Package 'adobeanalyticsr'

April 5, 2022

Type Package

Version 0.3.2

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), 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), R6, jose, openssl, lifecycle, glue, tibble, vctrs, progress, memoise, utils

Suggests knitr, testthat (>= 3.0.0), rmarkdown

RoxygenNote 7.1.2

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], Charles Gallagher [ctb], Mark Edmondson [ctb]

Maintainer Ben Woodard <benrwoodard@gmail.com>

Repository CRAN

Date/Publication 2022-04-05 18:12:29 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_auth *Generate an Access Token for the Adobe Analytics v2.0 API*

Description

Note: `aw_auth()` is the primary function used for authorization. `auth_oauth()` and `auth_jwt()` should typically not be called directly.

Usage

```
aw_auth(type = aw_auth_with(), ...)

auth_jwt(
  file = Sys.getenv("AW_AUTH_FILE"),
  private_key = Sys.getenv("AW_PRIVATE_KEY"),
  jwt_token = NULL,
  ...
)

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

Arguments

type	Either 'jwt' or 'oauth'. This can be set explicitly, but a best practice is to run `aw_auth_with()` to set the authorization type as an environment variable before running `aw_auth()`
...	Additional arguments passed to auth functions.
file	A JSON file containing service account credentials required for JWT authentication. This file can be downloaded directly from the Adobe Console, and should minimally have the fields `API_KEY`, `CLIENT_SECRET`, `ORG_ID`, and `TECHNICAL_ACCOUNT_ID`.
private_key	Filename of the private key for JWT authentication.
jwt_token	*(Optional)* A custom, encoded, signed JWT claim. If used, `client_id` and `client_secret` are still required.
client_id	The client ID, defined by a global variable or manually defined
client_secret	The client secret, defined by a global variable or manually defined
use_oob	if `FALSE`, use a local webserver for the OAuth dance. Otherwise, provide a URL to the user and prompt for a validation code. Defaults to the value of the `httr_oob_default` default, or TRUE if `httpuv` is not installed.

Value

The path of the cached token. This is returned invisibly.

Functions

- `auth_jwt`: Authenticate with JWT token
- `auth_oauth`: Authorize via OAuth 2.0

See Also

`aw_auth_with()`

aw_auth_with *Set authorization options*

Description

Get or **set** various authorization options. If called without an argument, then these functions return the current setting for the requested option (which can be NULL if the option has not been set). To clear the setting, pass NULL as an argument.

aw_auth_with sets the type of authorization for the session. This is used as the default by aw_auth() when no specific option is given.

aw_auth_path sets the file path for the cached authorization token. It should be a directory, rather than a filename. If this option is not set, the current working directory is used instead.

aw_auth_name sets the file name for the cached authorization token. If this option is not set, the default filename is aw_auth.rds

Usage

```
aw_auth_with(type)

aw_auth_path(path)

aw_auth_name(name)
```

Arguments

type	The authorization type: 'oauth' or 'jwt'
path	The location for the cached authorization token. It should be a directory, rather than a filename. If this option is not set, the current working directory is used instead. If the location does not exist, it will be created the first time a token is cached.
name	The filename, such as aw_auth.rds for the cached authorization token file. The file is stored as an RDS file, but there is no requirement for the .rds file extension. .rds is not appended automatically.

Value

The option value, invisibly

See Also

aw_auth()

`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,
  check_components = TRUE
)
```

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 length-2 vector with a start date and an end date. `POSIXt` objects are sent as is, for fine control over the date range. Numeric values are automatically converted to dates.
`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 logical 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"). If `TRUE`, undoes any efficiency gains from setting `check_components` to `FALSE`.

`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`.

`check_components` Specifies whether to check the validity of metrics and dimensions before running the query. This defaults to `TRUE`, which triggers several additional API calls behind the scenes to retrieve all dimensions and metrics from the API. This has a nominal performance impact and may not be ideal if you are running many queries. If you have many queries, consider implementing validity checking through other means (manually or within the code) and then set this value to `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","mobiledevicetype")` 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","dateran`
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","lasttouchchannel")`:

- `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 the specified dimensions and metrics.

