

# Package: cjar (via r-universe)

August 23, 2024

**Type** Package

**Title** R Client for 'Customer Journey Analytics' ('CJA') API

**Version** 0.1.2

**Description** Connect to the 'CJA' API, which powers 'CJA Workspace' <<https://github.com/AdobeDocs/cja-apis>>. The package was developed with the analyst in mind and will continue to be developed with the guiding principles of iterative, repeatable, timely analysis. New features are actively being developed and we value your feedback and contribution to the process.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Depends** R (>= 3.2.0)

**RoxygenNote** 7.1.2

**Imports** assertthat, httr, magrittr, purrr, R6, dplyr, tidyr, jsonlite, glue, jose, tibble, lubridate, progress, vctrs, stringr, rlang, memoise, openssl

**NeedsCompilation** no

**Author** Ben Woodard [aut, cre], Charles Gallagher [ctb], Search Discovery, LLC [cph]

**Maintainer** Ben Woodard <[ben.woodard@searchdiscovery.com](mailto:ben.woodard@searchdiscovery.com)>

**Date/Publication** 2022-04-18 14:44:30 UTC

**Repository** <https://benwoodard.r-universe.dev>

**RemoteUrl** <https://github.com/cran/cjar>

**RemoteRef** HEAD

**RemoteSha** 660405925e190bf50f2ced995ac2a09d2673cb4e

## Contents

cjar-package . . . . .	2
annotation_build . . . . .	3
cja_auth . . . . .	4
cja_auth_with . . . . .	6
cja_freeform_table . . . . .	7
cja_get_annotations . . . . .	11
cja_get_audit_logs . . . . .	13
cja_get_audit_logs_search . . . . .	14
cja_get_calculatedmetrics . . . . .	15
cja_get_dataviews . . . . .	17
cja_get_dateranges . . . . .	19
cja_get_dimensions . . . . .	20
cja_get_filter . . . . .	21
cja_get_filters . . . . .	22
cja_get_me . . . . .	23
cja_get_metrics . . . . .	24
cja_get_projects . . . . .	25
cja_get_project_config . . . . .	26
filter_build . . . . .	27
filter_con . . . . .	29
filter_rule . . . . .	30
filter_seq . . . . .	32
filter_then . . . . .	33
filter_val . . . . .	34
filter_verbs . . . . .	34
<b>Index</b>	<b>35</b>

---

cjar-package

cjar *Package*


---

## Description

Connect to the 'CJA' API <https://www.adobe.io/cja-apis/docs> which powers 'CJA 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.

## Author(s)

**Maintainer:** Ben Woodard <ben.woodard@searchdiscovery.com>

Other contributors:

- Charles Gallagher <charlesjgallagher15@gmail.com> [contributor]
- Search Discovery, LLC [copyright holder]

---

annotation\_build      *Build the annotation in CJA*

---

## Description

This function builds an annotation in Customer Journey Analytics

## Usage

```
annotation_build(
  name = NULL,
  description = NULL,
  date_range = c(Sys.Date() - 30, Sys.Date() - 1),
  color = "blue",
  applyToAllReports = FALSE,
  metric_id = NULL,
  metric_compType = NULL,
  filter_id = NULL,
  filter_verb = NULL,
  filter_dimType = NULL,
  filter_terms = NULL,
  filter_compType = NULL,
  create_annotation = TRUE,
  debug = FALSE,
  dataviewId = Sys.getenv("CJA_DATAVIEW_ID")
)
```

## Arguments

name	This is the name of the new segment (required)
description	This is the description of the segment (required)
date_range	The date range of the annotation
color	Color name string representing the annotation's color. Supported values include 'blue', 'purple', 'green', 'orange', 'red', 'light green', 'pink', 'dark green', and 'yellow'.
applyToAllReports	A boolean that determines if the annotation applies to all report suites.
metric_id	The id for the metric scope as found in the <code>aw_get_metrics()</code> and <code>aw_get_calculatedmetrics()</code> functions. This needs to be a "list()" array of items.
metric_compType	The component type. This is either 'm' for metric or 'cm' for calculated metric. This needs to be a "list()" array of items.
filter_id	The id for the metric scope as found in the <code>aw_get_dimensions()</code> and <code>aw_get_segments()</code> functions. This needs to be a "list()" array of items.

