Package ‘adobeanalyticsr’

March 1, 2021

Type Package

Version 0.1.5

Title R Client for 'Adobe Analytics' API 2.0

Description
Connect to the 'Adobe Analytics' API v2.0 <https://github.com/AdobeDocs/analytics-2.0-apis>
which powers 'Analysis Workspace'. The package was developed
with the analyst in mind, and it will continue to be
developed with the guiding principles of iterative,
repeatable, timely analysis.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.2.0)

Imports assertthat (>= 0.2.0), jsonlite (>= 1.5), dplyr (>= 0.8.1), stringr (>= 1.4.0), purrrlyr (>= 0.0.6), purrr (>= 0.3.3), httr (>= 1.3.1), tidyr (>= 1.0.0), rlang (>= 0.4.8), lubridate (>= 1.7.9), ggplot2 (>= 3.3.2), scales (>= 1.1.1), lifecycle, glue, tibble, knitr

RoxygenNote 7.1.1

Suggests testthat, rmarkdown

RdMacros lifecycle

VignetteBuilder knitr

BugReports https://github.com/benrwoodard/adobeanalyticsr/issues

URL https://github.com/benrwoodard/adobeanalyticsr

NeedsCompilation no

Author Ben Woodard [aut, cre],
Tim Wilson [aut, ctb],
Mark Edmondson [ctb]

Maintainer Ben Woodard <benrwoodard@gmail.com>

Repository CRAN

Date/Publication 2021-03-01 09:10:10 UTC

R topics documented:

aw_anomaly_report *Anomaly Report*

Description

Get an anomaly report for one or more metrics

Usage

```
aw_anomaly_report(
  company_id = Sys.getenv("AW_COMPANY_ID"),
  rsid = Sys.getenv("AW_REPORTSUITE_ID"),
  date_range = c(Sys.Date() - 31, Sys.Date() - 1),
  metrics,
  granularity = "day",
  segmentId = NA,
  quickView = FALSE,
  anomalyDetection = TRUE,
  countRepeatInstances = TRUE,
  debug = FALSE
)
```

Arguments

`company_id`	Company Id. Taken from the global environment by default if not provided.
`rsid`	Adobe report number
`date_range`	A two length vector of start and end Date objects (default set to show last 30 days)
`metrics`	Metric to request the anomaly detection. If multiple metrics, each metric and date will have it's own row.

`granularity` Use either hour, day (default), week, or month

`segmentId` Use segments to globally filter the results. Use 1 or many.

`quickView` Return a list of 3 lists per metric. 1. All Data 2. Data filtered to include only anomalous rows 3. Interactive ggplot line graph

`anomalyDetection`
logical statement for including anomaly. Default is TRUE

`countRepeatInstances`
Should the data include repeat instances

`debug` default is FALSE but set to TRUE to see the json request being sent to the Adobe API

Value

If quickView = 'FALSE' (default) then a data frame including the day, metric, data, dataExpected, dataUpperBound, dataLowerBound, and dataAnomalyDetected will be returned. If quickView = 'TRUE' then a list of three lists will be returned. The first list will be a data frame including all the default columns. The second list item will be a filtered data frame that includes rows where dataAnomalyDetected = 'TRUE'. The third list item is a visual made using 'ggplot2' with the error band and points where the dataAnomalyDetected = 'TRUE'. If more than one metric is in the request and quickView is set to TRUE then the lists will be split by each metric requested.

`aw_freeform_table` *Get a freeform table*

Description

Get a report analogous to a **Freeform Table** visualization in Analysis Workspace. The function uses the arguments to construct and execute a JSON-based query to the Adobe Analytics API and then returns the results as a data frame.

Usage

```
aw_freeform_table(
  company_id = Sys.getenv("AW_COMPANY_ID"),
  rsid = Sys.getenv("AW_REPORTSUITE_ID"),
  date_range = c(Sys.Date() - 30, Sys.Date() - 1),
  dimensions = c("page", "lasttouchchannel", "mobiledevicetype"),
  metrics = c("visits", "visitors"),
  top = c(5),
  page = 0,
  filterType = "breakdown",
  segmentId = NA,
  metricSort = "desc",
  include_unspecified = TRUE,
  search = NA,
  prettynames = FALSE,
  debug = FALSE
)
```

Arguments

`company_id` Company ID. If an environment variable called `AW_COMPANY_ID` exists in `.Renviron` or elsewhere and no `company_id` argument is provided, then the `AW_COMPANY_ID` value will be used. Use `get_me` to get a list of available `company_id` values.