See Also

`get_me()`, `aw_get_reportsuites()`, `aw_get_segments()`, `aw_get_dimensions()`, `aw_get_metrics()`, `aw_get_calculatedmetrics()`, `aw_segment_table()`

`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 = NULL,
  ownerId = NULL,
```

```
  filterByIds = NULL,
  toBeUsedInRsid = NULL,
  locale = "en_US",
  name = NULL,
  tagNames = NULL,
  favorite = NULL,
  approved = NULL,
  limit = 1000,
  page = 0,
  sortDirection = "DESC",
  sortProperty = NULL,
  expansion = NULL,
  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 = NULL,
  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 = NULL,
  company_id = Sys.getenv("AW_COMPANY_ID"),
  debug = FALSE,
  use_oob
)
```

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 = NULL,
  rsidContains = NULL,
  limit = 10,
  page = 0,
  expansion = NULL,
  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 = NULL,
  segmentFilter = NULL,
  locale = "en_US",
  name = NULL,
  tagNames = NULL,
  filterByPublishedSegments = "all",
  limit = 10,
  page = 0,
  sortDirection = "ASC",
  sortProperty = "id",
```

```
  expansion = NULL,
  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_segment_table` *Get a segment-row freeform table*

Description

This is the equivalent of a freeform table with segments as the row components. This type of table offers a few components that `aw_freeform_table` does not. For example, this function does not require (or allow) dimensions to be included in the breakdown. Segment IDs are automatically translated into their human-readable names.

Usage

```
aw_segment_table(
  company_id = Sys.getenv("AW_COMPANY_ID"),
  rsid = Sys.getenv("AW_REPORTSUITE_ID"),
  segmentRsids = NULL,
  date_range = c(Sys.Date() - 30, Sys.Date() - 1),
  metrics = c("visits", "visitors"),
  globalSegment = NULL,
  segmentIds = NULL,
  debug = FALSE
)
```

Arguments

`company_id`	Company ID
`rsid`	Report suite ID for the data pull
`segmentRsids`	Vector of report suite IDs used to discover the human-readable segment names. Passed to `aw_get_segments`. If `NULL`, then takes the same value as `rsid`.
`date_range`	Date range
`metrics`	Metrics to request for each segment
`globalSegment`	One or more segments to apply globally over all other segments
`segmentIds`	One or more segments that will compose the rows of the table
`debug`	Logical, whether to make verbose requests to the API and view the whole exchange

Details

This is a specialized function. To see segments broken down by dimensions, we recommend making multiple requests to `aw_freeform_table` with different global segments applied, and then row-binding them together yourself.

Unlike `aw_freeform_table`, this function automatically handles the 10-metric restriction imposed by the API.

Efficiency:

In short, segments are cheap, metrics are expensive. Adding 1 metric is the equivalent of adding 10 segments, judging by the number of requests necessary to collect the data.

Stacking segments:

The function does not currently support segment breakdowns, but you can stack segments by applying a global segment to your query.

Value

`tibble::tibble()` of segments and metrics. Rows are returned with segments in the order they were requested, not by metric sorting.

See Also

`aw_freeform_table()`

`aw_token` *OAuth2 Token for Adobe Analytics (deprecated)*

Description

This is the legacy mechanism for retrieving the authorization token using OAuth. It has been replaced by `aw_auth()`.

[Deprecated]

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.

See Also

aw_auth()

---

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"))
```

Arguments

req_body	The json string copied from Workspace
company_id	Company Id. Taken from the global environment by default if not provided.

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")
```

Arguments

req_path	The endpoint for that particular report

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 = NULL,
  ip = NULL,
  rsid = NULL,
  eventType = NULL,
  event = NULL,
  limit = 100,
  page = 0,
  debug = FALSE,
  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.
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 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)
```

`seg_build` *Build the Segment in Adobe Analytics*

Description

This function combines rules, containers and/or sequences into a single JSON string and can then make the post call to create the segment in Adobe Analytics or return the json string for use in other api calls or for validation.

Usage

```
seg_build(
  name = NULL,
  description = NULL,
  containers = NULL,
  rules = NULL,
  sequences = NULL,
  context = "hits",
  conjunction = "and",
  sequence = "in_order",
  sequence_context = "hits",
  exclude = FALSE,
  create_seg = FALSE,
  debug = FALSE,
  rsid = Sys.getenv("AW_REPORTSUITE_ID"),
  company_id = Sys.getenv("AW_COMPANY_ID")
)
```

Arguments

`name`	This is the name of the new segment (required)
`description`	This is the description of the segment (required)
`containers`	List of the container(s) that make up the segment. Containers are list objects created using the `seg_con()` function.
`rules`	List of the rule(s) to create a segment. Rules are list objects created using the `seg_rule()` function.

`sequences` List of the rule(s) and sequence container(s) that are combined to make a segment. Sequence containers are list objects created using the `seg_seq()` function.

`context` Defines the level that the segment logic should operate on. Valid values are visitors, visits, and hits. See Details

`conjunction` This will tell how the different containers and rules should be compared. Use either 'and' or 'or'.