filter_verb	The verb is the operator of the filter. The options include 'equals', 'equals_any', and 'exists'. If the 'dimension type' is 'enum' or 'ordered_enum' it can only use one of the verbs, 'equals' or 'exists'. This needs to be a "list()" array of items.
filter_dimType	The dimension type as defined by the 'type' column in <code>aw_get_dimensions()</code> if it is a dimension that is being used or it is blank because it is a segment. If it is a segment make sure to include an, "", empty list item. This needs to be a "list()" array of items
filter_terms	If the verb is "equals_any" then this argument should contain a list of values. If the verb is "equals" then the argument should be a single item. This needs to be a "list()" array of list items.
filter_compType	The component type is either a 'd' (dimension) or an 's' (segment). This needs to be a "list()" array of items.
create_annotation	Default is TRUE. Set this to FALSE if you want to get the json string that the function creates.
debug	This enables the api call information to show in the console for help with debugging issues. default is FALSE
dataviewId	<i>Required</i> The id of the dataview for which to retrieve dimensions. If an environment variable called CJA_DATAVIEW_ID exists in <code>.Renviro</code> n or elsewhere and no dataviewId argument is provided, then the CJA_DATAVIEW_ID value will be used. Use <code>cja_get_dataviews()</code> to get a list of available dataviewId.

**Value**

An id of the newly created annotation

---

cja\_auth

*Generate an access token for the Customer Journey Analytics API*

---

**Description**

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

**Usage**

```
cja_auth(type = "jwt", ...)

auth_jwt(
  file = Sys.getenv("CJA_AUTH_FILE"),
  private_key = Sys.getenv("CJA_PRIVATE_KEY"),
  jwt_token = NULL,
  ...
)
```

```

auth_oauth(
  client_id = Sys.getenv("CJA_CLIENT_ID"),
  client_secret = Sys.getenv("CJA_CLIENT_SECRET"),
  use_oob = TRUE
)

```

### Arguments

type	Either 'jwt' or 'oauth'. This can be set explicitly, but a best practice is to run <code>cja_auth_with()</code> to set the authorization type as an environment variable before running <code>cja_auth()</code>
...	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 <code>API_KEY</code> , <code>CLIENT_SECRET</code> , <code>ORG_ID</code> , and <code>TECHNICAL_ACCOUNT_ID</code> .
private_key	Filename of the private key for JWT authentication.
jwt_token	<i>(Optional)</i> A custom, encoded, signed JWT claim. If used, <code>client_id</code> and <code>client_secret</code> 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 <code>htr_oob_default</code> default, or TRUE if <code>httpuv</code> 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

[cja\\_auth\\_with\(\)](#)

---

`cja_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.

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

`cja_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.

`cja_auth_name` sets the file name for the cached authorization token. If this option is not set, the default filename is `cja_auth.rds`

### Usage

```
cja_auth_with(type)
```

```
cja_auth_path(path)
```

```
cja_auth_name(name)
```

### Arguments

<code>type</code>	The authorization type: 'oauth' or 'jwt'
<code>path</code>	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.
<code>name</code>	The filename, such as <code>cja_auth.rds</code> for the cached authorization token file. The file is stored as an RDS file, but there is no requirement for the <code>.rds</code> file extension. <code>.rds</code> is not appended automatically.