`rsid` Adobe report suite ID (RSID). If an environment variable called `AW_REPORTSUITE_ID` exists in `.Renviron` or elsewhere and no `rsid` argument is provided, then the `AW_REPORTSUITE_ID` value will be used. Use `aw_get_reportsuites` to get a list of available `rsid` values.

`date_range` A vector containing the start and end date for the report as **Date** objects.

`dimensions` A character vector of dimensions. There is currently a limit of 20 dimension breakdowns. Each dimension value that gets broken down by another dimension requires an additional API call, so the more dimensions that are included, the longer the function will take to return results. This is how the Adobe Analytics API works. Use `aw_get_dimensions` to get a list of available `dimensions` IDs.

`metrics` A character vector of metrics. Use `aw_get_metrics` and `aw_get_calculatedmetrics` to get a list of available `metrics` IDs.

`top` The number of values to be pulled for each dimension. The default is 5 and the "top" is based on the first `metric` value (along with `metricSort`). If there are multiple dimensions, then this argument can either be a vector that includes the number of values to include at each level (each breakdown) or, if a single value is used, then that will be the maximum number of values to return at each level. See the **Details** for information on the unique handling of `daterange...` values.

`page` Used in combination with `top` to return the next page of results. Uses 0-based numbering (e.g., `top = 50000` and `page = 1` will return the top 50,000 items *starting at 50,001*).

`filterType` This is a placeholder argument for use as additional functionality is added to the package. Currently, it defaults to `breakdown`, and that is the only supported value.

`segmentId` A single segment ID or a vector of multiple segment IDs to apply to the overall report. If multiple `segmentId` values are included, the segments will be effectived ANDed together, just as if multiple segments were added to the header of an Analysis Workspace panel. Use `aw_get_segments` to get a list of available `segmentId` values.

`metricSort` Pre-sorts the table by metrics. Values are either `asc` (ascending) or `desc` (descending).

`include_unspecified`
Whether or not to include **Unspecified** values in the results. This is the equivalent of the **Include Unspecified (None)** checkbox in freeform tables in Analysis Workspace. This defaults to `TRUE`, which includes **Unspecified** values in the results.

`search` Criteria to filter the results by one or more dimensions. Searches are case-insenstive. Refer to the **Details** for more information on constructing values for this argument.

`prettynames` A Boolean that determines whether the column names in the results use the API field name (e.g., "mobiledevicetype", "pageviews") or the "pretty name" for the field (e.g., "Mobile Device Type", "Page Views"). This applies to both dimensions and metrics. The default value is `FALSE`, which returns the API field names. For custom eVars, props, and events, the non-pretty values are simply the variable number (e.g., "evar2", "prop3", "event15").

`debug` Set to `TRUE` to publish the full JSON request(s) being sent to the API to the console when the function is called. The default is `FALSE`.

Details

This function is based on the **Freeform Table** visualization in Analysis Workspace. It is accessing the same API call type that is used to generate those visualizations.

Dimension Ordering

Adobe Analytics only queries one dimension at a time, even though the results get returned in a single data frame (or table in the case of Analysis Workspace). The more dimensions are included in the report–the more breakdowns of the data–the more queries are required. As a result, the *order* of the dimensions *can* have a dramatic impact on the total query time, even if the resulting data is essentially identical.

One way to understand this is to consider how much dragging and dropping would be required to return the data in Analysis Workspace *if you were not able to <Shift>-<click> to highlight multiple values before dragging a new dimension to break down existing values*.

Consider a scenario where you are pulling metrics for the last 30 days (`daterangeday`) for **Mobile Device Type** (`mobiledevicetype`), which has 7 unique values. Setting `dimensions = c("daterangeday","mobiledevicet` would make one query to get the values of the 30 days included. The query would then run a separate query for *each of those 30 days* to get the `mobiledevicetype` results for each day. So, this would be **31 API calls**.

If, instead, the function was called with the `dimension` values reversed (`dimensions = c("mobiledevicetype","daterange` then the first query would return the 7 `mobiledevicetype` values, and then would run an additional query for each of those *7 mobile device type values* to return the results for the 30 days within each device type. This would be only **7 API calls**.

