector of ids from the same Google Ads MCC: c("xxx-xxx-xxxx", "xxx-xxx-xxxx")
fields	character vector, list of report fields, all report has own fields list, for example see field list of ad group report .
where	Filter, for example you can filter campaigns by status where = "campaign.status = 'ENABLED'".
order_by	Sorting, character vectors of fields and sorting directions, for example order_by = c("campaign.name DESC", "metrics.clicks").
limit	Maximun rows in report
parameters	Query parameters, for example parameters = "include_drafts=true".
login_customer_id	Google Ads manager customer id
include_resource_name	Get resource names fields in report
cl	A cluster object created by makeCluster , or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).
verbose	Console log output

Value

tibble with ad group criterions dicrionary

See Also

[Google Ads Query Builder](#)

Examples

```
## Not run:
# set client customer id
gads_set_login_customer_id('xxx-xxx-xxxx')

# set manager id if you work under MCC
gads_set_customer_id('xxx-xxx-xxxx')

# load ad groups keywords list
kw <- gads_get_ad_group_criteria()

## End(Not run)
```

gads_get_campaigns	<i>Get Campaigns Dictionary From Google Ads Client Account</i>
--------------------	--

Description

Get Campaigns Dictionary From Google Ads Client Account

Usage

```
gads_get_campaigns(
  fields = c("campaign.id", "campaign.name", "campaign.accessible_bidding_strategy",
    "campaign.ad_serving_optimization_status", "campaign.advertising_channel_sub_type",
    "campaign.advertising_channel_type", "campaign.app_campaign_setting.app_id",
    "campaign.app_campaign_setting.app_store", "campaign.base_campaign",
    "campaign.bidding_strategy",
    "campaign.app_campaign_setting.bidding_strategy_goal_type",
    "campaign.campaign_budget", "campaign.bidding_strategy_type",
    "campaign.dynamic_search_ads_setting.language_code",
    "campaign.start_date",
    "campaign.end_date", "campaign.status", "campaign.manual_cpm", "campaign.manual_cpv",
    "campaign.maximize_conversion_value.target_roas",
    "campaign.maximize_conversions.target_cpa_micros",
    "campaign.network_settings.target_content_network",
    "campaign.network_settings.target_google_search",
    "campaign.network_settings.target_partner_search_network",
    "campaign.network_settings.target_search_network",
    "campaign.optimization_goal_setting.optimization_goal_types",
    "campaign.optimization_score",
    "campaign.payment_mode",
    "campaign.serving_status", "campaign.shopping_setting.campaign_priority",
    "campaign.target_roas.target_roas", "campaign.tracking_url_template",
    "customer.descriptive_name", "customer.id"),
  where = NULL,
  order_by = NULL,
  limit = NULL,
  parameters = NULL,
  customer_id = getOption("gads.customer.id"),
  login_customer_id = getOption("gads.login.customer.id"),
  include_resource_name = FALSE,
  cl = NULL,
  verbose = TRUE
)
```

Arguments

fields	character vector, list of report fields, all report has own fields list, for example see field list of campaigns report .
--------	--

where	Filter, for example you can filter campaigns by status where = "campaign.status = 'ENABLED'".
order_by	Sorting, character vectors of fields and sorting directions, for example order_by = c("campaign.name DESC", "metrics.clicks").
limit	Maximun rows in report
parameters	Query parameters, for example parameters = "include_drafts=true".
customer_id	Google Ads client customer id, supports a single account id: "xxx-xxx-xxxx" or a vector of ids from the same Google Ads MCC: c("xxx-xxx-xxxx", "xxx-xxx-xxxx")
login_customer_id	Google Ads manager customer id
include_resource_name	Get resource names fields in report
cl	A cluster object created by makeCluster , or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).
verbose	Console log output

Value

tibble with campaigns dicrionary

See Also

[Google Ads Query Builder](#)

Examples

```
## Not run:
# set client customer id
gads_set_login_customer_id('xxx-xxx-xxxx')

# set manager id if you work under MCC
gads_set_customer_id('xxx-xxx-xxxx')

# load campaing list
camps <- gads_get_campaigns(
  where = "campaign.status = 'ENABLED'"
)

## End(Not run)
```

gads_get_fields	<i>Get resource or field information.</i>
-----------------	---

Description

Get resource or field information.