### Value

The option value, invisibly

### See Also

[cja\\_auth\(\)](#)

---

cja\_freeform\_table      *Get a freeform table*

---

## Description

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

## Usage

```
cja_freeform_table(
  dataviewId = Sys.getenv("CJA_DATAVIEW_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,
  allowRemoteLoad = "default",
  useCache = TRUE,
  useResultsCache = FALSE,
  includeOberonXml = FALSE,
  includePlatformPredictiveObjects = FALSE,
  debug = FALSE,
  check_components = FALSE
)
```

## Arguments

dataviewId	CJA Data View ID (dv). If an environment variable called CJA_DATAVIEW_ID exists in .Renviron or elsewhere and no dataviewId argument is provided, then the CJA_DATAVIEW_ID value will be used. Use <a href="#">cja_get_dataviews()</a> to get a list of available dataviewId. Required
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 CJA API works. Use <a href="#">cja_get_dimensions()</a> to get a list of available dimensions IDs.

metrics	A character vector of metrics. Use <code>cja_get_metrics()</code> and <code>cja_get_calculatedmetrics()</code> 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 <code>metricSort</code> ). 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 <b>Details</b> for information on the unique handling of <code>daterange...</code> values.
page	Used in combination with <code>top</code> to return the next page of results. Uses 0-based numbering (e.g., <code>top = 50000</code> and <code>page = 1</code> will return the top 50,000 items <i>starting at 50,001</i> ).
filterType	This is a placeholder argument for use as additional functionality is added to the package. Currently, it defaults to <code>breakdown</code> , 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 <code>segmentId</code> values are included, the segments will be effective ANDed together, just as if multiple segments were added to the header of an Analysis Workspace panel. Use <code>cja_get_filters()</code> to get a list of available <code>segmentId</code> values.
metricSort	Pre-sorts the table by metrics. Values are either <code>asc</code> (ascending) or <code>desc</code> (descending).
include_unspecified	Whether or not to include <b>Unspecified</b> values in the results. This is the equivalent of the <b>Include Unspecified (None)</b> checkbox in freeform tables in Analysis Workspace. This defaults to <code>TRUE</code> , which includes <b>Unspecified</b> values in the results.
search	Criteria to filter the results by one or more dimensions. Searches are case-insensitive. Refer to the <b>Details</b> for more information on constructing values for this argument.
pretty_names	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 <code>FALSE</code> , 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 <code>TRUE</code> , undoes any efficiency gains from setting <code>check_components</code> to <code>FALSE</code> .
allowRemoteLoad	Controls if Oberon should remote load data. Default behavior is <code>true</code> with fallback to <code>false</code> if remote data does not exist. The default is "default" but options include: "true", "false", or "default".
useCache	Use caching for faster requests (Use cached dimensions to speed up permission checks - This does not do any report caching). <code>TRUE</code> (default) or <code>FALSE</code>
useResultsCache	Use results caching for faster reporting times (This is a pass through to Oberon which manages the Cache) <code>FALSE</code> (default) or <code>TRUE</code>



includeOberonXml	Controls if Oberon XML should be returned in the response - DEBUG ONLY. FALSE (default) or TRUE
includePlatformPredictiveObjects	Controls if platform Predictive Objects should be returned in the response. Only available when using Anomaly Detection or Forecasting- DEBUG ONLY. FALSE (default) or TRUE
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	Logical, whether to check the validity of metrics and dimensions before running the query. Defaults to TRUE, but causes cja_freeform_report to request all dimensions and metrics from the API, which may be inefficient if you're running many queries. If you have many queries, it's more efficient to implement validity checking yourself on either side of your queries.

## 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:

CJA 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", "daterangeday")), 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 CJA 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 dimensions and metrics.

### See Also

[cja\\_get\\_me\(\)](#), [cja\\_get\\_dataviews\(\)](#), [cja\\_get\\_filters\(\)](#), [cja\\_get\\_dimensions\(\)](#), [cja\\_get\\_metrics\(\)](#)

Use [cja\\_get\\_me\(\)](#) to get started.