`sequence` Used to define if the segment should be 'in_order' (default), 'after', or 'before' the sequence of events

`sequence_context` Used to define the sequential items context which should be below the container context. ex. if container context is visitors then the sequence_context should be visits or hits

`exclude` Excludes the main container which will include all rules. Only used when the rule arguments are used.

`create_seg` Used to determine if the segment should be created in the report suite or if the definition should be returned to be used in a freeform table API call. Default is FALSE

`debug` This enables the api call information to show in the console for help with debugging issues. default is FALSE

`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.

`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.

Details

Context

The rules in a segment have a context that specify the level of operation. The context can be "visitors", "visits" or "hits." As an example, let's build a segment rule where revenue is greater than 0 (meaning a purchase took place) and change the context to see how things change. If the context is set to "visitors", the segment includes all hits from visitors that have a purchase of some kind during a visit. This is useful in analyzing customer behavior in visits leading up to a purchase and possibly behavior after a purchase. If the context is set to "visits", the segment includes all hits from visits where a purchase occurred. This is useful for seeing the behavior of a visitor in immediate page views leading up to the purchase. If the context is set to "hits", the segment only includes hits where a purchase occurred, and no other hits. This is useful in seeing which products were most popular. In the above example, the context for the container listed is hits. This means that the container only evaluates data at the hit level, (in contrast to visit or visitor level). The rows in the container are also at the hit level.

Value

If the "create_seg" argument is set to FALSE a JSON string definition will be returned. If the "create_seg" argument is set to TRUE and the segment is valid it will return a data frame of the newly created segment id along with some other basic meta data. If it returns an error then the error response will be returned to help understand what needs to be corrected.

`seg_con` *Create the segment container*

Description

This function combines rules into a container.

Usage

```
seg_con(context = "hits", conjunction = "and", rules = NULL, exclude = FALSE)
```

Arguments

`context`	Defines the level that the segment logic should operate on. Valid values are visitors, visits, and hits. See Details
`conjunction`	This defines the relationship of the rules. `And` (default) and `or` are the two options.
`rules`	List of rules and/or containers. Must be wrapped in a list() function. Adding a container list item will nest it within a container.
`exclude`	Exclude the entire container

Details

Context

The rules in a segment have a context that specify the level of operation. The context can be "visitors", "visits" or "hits." As an example, let's build a segment rule where revenue is greater than 0 (meaning a purchase took place) and change the context to see how things change. If the context is set to "visitors", the segment includes all hits from visitors that have a purchase of some kind during a visit. This is useful in analyzing customer behavior in visits leading up to a purchase and possibly behavior after a purchase. If the context is set to "visits", the segment includes all hits from visits where a purchase occurred. This is useful for seeing the behavior of a visitor in immediate page views leading up to the purchase. If the context is set to "hit", the segment only includes hits where a purchase occurred, and no other hits. This is useful in seeing which products were most popular. In the above example, the context for the container listed is hits. This means that the container only evaluates data at the hit level, (in contrast to visit or visitor level). The rules in the container are also at the hit level.

Value

a structured list of containers to be used to build the segment

seg_rule *Create the segment rule*

Description

This function creates the simple rule of a segment.

Usage

```
seg_rule(
  dimension = NULL,
  metric = NULL,
  verb = NULL,
  object = NULL,
  description = NULL,
  is_distinct = FALSE,
  attribution = "repeating",
  attribution_context = "visitors",
  validate = FALSE,
  rsid = Sys.getenv("AW_REPORTSUITE_ID"),
  company_id = Sys.getenv("AW_COMPANY_ID")
)
```

Arguments

dimension	This is the subject of the rule. The value should be the dimension id. Only the dimension or metric can be used at a time.
metric	This is the subject of the rule. The value should be the metric id. Only the dimension or metric can be used at a time.
verb	Choose from any of the 30 different verbs. Use the seg_verbs package data to see all available verbs along with the descriptions.
object	This is the object of the rule and answers the question what or how many
description	The internal description for the rule. (optional) This will not show in the UI but could be very helpful when using the API.
is_distinct	This will segment on a distinct count of items within a dimension. Examples: "Visitors who viewed more than 5 distinct products," or "Visits where more than 5 distinct pages were seen."
attribution	Define the type of attribution. Either repeating (default), instance, or nonrepeating. See Details for more information.
attribution_context	When applying a non-repeating instance attribution model to a rule the context for the attribution must be visitors (default) or visits
validate	Set to TRUE when metric or dimension validation is preferred. Default is FALSE. Validation will slow down the function response time but ensure a valid rule result.

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.
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.

Details

Attribution Models Available for dimensions only, these models determine what values in a dimension to segment for. Dimension models are particularly useful in sequential segmentation.