Strategically ordering dimensions–and then wrangling the resulting data set as needed–is one of the best ways to improve query performance.

Date Handling

Date handling has several special characteristics that are worth getting familiar with:

- The API names for day, week, month, etc. are prepended with `daterange`, so daily data uses `daterangeday`, weekly data uses `daterangeweek`, monthly data uses `daterangemonth`, etc.
- When setting the argument for `top`, if the first (or only) `dimension` value is a `daterange...` object, then, if this argument is not explicitly specified *or* if it uses only a single value (e.g., `top = 10`), the function will still return all of the values that fall in that date range. For instance, if the `date_range` was set for a 30-day period and the first `dimension` value was `daterangeday`, *and* no value is specified for `top`, rather than simply returning the first 5 dates in the range, all 30 days will be returned. In the same scenario, if `top = 10` was set, then all 30 days would still be returned, and the `10` would simply be applied to the additional dimensions.

- If you want to return all of the date/time values but then have specific control over the number of values returned for each of the drilldown dimensions, then set `0` as the first value in the `top` argument and then specify different numbers for each breakdown (e.g., `top = c(0,3,10)` would return all of the date/time values for the specified `date_range`, the top 3 values for the second specified `dimension`, and then the top 10 values for each of the next dimension's results).
- If you are using a `daterange...` value *not* as the first dimension, then simply using `0` at the same level in the `top` argument specification will return all of the values for that date/time value.

Search/Filtering

There are powerful filtering abilities within the function. However, to support that power requires a syntax that can feel a bit cumbersome for simple queries. ***Note:*** search filters are case-insensitive. This is Adobe Analytics API functionality and can not be specified otherwise in queries.

The `search` argument takes a vector of search strings, with each value in the vector corresponding to the `dimension` value that is at the same position. These search strings support a range of operators, including `AND`, `OR`, `NOT`, `MATCH`, `CONTAINS`, `BEGINS-WITH`, and `ENDS-WITH`.

The default for any search string is to use `CONTAINS`. Consider a query where `dimensions = c("mobiledevicetype","lastt`

- `search = "CONTAINS 'mobile'"` will return results where `mobiledevicetype` contains "mobile", so would return all rows for **Mobile Phone**.
- This could be shortened to `search = "'mobile'"` and would behave exactly the same, since `CONTAINS` is the default operator
- `search = c("CONTAINS 'mobile'","CONTAINS 'search'")` will return results where `mobiledevicetype` contains "mobile" and, within those results, results where `lasttouchchannel` contains "search".
- `search = c("(CONTAINS 'mobile') OR (CONTAINS 'tablet')","(MATCH 'paid search')")` will return results where `mobiledevicetype` contains "mobile" *or* "tablet" and, within those results, will only include results where `lasttouchchannel` exactly matches "paid search" (but is case-insensitive, so would return "Paid Search" values).

Value

A data frame with dimensions and metrics.

See Also

`get_me`, `aw_get_reportsuites`, `aw_get_segments`, `aw_get_dimensions`, `aw_get_metrics`, `aw_get_calculatedmetrics`

`aw_get_calculatedmetrics` *Get a list of calculated metrics.*

Description

Retrieve a list of available calculated metrics. The results will always include these default items: `id`, `name`, `description`, `rsid`, `owner`, `polarity`, `precision`, `type`. Other attributes can be optionally requested through the `expansion` field.

Usage

```
aw_get_calculatedmetrics(
  company_id = Sys.getenv("AW_COMPANY_ID"),
  rsids = NA,
  ownerId = NA,
  filterByIds = NA,
  toBeUsedInRsid = NA,
  locale = "en_US",
  name = NA,
  tagNames = NA,
  favorite = NA,
  approved = NA,
  limit = 1000,
  page = 0,
  sortDirection = "DESC",
  sortProperty = NA,
  expansion = NA,
  includeType = "all",
  debug = FALSE
)
```

Arguments

`company_id` Company ID. If an environment variable called `AW_COMPANY_ID` exists in `.Renviron` or elsewhere and no `company_id` argument is provided, then the `AW_COMPANY_ID` value will be used. Use `get_me` to get a list of available `company_id` values.