---

cja\_get\_annotations    *Get list of annotations*

---

## Description

Retrieve all annotations or filter to return only one

## Usage

```
cja_get_annotations(
  id = NULL,
  expansion = NULL,
  includeType = "all",
  locale = "en_US",
  filterByModifiedAfter = NULL,
  filterByDateRange = NULL,
  limit = 10,
  page = 0,
  debug = FALSE
)
```

## Arguments

id	Filter the results to one specific annotation by the annotation id. If not used, a list of annotations will be returned limited by the 'limit' and 'page' arguments.
expansion	Obtain additional information around an annotation. You can include multiple expansions using the 'c()' function. See details for options.
includeType	Include additional segments not owned by the user. Available values are all (default) and shared. The all option takes precedence over "shared".
locale	A query string that returns strings localized by Adobe into the desired language. Localization does not apply to user-defined fields, such as annotation names. See details for options.
filterByModifiedAfter	An ISO 8601 date that returns only annotations that were modified after the desired date. example datetime format: 'YYYY-MM-DDTHH:MM:SSZ'
filterByDateRange	Two ISO 8601 dates separated by a forward slash (/) that returns only annotations that fully reside within the desired date range. example format: 'MM:SSZ/YYYY-MM-DDTHH:MM:SSZ'
limit	An integer that represents the number of results per page. Default is 10
page	An integer that represents which page to return results. The first page is 0. The API supports up to 1000 pages
debug	Include the output and input of the api call in the console for debugging. Default is FALSE

## Details

- Expansion options include the following:
  - **name:** The name of the annotation.
  - **description:** The annotation's description.
  - **dateRange:** The date range of the annotation.
  - **color:** An enum representing the annotation's color. Supported values include STANDARD1 through STANDARD9. These correspond with 'blue', 'purple', 'green', 'orange', 'red', 'light green', 'pink', 'dark green', and 'yellow', in that order.
  - **applyToAllReports:** A boolean that determines if the annotation applies to all report suites.
  - **scope:** An object including the metrics and filters that the annotation uses.
  - **createdDate:** The date that the annotation was created.
  - **modifiedDate:** The date that the annotation was last modified.
  - **modifiedById:** The ID of the user who last modified the annotation.
  - **tags:** The tags applied to the annotation.
  - **shares:** The shares applied to the annotation.
  - **approved:** A boolean that determines if the annotation is approved by an admin.
  - **favorite:** A boolean that determines if the user has this annotation favorited (starred).
  - **usageSummary:** An object that shows where this annotation is used.
  - **owner:** An object showing the ID, name, and login of the user that created the annotation.
  - **imsOrgId:** The IMS org of the annotation.
  - **dataName:** The Data View name.
  - **dataId:** The Data View ID.
- Locale options include the following:
  - **en\_US:** English
  - **fr\_FR:** French
  - **ja\_JP:** Japanese
  - **de\_DE:** German
  - **es\_ES:** Spanish
  - **ko\_KR:** Korean
  - **pt\_BR:** Brazilian Portuguese
  - **zh\_CN:** Simplified Chinese
  - **zh\_TW:** Traditional Chinese

## Value

A data frame of segments and their meta data.

---

cja\_get\_audit\_logs      *Get audit logs*

---

## Description

This function will pull a list of audit logs defined by the different defined parameters.

## Usage

```
cja_get_audit_logs(  
    startDate = NULL,  
    endDate = NULL,  
    action = NULL,  
    component = NULL,  
    componentId = NULL,  
    userType = NULL,  
    userId = NULL,  
    userEmail = NULL,  
    description = NULL,  
    pageSize = 100,  
    pageNumber = 0,  
    debug = FALSE  
)
```

## Arguments

startDate	Date is not required, but if you filter by date, both start & end date must be set.
endDate	Date is not required, but if you filter by date, both start & end date must be set.
action	The action you want to filter by. See details section for options
component	The type of component you want to filter by. See details section for options
componentId	The ID of the component.
userType	The type of user.
userId	The ID of the user.
userEmail	The email address of the user.
description	The log description you want to filter by.
pageSize	Number of results per page. If left null, the default size will be set to 100.
pageNumber	Page number (base 0 - first page is "0")
debug	Used to help troubleshoot api call issues. Shows the call and result in the console