- *repeating* (default): Includes instances and persisted values for the dimension.
- *instance*: Includes instances for the dimension.
- *nonrepeating* instance: Includes unique instances (non-repeating) for the dimension. This is the model applied in Flow when repeat instances are excluded.

Value

A structured list defining the rule for a segment

---

seg_seq *Create the segment sequence container*

Description

This function combines rules into a sequence container.

Usage

```
seg_seq(
  context = "visits",
  rules = NULL,
  sequence = "in_order",
  exclude = FALSE,
  exclude_checkpoint = NULL
)
```

Arguments

context	Defines the level that the segment logic should operate on. Valid values for sequential segments is visitors and visits. See Details
rules	List of rules created using seg_rule() function. Must be wrapped in a list() function.
sequence	How should the sequence of items be considered. Options: in_order (default), before, after, and, or

`exclude` Excludes the entire sequence container which will include all rules.

`exclude_checkpoint`
Which checkpoints (rules) should be excluded. Example `c(1,4)`. See Details

Details

Context

The rules in a segment have a context that specify the level of operation. The context can be "visitors", "visits" or "hits." As an example, let's build a segment rule where revenue is greater than 0 (meaning a purchase took place) and change the context to see how things change. If the context is set to "visitors", the segment includes all hits from visitors that have a purchase of some kind during a visit. This is useful in analyzing customer behavior in visits leading up to a purchase and possibly behavior after a purchase. If the context is set to "visits", the segment includes all hits from visits where a purchase occurred. This is useful for seeing the behavior of a visitor in immediate page views leading up to the purchase. If the context is set to "hits", the segment only includes hits where a purchase occurred, and no other hits. This is useful in seeing which products were most popular. In the above example, the context for the container listed is hits. This means that the container only evaluates data at the hit level, (in contrast to visit or visitor level). The rows in the container are also at the hit level.

Exclude checkpoint

Ensures the next checkpoint doesn't happen between the preceding checkpoint and the subsequent checkpoint. If there is no subsequent checkpoint then the excluded checkpoint must not occur at any point after the preceding checkpoint. If there is no preceding checkpoint then the excluded checkpoint must not have occurred at any point preceding the subsequent checkpoint.

More Information

Sequential segments can be difficult to get right. Referencing this article can help: https://experienceleague.adobe.com/docs/a
workflow/seg-sequential-build.html?lang=en

Value

a structured list of containers to be used to build the segment

---

`seg_then` *Create the segment sequence then object*

Description

This function creates a then list object which restricts the time constraint of a segment to be added to a sequence segment.

Usage

```
seg_then(limit = "within", count = 1, unit = "year")
```

Arguments

limit	The limitation of the restriction. Either within (default) or after
count	How many of the units should be used. 1 is set as default.
unit	A unit of time. Valid values are hit, visit, minute, hour, day, week (default), month, quarter, or year. Always use the singular form.

Details

Combining seg_then **arguments:**

In the UI you can add 'after' and 'within' statements to create a more complex time restriction. The same can be accomplished using this function by listing the limits, counts, and units in a c() function. This would look like: limit = c('within', 'after'), count = c(5, 1), unit = c('hit', 'visit')

Using within and after in the same time seg_then **function call:**

Time restrictions can only be combined using 'within' first before 'after'. The function will automatically align these to be in the correct list item order.

A word about unit values:

Currently pageviews and dimensions are not supported unit values.

Value

a structured list of time restrictions to be used to build the sequential segment

---

seg_val *Validate a segment in adobe analytics*

Description

Returns a segment validation response for a segment contained in a json string object.

Usage

```
seg_val(
  segment_body = NULL,
  rsid = Sys.getenv("AW_REPORTSUITE_ID"),
  debug = FALSE,
  company_id = Sys.getenv("AW_COMPANY_ID")
)
```

Arguments

segment_body	The json string of the segment that is being validated (required)
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.
debug	This enables the api call information to show in the console for help with debugging issues. 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

If the segment is valid a message saying the segment validates is returned. If the segment doesn't validate the errors are returned in a data frame.

---

seg_verbs *Verbs available in segment rules.*

Description

A dataset containing the list of available verbs to be used in segment rules.

Usage

```
seg_verbs
```

Format

A data frame with 34 rows and 5 variables:

type one of number, string, or exists

class gives the context of the type of value is expected, either string, list, glob, number, or exists

verb the actual verb id to be used in the segment defition

description a simple description of the verb

arg specifies what argument to use when building the segment verb function ...

Source

https://developer.adobe.com/analytics-apis/docs/2.0/guides/endpoints/segments/definition/#available-data-comparison-functions

Index