Usage

```
gads_get_fields(object_name)
```

Arguments

object_name name of resource, resource's field, segmentation field or metric

Value

List of resource or field metadata

See Also

[Resource Metadata API documentation](#)

Examples

```
## Not run:  
ad_group_info <- gads_get_fields("ad_group")  
  
## End(Not run)
```

gads_get_geo_targets	<i>Download CSV of geo targets</i>
----------------------	------------------------------------

Description

Download CSV of geo targets

Usage

```
gads_get_geo_targets(  
  doc_page = "https://developers.google.com/google-ads/api/reference/data/geotargets",  
  file_link = "auto"  
)
```

Arguments

doc_page	Link to Google Ads API Reference page
file_link	Link to csv file, default is 'auto'

Value

data.frame with geo targets dictionary

See Also

[Google Ads Geo Targets document page](#)

Examples

```
## Not run:
geo_dict <- gads_get_geo_targets()

## End(Not run)
```

gads_get_keywords	<i>Get Keyword Dictionary From Google Ads Client Account</i>
-------------------	--

Description

Get Keyword Dictionary From Google Ads Client Account

Usage

```
gads_get_keywords(
  customer_id = getOption("gads.customer.id"),
  fields = c("ad_group_criterion.criterion_id", "ad_group_criterion.keyword.text",
    "ad_group_criterion.keyword.match_type", "ad_group_criterion.status",
    "ad_group_criterion.approval_status", "ad_group_criterion.system_serving_status",
    "ad_group_criterion.quality_info.quality_score",
    "ad_group_criterion.quality_info.creative_quality_score",
    "ad_group_criterion.quality_info.post_click_quality_score", "ad_group.id",
    "ad_group.name", "ad_group.status", "campaign.id", "campaign.name", "customer.id",
    "customer.descriptive_name",
    "metrics.average_cpc", "metrics.average_cost",
    "metrics.ctr", "metrics.bounce_rate"),
  where = NULL,
  order_by = NULL,
  limit = NULL,
  parameters = NULL,
  login_customer_id = getOption("gads.login.customer.id"),
  include_resource_name = FALSE,
  cl = NULL,
  verbose = TRUE
)
```

Arguments

customer_id	Google Ads client customer id, supports a single account id: "xxx-xxx-xxxx" or a vector of ids from the same Google Ads MCC: c("xxx-xxx-xxxx", "xxx-xxx-xxxx")
fields	character vector, list of report fields, all report has own fields list, for example see field list of keyword report .
where	Filter, for example you can filter campaigns by status where = "campaign.status = 'ENABLED'".
order_by	Sorting, character vectors of fields and sorting directions, for example order_by = c("campaign.name DESC", "metrics.clicks").
limit	Maximun rows in report
parameters	Query parameters, for example parameters = "include_drafts=true".
login_customer_id	Google Ads manager customer id
include_resource_name	Get resource names fields in report
cl	A cluster object created by makeCluster , or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).
verbose	Console log output

Value

tibble with Keyword criterions dicrionary

See Also

[Google Ads Query Builder](#)

gads_get_metadata	<i>Get metada of object, RESOURCE, ATTRIBUTE, METRIC or SEGMENT</i>
-------------------	---

Description

Get metada of object, RESOURCE, ATTRIBUTE, METRIC or SEGMENT

Usage

```
gads_get_metadata(
  category = c("RESOURCE", "ATTRIBUTE", "METRIC", "SEGMENT", "ALL"),
  fields = c("name", "category", "data_type", "selectable", "filterable", "sortable",
    "selectable_with", "metrics", "segments", "is_repeated", "type_url", "enum_values",
    "attribute_resources")
)
```

Arguments

category	Object category
fields	Metadata fields

Value

tibble with object metadata important arrays in result:

attributeResources Resources that can be using in resource argument in [gads_get_report](#).

metrics Metrics that are available to be selected with the resource in the field argument in [gads_get_report](#). Only populated for fields where the category is RESOURCE.

segments Segment keys that can be selected with the resource in the field argument in [gads_get_report](#). These segment the metrics specified in the query. Only populated for fields where the category is RESOURCE.

selectableWith Fields that can be selected alongside a given field, when not in the FROM clause. This attribute is only relevant when identifying resources or segments that are able to be selected in a query where they are not included by the resource in the FROM clause. As an example, if we are selecting `ad_group.id` and `segments.date` from `ad_group`, and we want to include attributes from `campaign`, we would need to check that `segments.date` is in the `selectableWith` attribute for `campaign`, since it's being selected alongside the existing `segments.date` field.