**Details**

*startDate/endDate* format

*Action* available values are: 'CREATE', 'EDIT', 'DELETE', 'LOGIN\_FAILED', 'LOGIN\_SUCCESSFUL', 'API\_REQUEST', 'LOGOUT', 'APPROVE', 'UNAPPROVE', 'SHARE', 'UNSHARE', 'TRANSFER', 'ORG\_CHANGE'

*Component* available values are: 'ANNOTATION', 'CALCULATED\_METRIC', 'CONNECTION', 'DATA\_GROUP', 'DATA\_VIEW', 'DATE\_RANGE', 'FILTER', 'MOBILE', 'PROJECT', 'REPORT', 'SCHEDULED\_PROJECT', 'USER', 'USER\_GROUP', 'IMS\_ORG', 'FEATURE\_ACCESS'

**Value**

A data frame of audit logs and corresponding metadata

**Examples**

```
## Not run:
cja_get_audit_logs()

## End(Not run)
```

---

```
cja_get_audit_logs_search
      Get audit logs search
```

---

**Description**

This function will pull a list of audit logs.

**Usage**

```
cja_get_audit_logs_search(body = NULL, debug = FALSE)
```

**Arguments**

body	The json string with the search functions included
debug	Set to TRUE if needed to help troubleshoot api call errors

**Value**

A data frame of audit logs and corresponding metadata

**Examples**

```
## Not run:
cja_get_audit_logs_search(body = jsonrequest)

## End(Not run)
```

---

 cja\_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, owner, polarity, precision, type. Other attributes can be optionally requested through the expansion field.

### Usage

```

cja_get_calculatedmetrics(
  expansion = NULL,
  includeType = "all",
  dataviewIds = NULL,
  ownerId = NULL,
  filterByIds = NULL,
  toBeUsedInRsid = NULL,
  locale = "en_US",
  favorite = NULL,
  approved = NULL,
  pagination = TRUE,
  limit = 10,
  page = 0,
  sortDirection = "DESC",
  sortProperty = NULL,
  debug = FALSE
)

```

### Arguments

expansion	Additional calculated metric metadata fields to include in the results: "dataName" "approved" "favorite" "shares" "tags" "sharesFullName" "usageSummary" "usageSummaryWithRelevancyScore" "reportSuiteName" "siteTitle" "ownerFullName" "modified" "migratedIds" "isDeleted" "definition" "authorization" "compatibility" "legacyId" "internal" "dataGroup" "categories".
includeType	Include additional calculated metrics not owned by user. Available values are all (default), shared, templates, unauthorized, deleted, internal, and curatedItem. The all option takes precedence over shared
dataviewIds	Filter the list to only include calculated metrics tied to a specified dataviewId or list of dataviewIds. Specify multiple dataviewIds as a vector (i.e., "dataviewIds = c("dataviewid_1", ...). Use <a href="#">cja_get_dataviews</a> to get a list of available dataviewId 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 data view where the calculated metric intended to be used. This data view will be 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 the compatibility expansion is specified and toBeUsedInRsid is not then the compatibility returned is based off 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.
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.
pagination	return paginated results. Set to 'TRUE' by default
limit	Number of results per page. 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 = 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.
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 cja\_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.

Other Expansion options that are available: "dataName", "approved", "favorite", "shares", "shares-FullName", "usageSummary", "usageSummaryWithRelevancyScore", "siteTitle", "migratedIds", "isDeleted", "authorization", "legacyId", "internal", "dataGroup", "categories"

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

[cja\\_get\\_metrics](#)

---

cja\_get\_dataviews      *Get data view ids*

---

### Description

This function will pull a list of data views ids that you have access to. These are similar to report suites in Adobe Analytics.