`rsids` Filter the list to only include calculated metrics tied to a specified RSID or list of RSIDs. Specify multiple RSIDs as a vector (i.e., "rsids = c("rsid_1", rsid_2",...rsid_n")"). Use `aw_get_reportsuites` to get a list of available `rsid` values.

`ownerId` Filter the list to only include calculated metrics owned by the specified loginId.

`filterByIds` Filter the list to only include calculated metrics in the specified list as specified by a single string or as a vector of strings.

`toBeUsedInRsid` The report suite where the calculated metric is intended to be used. This report suite is used to determine things like compatibility and permissions. If it is not specified, then the permissions will be calculated based on the union of all metrics authorized in all groups the user belongs to. If **compatibility** is specified for `expansion`, and `toBeUsedInRsid` is not, then the compatibility returned is based off of the compatibility from the last time the calculated metric was saved.

`locale` The locale that system-named metrics should be returned in. Non-localized values will be returned for title, name, description, etc. if a localized value is not available.

`name` Filter the list to only include calculated metrics that contain the specified **name**. This is case-insensitive and is a simple, single string match.

`tagNames` Filter the list to only include calculated metrics that contain one of the tags as specified by a single string or vector of strings.

`favorite` Set to `TRUE` to only include calculated metrics that are favorites in the results. A value of `FALSE` will return all calculated metrics, including those that are favorites.

`approved` Set to `TRUE` to only include calculated metrics that are approved in the results. A value of `FALSE` will return all calculated metrics, including those that are approved and those that are not.

`limit` The number of results to return per page. The default is 1,000.

`page` The "page" of results to display. This works in conjunction with the `limit` argument and is zero-based. For instance, if `limit = 10` and `page = 1`, the results returned would be 11 through 20.

`sortDirection` The sort direction for the results: `ASC` (default) for ascending or `DESC` for descending. (This is case insensitive, so `asc` and `desc` work as well.)

`sortProperty` The property to sort the results by. Currently available values are `id` (default), `name`, and `modified_date`. Note that setting `expansion = modified` returns results with a column added called `modified`, which is the last date the calculated metric was modified. When using this value for `sortProperty`, though, the name of the argument is `modified_date`, because why would we expect locked-in consistency from Adobe?

`expansion` Additional calculated metric metadata fields to include in the results: `reportSuiteName`, `ownerFullName`, `modified`, `tags`, `definition`, `compatability`, `categories`. See **Details** for more information about the quirks of this argument.

`includeType` Include additional calculated metrics not owned by user. Available values are `all` (default), `shared`, and `templates`. The `all` option takes precedence over "shared"

`debug` Include the output and input of the api call in the console for debugging. Default is FALSE

Details

This function is useful/needed to identify the specific ID of a calculated metric for use in other functions like `aw_freeform_report`.

The `expansion` argument accepts the following values, which will then include additional columns in the results:

- **ownerFullName**: adds `owner.name` and `owner.login` columns to the results (`owner.id` is already included by default).
- **modified**: adds a `modified` column to the output with the date (ISO 8601 format) each calculated metric was last modified.
- **definition**: adds *multiple* columns (the number will vary based on the number and complexity of calculated metrics returns) that provide the actual formula for each of the calculated metrics. This is returned from the API as a JSON object and converted into columns by the function, which means it is pretty messy, so, really, it’s not recommended that you use this value.
- **compatability**: should add a column with the products that the metric is compatible with, but this behavior has not actually been shown to be true, so this may actually do nothing if included.

- **reportSuiteName**: adds a `reportSuiteName` and a `siteTitle` column with the friendly report suite name for the RSID.
- **tags**: adds a column with an embedded data frame with all of the existing tags that are associated with the calculated metric. This can be a bit messy to work with, but the information is, at least, there.

Multiple values for `expansion` can be included in the argument as a vector. For instance, `expansion = c("tags","modified")` will add both a `tags` column and a `modified` column to the output.

Value

A data frame of calculated metrics and their metadata.

See Also

`aw_get_metrics`

`aw_get_dimensions` *Get list of dimensions*

Description

This will generate an extensive list of all the dimensions in the reportsuite.