See Also

[The Query Builder Blog Series: Part 3 - Creating a Resource Schema and Resource Metadata API documentation](#)

Examples

```
## Not run:
# get resource list
resources <- gads_get_metadata("RESOURCE")

# get list of all objects
metadata <- gads_get_metadata("ALL")

## End(Not run)
```

gads_get_report

Get data from Google Ads API

Description

Get data from Google Ads API

Usage

```

gads_get_report(
  resource = "campaign",
  fields = c("campaign.id", "campaign.name", "customer.id", "customer.descriptive_name",
    "campaign.status", "segments.date", "metrics.all_conversions", "metrics.clicks",
    "metrics.cost_micros", "metrics.ctr", "metrics.impressions",
    "metrics.interaction_rate", "metrics.interactions", "metrics.invalid_clicks"),
  where = NULL,
  order_by = NULL,
  limit = NULL,
  parameters = NULL,
  date_from = Sys.Date() - 15,
  date_to = Sys.Date() - 1,
  during = c(NA, "TODAY", "YESTERDAY", "LAST_7_DAYS", "LAST_BUSINESS_WEEK", "THIS_MONTH",
    "LAST_MONTH", "LAST_14_DAYS", "LAST_30_DAYS", "THIS_WEEK_SUN_TODAY",
    "THIS_WEEK_MON_TODAY", "LAST_WEEK_SUN_SAT", "LAST_WEEK_MON_SUN"),
  customer_id = getOption("gads.customer.id"),
  login_customer_id = getOption("gads.login.customer.id"),
  include_resource_name = FALSE,
  gaql_query = NULL,
  cl = NULL,
  verbose = TRUE
)

```

Arguments

resource	Report type, you can get list of all accessible resource using gads_get_metadata . For more information see link with list of all resources
fields	character vector, list of report fields, all report has own fields list. You can get list of accesible resource fields using gads_get_fields for example see field list of campaign report .
where	Filter, for example you can filter campaigns by status where = "campaign.status = 'ENABLED'".
order_by	Sorting, character vectors of fields and sorting directions, for example order_by = c("campaign.name DESC", "metrics.clicks").
limit	Maximun rows in report
parameters	Query parameters, for example parameters = "include_drafts=true".
date_from	Beginning of date range. Format: 2018-01-01
date_to	End of date rage. Format: 2018-01-10
during	Predefined date range. See documentation for more details.
customer_id	Google Ads client customer id, supports a single account id: "xxx-xxx-xxxx" or a vector of ids from the same Google Ads MCC: c("xxx-xxx-xxxx", "xxx-xxx-xxxx")
login_customer_id	Google Ads manager customer id

include_resource_name	Get resource names fields in report
gaql_query	GAQL Query, you can make it in gads_get_metadata . For more information see Query Builder . If you use gaql_query, you don't need set other query parameters like resource, fields, where, dates etc.
cl	A cluster object created by makeCluster , or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).
verbose	Console log output

Value

tibble with the Google Ads Data.

See Also

- [Official Google Ads API Reports documantation](#)
- [Google Ads Query Builder](#)

Examples

```
## Not run:
# set client id
gads_set_login_customer_id('xxx-xxx-xxxx')

# set manager id if you work under MCC
gads_set_customer_id('xxx-xxx-xxxx')

# default paramas is campaign performance report
campaign_stat <- gads_get_report()

# you can load data from several client accounts at once
# from the same Google Ads MCC
# client ids
accounts <- c('xxx-xxx-xxxx', 'yyy-yyy-yyyy')
# loading data
multi_rep <- gads_get_report(
  date_from = as.Date('2021-06-10'),
  date_to = as.Date('2021-06-17'),
  customer_id = accounts
)

# -----
# using more arguments for other reports
group_report <- gads_get_report(
  customer_id = 4732519773,
  resource    = "ad_group",
  fields = c("ad_group.campaign",
             "ad_group.id",
             "ad_group.name",
```

```

        "ad_group.status",
        "metrics.clicks",
        "metrics.cost_micros"),
date_from = "2021-06-10",
date_to   = "2021-06-17",
where     = "ad_group.status = 'ENABLED'",
order_by  = c("metrics.clicks DESC", "metrics.cost_micros"),
limit     = 30000
)

# -----
# parallel loading mode
# note: you must using login_customer_id agrument in parallel mode
# because oprions gads_set_login_customer_id() does't work in parallel mode loading
library(parallel)

# make core cluster
cl <- makeCluster(4)

# loading data
multi_rep <- gads_get_report(
  date_from      = as.Date('2021-06-10'),
  date_to        = as.Date('2021-06-17'),
  customer_id    = c('111-111-1111',
                     '222-222-2222',
                     '333-333-3333',
                     '444-444-4444',
                     '555-555-5555'),
  login_customer_id = "999-999-9999",
  cl              = cl
)

# stop cluster
stopCluster(cl)

## End(Not run)

```

gads_has_token

Is there a token on hand?