### Usage

```
cja_get_dataviews(  
  expansion = c("name"),  
  parentDataGroupId = NULL,  
  externalIds = NULL,  
  externalParentIds = NULL,  
  dataviewIds = NULL,  
  includeType = NULL,  
  cached = TRUE,  
  limit = 1000,  
  page = 0,  
  sortDirection = "ASC",  
  sortProperty = "id",  
  debug = FALSE  
)
```

**Arguments**

expansion	Comma-delimited list of additional fields to include on response. Options include: "name" "description" "owner" "isDeleted" "parentDataGroupId" "segmentList" "currentTimezoneOffset" "timezoneDesignator" "modified" "createdDate" "organization" "curationEnabled" "recentRecordedAccess" "sessionDefinition" "externalData" "containerNames"
parentDataGroupId	Filters data views by a single parentDataGroupId
externalIds	Comma-delimited list of external ids to limit the response with
externalParentIds	Comma-delimited list of external parent ids to limit the response with.
dataviewIds	Comma-delimited list of data view ids to limit the response with.
includeType	Include additional DataViews not owned by user. Options: "deleted"
cached	return cached results. TRUE (default) or FALSE
limit	number of results per page. 10 is default
page	Page number (base 0 - first page is 0). 0 is default
sortDirection	Sort direction ('ASC' (default) or DESC)
sortProperty	property to sort by (only modifiedDate and id are currently allowed). 'id' is default
debug	Used to help troubleshoot api call issues. Shows the call and result in the console

**Details**

**Expansion** available items: "name" "description" "owner" "isDeleted" "parentDataGroupId" "segmentList" "currentTimezoneOffset" "timezoneDesignator" "modified" "createdDate" "organization" "curationEnabled" "recentRecordedAccess" "sessionDefinition" "externalData" "containerNames"

**Value**

A data frame of dataview ids and their corresponding metadata

**Examples**

```
## Not run:
cja_get_dataviews()

## End(Not run)
```

---

`cja_get_dateranges`      *Get a paginated list of dateranges in CJA*

---

## Description

This function allows users to pull a list of stored date ranges so that they can be reused in an analysis.

## Usage

```
cja_get_dateranges(
  locale = "en_US",
  filterByIds = NULL,
  limit = 10,
  page = 0,
  expansion = "definition",
  includeType = "all",
  debug = FALSE
)
```

## Arguments

<code>locale</code>	Locale - Default: "en_US"
<code>filterByIds</code>	Filter list to only include date ranges in the specified list (comma-delimited list of IDs). This has filtered Ids from tags, approved, favorites and user specified Ids list.
<code>limit</code>	Number of results per page. default is 10
<code>page</code>	Page number (base 0 - first page is "0")
<code>expansion</code>	Comma-delimited list of additional date range metadata fields to include on response.
<code>includeType</code>	Include additional filters not owned by user. Default is "all". Options include: "all" (default), "shared", "templates"
<code>debug</code>	Used to help troubleshoot api call issues. Shows the call and result in the console

## Details

*expansion* options can include any of the following: "definition" "modified" "ownerFullName" "sharesFullName" "shares" "tags"

*includeType* options can include any of the following: "all", "shared", "templates"

## Value

A data frame of dateranges and their corresponding metadata

**Examples**

```
## Not run:
cja_get_dateranges()

## End(Not run)
```

---

`cja_get_dimensions`      *Get a list of dimensions in CJA*

---

**Description**

Retrieves a list of dimensions available in a specified `dataviewId`

**Usage**

```
cja_get_dimensions(
  dataviewId = Sys.getenv("CJA_DATAVIEW_ID"),
  expansion = "description",
  includeType = NULL,
  locale = "en_US",
  debug = FALSE
)
```