Usage

```
aw_get_dimensions(
  rsid = Sys.getenv("AW_REPORTSUITE_ID"),
  locale = "en_US",
  segmentable = FALSE,
  reportable = FALSE,
  classifiable = FALSE,
  expansion = NA,
  debug = FALSE,
  company_id = Sys.getenv("AW_COMPANY_ID")
)
```

Arguments

`rsid`	Adobe report suite ID (RSID). If an environment variable called `AW_REPORTSUITE_ID` exists in `.Renviron` or elsewhere and no `rsid` argument is provided, then the `AW_REPORTSUITE_ID` value will be used. Use `aw_get_reportsuites` to get a list of available `rsid` values.
`locale`	The locale that dimension details should be returned in. The default is `en_US`.
`segmentable`	Boolean that determines whether or not to include dimensions that can be used in segments. `FALSE` (the default) returns *all* dimensions (*not* just the non-segmentable ones). Examples of dimensions that cannot be used in segments are `clickmapaction`, `codeversion`, `newvisit`, and `pageurl`.

`reportable` Boolean that determines whether or not to include dimensions that can be used in reports `FALSE` (the default) returns *all* dimensions (*not* just the non-segmentable ones).

`classifiable` Boolean that determines whether or not to include dimensions that can be used in classifications `FALSE` (the default) returns *all* dimensions (*not* just the non-segmentable ones).

`expansion` Additional dimension metadata to include in the results: `tags`, `allowedForReporting`, and `categories`. This argument takes a single value (e.g., `expansion = "tags"`) or a vector of values (e.g., `expansion = c("tags","categories")`).

`debug` Include the output and input of the api call in the console for debugging. Default is FALSE

`company_id` Company ID. If an environment variable called `AW_COMPANY_ID` exists in `.Renviron` or elsewhere and no `company_id` argument is provided, then the `AW_COMPANY_ID` value will be used. Use `get_me` to get a list of available `company_id` values.

Value

A data frame of dimensions and their meta data.

`aw_get_metrics` *Get list of metrics*

Description

Get a data frame with all of the standard (non-calculated) metrics (measures) in the report suite.

Usage

```
aw_get_metrics(
  rsid = Sys.getenv("AW_REPORTSUITE_ID"),
  locale = "en_US",
  segmentable = "NULL",
  expansion = NA,
  company_id = Sys.getenv("AW_COMPANY_ID"),
  debug = FALSE,
  use_oob = TRUE
)
```

Arguments

`rsid` Adobe report suite ID (RSID). If an environment variable called `AW_REPORTSUITE_ID` exists in `.Renviron` or elsewhere and no `rsid` argument is provided, then the `AW_REPORTSUITE_ID` value will be used. Use `aw_get_reportsuites` to get a list of available `rsid` values.

`locale` The locale that system-named metrics should be returned in. Non-localized values will be returned for title, name, description, etc. if a localized value is not available.

`segmentable` Boolean that determines whether or not to include metrics that can be used in segments. `NULL` (the default) and `FALSE` return *all* metrics (*not* just the non-segmentable ones). Examples of metrics that cannot be used in segments are `bounces`, bounce rate, `entries`, and `visitors`.

`expansion` Additional metrics metadata to include in the results: `tags`, `allowedForReporting`, and `categories`. This argument takes a single value (e.g., `expansion = "tags"`) or a vector of values (e.g., `expansion = c("tags","categories")`).

`company_id` Company ID. If an environment variable called `AW_COMPANY_ID` exists in `.Renviron` or elsewhere and no `company_id` argument is provided, then the `AW_COMPANY_ID` value will be used. Use `get_me` to get a list of available `company_id` values.

`debug` Include the output and input of the api call in the console for debugging. Default is FALSE

`use_oob` Always set to TRUE. Needed for tests

Details

This function is commonly used to get the correct ID for a specific metric or metrics that will be used in other function calls. The results returned are:

- All of the "out of the box" metrics like visits, page views, visitors, orders, revenue, bounce rate, etc.
- All of the enabled events that are configured in the report suite.
- An instances metric for each enabled eVar.

This function does *not* return calculated metrics.

Value

A data frame of metrics (excluding calculated metrics) and their meta data.

See Also

`aw_get_calculatedmetrics`

`aw_get_reportsuites` *Get list of report suites*

Description

Retrieve a list of report suites and meta data about each one.