Description

Reports whether rgoogleads has stored a token, ready for use in downstream requests.

Usage

```
gads_has_token()
```

Value

Logical.

See Also

Other low-level API functions: [gads_token\(\)](#)

`gads_keyword_plan_forecast_metrics`

Returns the requested Keyword Plan forecasts.

Description

Returns the requested Keyword Plan forecasts.

Usage

```
gads_keyword_plan_forecast_metrics(  
  keyword_plan_id,  
  customer_id = getOption("gads.customer.id"),  
  login_customer_id = getOption("gads.login.customer.id"),  
  verbose = TRUE  
)
```

Arguments

<code>keyword_plan_id</code>	Keyword plan id, you can get list of your keyword plans using gads_get_report with resource <code>keyword_plan</code>
<code>customer_id</code>	Google Ads client customer id, supports a single account id: "xxx-xxx-xxxx" or a vector of ids from the same Google Ads MCC: c("xxx-xxx-xxxx", "xxx-xxx-xxxx")
<code>login_customer_id</code>	Google Ads manager customer id
<code>verbose</code>	Console log output

Value

tibble with keyword plan historical metrics

See Also

[Keyword Planning API Documentation](#)

Examples

```
## Not run:
# set client id
gads_set_customer_id('xxx-xxx-xxxx')

# set manager id
gads_set_login_customer_id('xxx-xxx-xxxx')

# get list of plan
plan_data <- gads_get_report(
  resource = 'keyword_plan',
  fields = c('keyword_plan.id')
)

# get keyword historical data
historical_plan_data <- gads_keyword_plan_forecast_metrics(
  keyword_plan_id = plan_data$keyword_plan_id[1]# '
)

## End(Not run)
```

gads_keyword_plan_forecast_timeseries

Returns a forecast in the form of a time series for the Keyword Plan over the next 52 weeks.

Description

Returns a forecast in the form of a time series for the Keyword Plan over the next 52 weeks.

Usage

```
gads_keyword_plan_forecast_timeseries(
  keyword_plan_id,
  customer_id = getOption("gads.customer.id"),
  login_customer_id = getOption("gads.login.customer.id"),
  verbose = TRUE
)
```

Arguments

keyword_plan_id	Keyword plan id, you can get list of your keyword plans using gads_get_report with recourse keyword_plan
customer_id	Google Ads client customer id, supports a single account id: "xxx-xxx-xxxx" or a vector of ids from the same Google Ads MCC: c("xxx-xxx-xxxx", "xxx-xxx-xxxx")

login_customer_id
 Google Ads manager customer id
 verbose Console log output

Value

tibble with keyword plan historical metrics

See Also

[Keyword Planning API Documentation](#)

Examples

```
## Not run:
# set client id
gads_set_customer_id('xxx-xxx-xxxx')

# set manager id
gads_set_login_customer_id('xxx-xxx-xxxx')

# get list of plan
plan_data <- gads_get_report(
  resource = 'keyword_plan',
  fields = c('keyword_plan.id')
)

# get keyword historical data
historical_plan_data <- gads_keyword_plan_forecast_timeseries(
  keyword_plan_id = plan_data$keyword_plan_id[1]# '
)

## End(Not run)
```

`gads_keyword_plan_historical_metrics`

Returns the requested Keyword Plan historical metrics.

Description

Returns the requested Keyword Plan historical metrics.

Usage

```
gads_keyword_plan_historical_metrics(
  keyword_plan_id,
  customer_id = getOption("gads.customer.id"),
  login_customer_id = getOption("gads.login.customer.id"),
  verbose = TRUE
)
```

Arguments

keyword_plan_id	Keyword plan id, you can get list of your keyword plans using <code>gads_get_report</code> with resource keyword_plan
customer_id	Google Ads client customer id, supports a single account id: "xxx-xxx-xxxx" or a vector of ids from the same Google Ads MCC: c("xxx-xxx-xxxx", "xxx-xxx-xxxx")
login_customer_id	Google Ads manager customer id
verbose	Console log output

Value

tibble with keyword plan historical metrics

Examples

```
## Not run:
# set client id
gads_set_customer_id('xxx-xxx-xxxx')

# set manager id
gads_set_login_customer_id('xxx-xxx-xxxx')

# get list of plan
plan_data <- gads_get_report(
  resource = 'keyword_plan',
  fields = c('keyword_plan.id')
)

# get keyword historical data
historical_plan_data <- gads_keyword_plan_historical_metrics(
  keyword_plan_id = plan_data$keyword_plan_id[1]#
)

# main plan data
data <- historical_plan_data$main_data
historical_data <- historical_plan_data$historical_data

## End(Not run)
```