Usage

```
aw_get_reportsuites(
  company_id = Sys.getenv("AW_COMPANY_ID"),
  rsids = "",
  rsidContains = "",
  limit = 10,
  page = 0,
  expansion = NA,
  debug = FALSE
)
```

Arguments

company_id	Company ID. If an environment variable called AW_COMPANY_ID exists in .Renviron or elsewhere and no company_id argument is provided, then the AW_COMPANY_ID value will be used. Use get_me to get a list of available company_id values.
rsids	Filter the results to include one or more specific report suites. Specify multiple RSIDs as a vector (i.e., "rsids = c("rsid_1", rsid_2",...rsid_n")").
rsidContains	Filter the results list to only include suites that contain the specified string within the RSID. This is case-insensitive and is a simple, single string match.
limit	The number of results to return per page. This argument works in conjunction with the page argument. The default is 10.
page	The "page" of results to display. This works in conjunction with the limit argument and is zero-based. For instance, if limit = 20 and page = 1, the results returned would be 21 through 40.
expansion	Additional segment metadata fields to include in the results: name, parentRsid, currency, calendarType, timezoneZoneinfo. This argument takes a single value (e.g., expansion = "name") or a vector of values (e.g., expansion = c("name","currency")).
debug	Include the output and input of the api call in the console for debugging. Default is FALSE

Value

A data frame of report suites and their meta data.

aw_get_segments *Get list of segments*

Description

Retrieve all segments

Usage

```
aw_get_segments(
  company_id = Sys.getenv("AW_COMPANY_ID"),
  rsids = NA,
  segmentFilter = NA,
  locale = "en_US",
  name = NA,
  tagNames = NA,
  filterByPublishedSegments = "all",
  limit = 10,
  page = 0,
  sortDirection = "ASC",
  sortProperty = "id",
  expansion = NA,
  includeType = "all",
  debug = FALSE
)
```

Arguments

`company_id` Company ID. If an environment variable called `AW_COMPANY_ID` exists in `.Renviron` or elsewhere and no `company_id` argument is provided, then the `AW_COMPANY_ID` value will be used. Use `get_me` to get a list of available `company_id` values.

`rsids` Filter the list to only include segments tied to a specified RSID or list of RSIDs. Specify multiple RSIDs as a vector (i.e., "rsids = c("rsid_1", rsid_2",...rsid_n")"). Use `aw_get_reportsuites` to get a list of available `rsid` values.

`segmentFilter` Filter list to only include suites in this list of segment IDs (comma-delimited)

`locale` The locale that segment details should be returned in. The default is `en_US`.

`name` Filter the list to only include segments that contain the specified **name**. This is case-insensitive and is a simple, single string match.

`tagNames` Filter the list to only include segments that contain one of the tags.

`filterByPublishedSegments`
Filter the list to only include segments where the published field is set to one of the allowable values: `all` (the default), `TRUE`, or `FALSE`.

`limit` The number of results to return per page. This argument works in conjunction with the page argument. The default is 10.

`page` The "page" of results to display. This works in conjunction with the `limit` argument and is zero-based. For instance, if `limit = 20` and `page = 1`, the results returned would be 21 through 40.

`sortDirection` The sort direction for the results: `ASC` (default) for ascending or `DESC` for descending. (This is case insensitive, so `asc` and `desc` work as well.)

`sortProperty` The property to sort the results by. Currently available values are `id` (default), `name`, and `modified_date`. Note that setting `expansion = modified` returns results with a column added called `modified`, which is the last date the calculated metric was modified. When using this value for `sortProperty`, though, the name of the argument is `modified_date`, because why would we expect locked-in consistency from Adobe?

`expansion` Additional segment metadata fields to include in the results: `reportSuiteName`, `ownerFullName`, `modified`, `tags`, `compatibility`, `definition`, `publishingStatus`, `definitionLastModified`, and `categories`. This argument takes a single value (e.g., `expansion = "modified"`) or a vector of values (e.g., `expansion = c("modified","ownerFullName")`).

`includeType` Include additional segments not owned by the user. Available values are `all` (default), `shared`, and `templates`. The `all` option takes precedence over "shared".