`gads_last_request_ids` *Get last API request ID for Google Ads API support ticket*

Description

Get last API request ID for Google Ads API support ticket

Usage

```
gads_last_request_ids()
```

Value

Request ID

Examples

```
## Not run:  
gads_last_request_ids()  
  
## End(Not run)
```

gads_set_customer_id *Set client customer id in current R session*

Description

Set client customer id in current R session

Usage

```
gads_set_customer_id(customer_id)
```

Arguments

customer_id your client customer id

Value

only set options

gads_set_login_customer_id
 Set manager customer id in current R session

Description

Set manager customer id in current R session

Usage

```
gads_set_login_customer_id(customer_id)
```

Arguments

customer_id your manager customer id

Value

only set options

gads_token	<i>Produce configured token</i>
------------	---------------------------------

Description

For internal use or for those programming around the Google Ads API. Returns a token pre-processed with [httpr::config\(\)](#). Most users do not need to handle tokens "by hand" or, even if they need some control, [gads_auth\(\)](#) is what they need. If there is no current token, [gads_auth\(\)](#) is called to either load from cache or initiate OAuth2.0 flow. If auth has been deactivated via [gads_deauth\(\)](#), [gads_token\(\)](#) returns NULL.

Usage

```
gads_token()
```

Value

A request object (an S3 class provided by [httpr](#)).

See Also

Other low-level API functions: [gads_has_token\(\)](#)

gads_user	<i>Get info on current user</i>
-----------	---------------------------------

Description

Reveals the email address of the user associated with the current token. If no token has been loaded yet, this function does not initiate auth.

Usage

```
gads_user()
```

Value

An email address or, if no token has been loaded, NULL.

See Also

[gargle::token_userinfo\(\)](#), [gargle::token_email\(\)](#), [gargle::token_tokeninfo\(\)](#)

Index

* auth functions

- `gads_auth`, 4
- `gads_auth_configure`, 6
- `gads_deauth`, 10

* low-level API functions

- `gads_has_token`, 28
- `gads_token`, 34

- `gads_api_key (gads_auth_configure)`, 6
- `gads_api_key()`, 10
- `gads_auth`, 4, 7, 10
- `gads_auth()`, 6, 34
- `gads_auth_cache_path`
 - `(gads_auth_configure)`, 6
- `gads_auth_configure`, 5, 6, 10
- `gads_auth_configure()`, 5, 10
- `gads_check_errors`, 8
- `gads_customer`, 8
- `gads_customer_id_from_env`, 9
- `gads_customer_id_to_env`, 9
- `gads_deauth`, 5, 7, 10
- `gads_deauth()`, 6, 34
- `gads_developer_token`
 - `(gads_auth_configure)`, 6
- `gads_fix_names`, 10
- `gads_get_accessible_customers`, 11
- `gads_get_account_hierarchy`, 11
- `gads_get_ad_group_criteria`, 17
- `gads_get_ad_groups`, 15
- `gads_get_ads`, 12
- `gads_get_campaigns`, 20
- `gads_get_fields`, 22, 26
- `gads_get_geo_targets`, 22
- `gads_get_keywords`, 23
- `gads_get_metadata`, 24, 26, 27
- `gads_get_report`, 25, 25, 29, 30, 32
- `gads_has_token`, 28, 34
- `gads_keyword_plan_forecast_metrics`, 29
- `gads_keyword_plan_forecast_timeseries`, 30

- `gads_keyword_plan_historical_metrics`, 31
- `gads_last_request_ids`, 32
- `gads_oauth_app (gads_auth_configure)`, 6
- `gads_open_auth_cache_folder`
 - `(gads_auth_configure)`, 6
- `gads_set_customer_id`, 33
- `gads_set_login_customer_id`, 33
- `gads_token`, 29, 34
- `gads_user`, 34
- `gargle::AuthState`, 7
- `gargle::gargle_oauth_client_from_json()`, 7
- `gargle::gargle_options`, 5
- `gargle::token_email()`, 34
- `gargle::token_fetch()`, 4, 5
- `gargle::token_tokeninfo()`, 34
- `gargle::token_userinfo()`, 34
- `httr`, 34
- `httr::config()`, 34
- `jsonlite::fromJSON()`, 7
- `makeCluster`, 14, 16, 19, 21, 24, 27
- `rgoogleads (rgoogleads-package)`, 2
- `rgoogleads-package`, 2
- `Token2.0`, 4, 5