`debug` Include the output and input of the api call in the console for debugging. Default is FALSE

Value

A data frame of segments and their meta data.

aw_token *OAuth2 Token for Adobe Analytics*

Description

OAuth2 Token for Adobe Analytics

Usage

```
aw_token(
  client_id = Sys.getenv("AW_CLIENT_ID"),
  client_secret = Sys.getenv("AW_CLIENT_SECRET"),
  use_oob = TRUE
)
```

Arguments

`client_id` defined by global variable or manually defined

`client_secret` defined by global variable or manually defined

`use_oob` for the purpose of testing. Default is set to TRUE

Value

An authorization token is saved the file name aa.oauth. If the file aa.oauth does not exist then one will be created at the end of the authorization process.

aw_workspace_report *Use a prebuilt json query to pull a ranked report*

Description

Organizes the arguments into a json string and then structures the data after the internal function makes the api call. Only runs a single dimension with as many metrics as you want.

Usage

```
aw_workspace_report(
  req_body = "",
  company_id = Sys.getenv("AW_COMPANY_ID"),
  client_id = Sys.getenv("AW_CLIENT_ID"),
  client_secret = Sys.getenv("AW_CLIENT_SECRET")
)
```

Arguments

`req_body`	The json string copied from Workspace
`company_id`	Company Id. Taken from the global environment by default if not provided.
`client_id`	Set in environment args, or pass directly here
`client_secret`	Set in environment args, or pass directly here

Value

A data frame of dimensions and metrics

get_me *Get Company Ids*

Description

This function will quickly pull the list of company ids that you have access to.

Usage

```
get_me(
  req_path = "discovery/me",
  client_id = Sys.getenv("AW_CLIENT_ID"),
  client_secret = Sys.getenv("AW_CLIENT_SECRET")
)
```

Arguments

`req_path` The endpoint for that particular report

`client_id` Set in environment args, or pass directly here

`client_secret` Set in environment args, or pass directly here

Value

A data frame of company ids and company names

Examples

```
## Not run:
get_me()

## End(Not run)
```

get_usage_logs *Get a list of user usage*

Description

This function returns the usage and access logs for a given date range within a 3 month period. The user must have Admin Console / Logs permissions (must be able to view the **Usage & Access Log** data in the web interface) in order to use this function.

Usage

```
get_usage_logs(
  startDate = Sys.Date() - 91,
  endDate = Sys.Date() - 1,
  login = NA,
  ip = NA,
  rsid = NA,
  eventType = NA,
  event = NA,
  limit = 100,
  page = 0,
  company_id = Sys.getenv("AW_COMPANY_ID")
)
```

Arguments

`startDate` Start date for the maximum of a 3 month period.

`endDate` End date for the maximum of a 3 month period.

`login` The login value of the user you want to filter logs by.

`ip` The IP address you want to filter logs by.

rsid The report suite ID you want to filter logs by.

eventType The numeric id for the event type you want to filter logs by. Leaving this blank returns all events. See the Usage Logs API Guide for a complete list of event types.

event The event description you want to filter logs by. No wildcards are permitted.

limit The number of results to return per page. This argument works in conjunction with the page argument. The default is 10.

page The "page" of results to display. This works in conjunction with the limit argument and is zero-based. For instance, if limit = 20 and page = 1, the results returned would be 21 through 40.

company_id Company ID. If an environment variable called AW_COMPANY_ID exists in .Renviron or elsewhere and no company_id argument is provided, then the AW_COMPANY_ID value will be used. Use get_me to get a list of available company_id values.

Value

A data frame of logged events and the event meta data.

Examples

```
## Not run:
get_usage_logs(startDate = Sys.Date()-91, endDate = Sys.Date()-1, limit = 100, page = 0)

## End(Not run)
```

get_users *Get list of users*

Description

Retrieves a list of all users for the company designated by the auth token.

Usage

```
get_users(company_id = Sys.getenv("AW_COMPANY_ID"), limit = 10, page = 0)
```

Arguments

company_id Company ID. If an environment variable called AW_COMPANY_ID exists in .Renviron or elsewhere and no company_id argument is provided, then the AW_COMPANY_ID value will be used. Use get_me to get a list of available company_id values.

limit The number of results to return per page. This argument works in conjunction with the page argument. The default is 10.

page The "page" of results to display. This works in conjunction with the limit argument and is zero-based. For instance, if limit = 20 and page = 1, the results returned would be 21 through 40.

Value

A data frame of users and their meta data.

Examples

```
## Not run:
get_users(limit = 10, page = 0)

## End(Not run)
```

Index