**Arguments**

<code>dataviewId</code>	<i>Required</i> The id of the dataview for which to retrieve dimensions. If an environment variable called <code>CJA_DATAVIEW_ID</code> exists in <code>.Renv</code> or elsewhere and no <code>dataviewId</code> argument is provided, then the <code>CJA_DATAVIEW_ID</code> value will be used. Use <a href="#">cja_get_dataviews()</a> to get a list of available <code>dataviewId</code> .
<code>expansion</code>	Comma-delimited list of additional segment metadata fields to include on response. See Details for all options available.
<code>includeType</code>	Include additional segments not owned by user. Options include: "shared" "templates" "deleted" "internal"
<code>locale</code>	Locale - Default: "en_US"
<code>debug</code>	Used to help troubleshoot api call issues. Shows the call and result in the console

**Details**

*Expansion* options can include the following: "approved" "favorite" "tags" "usageSummary" "usageSummaryWithRelevancyScore" "description" "sourceFieldId" "segmentable" "required" "hideFromReporting" "hidden" "includeExcludeSetting" "fieldDefinition" "bucketingSetting" "noValueOptionsSetting" "defaultDimensionSort" "persistenceSetting" "storageId" "tableName" "dataSetIds" "dataSetType" "type" "schemaPath" "hasData" "sourceFieldName" "schemaType" "sourceFieldType" "fromGlobalLookup" "multiValued" "precision"

**Value**

A data frame of dimensions in a specified dataview

**Examples**

```
## Not run:
cja_get_dimensions(dataviewId = "dv_5f4f1e2572ea0000003ce262")

## End(Not run)
```

---

cja_get_filter	<i>Get a filter in CJA</i>
----------------	----------------------------

---

**Description**

Retrieves a specific filter, also known as a segment in Adobe Analytics.

**Usage**

```
cja_get_filter(
  id = NULL,
  toBeUsedInRsid = NULL,
  locale = "en_US",
  expansion = "definition",
  debug = FALSE
)
```

**Arguments**

<code>id</code>	The filter id to retrieve
<code>toBeUsedInRsid</code>	The data view where the filter is intended to be used. This data view will be used to determine things like compatibility and permissions.
<code>locale</code>	Locale - Default: "en_US"
<code>expansion</code>	Comma-delimited list of additional filter metadata fields to include on response. See Details for all options available
<code>debug</code>	Used to help troubleshoot api call issues. Shows the call and result in the console

**Details**

*Expansion* options can include the following: "compatibility", "definition", "internal", "modified", "isDeleted", "definitionLastModified", "createdDate", "recentRecordedAccess", "performanceScore", "owner", "dataId", "ownerFullName", "dataName", "sharesFullName", "approved", "favorite", "shares", "tags", "usageSummary", "usageSummaryWithRelevancyScore"

**Value**

A filter list

**Examples**

```
## Not run:
cja_get_filter()

## End(Not run)
```

---

cja\_get\_filters

*Get a paginated list of filters in CJA*


---

**Description**

Retrieves a paginated list of filters, also known as segments in Adobe Analytics.

**Usage**

```
cja_get_filters(
  expansion = NULL,
  includeType = "all",
  dataviewIds = NULL,
  ownerId = NULL,
  filterByIds = NULL,
  toBeUsedInRsid = NULL,
  locale = "en_US",
  name = NULL,
  filterByModifiedAfter = NULL,
  cached = TRUE,
  pagination = TRUE,
  limit = 10,
  page = 0,
  sortDirection = "ASC",
  sortProperty = "id",
  debug = FALSE
)
```

**Arguments**

expansion	Comma-delimited list of additional segment metadata fields to include on response. See Details for all options available
includeType	Include additional filters not owned by user. Default is "all". Options include: "shared" "templates" "deleted" "internal"
dataviewIds	Filter list to only include filters tied to the specified data group ID list (comma-delimited)
ownerId	Filter list to only include filters owned by the specified imsUserId
filterByIds	Filter list to only include filters in the specified list (comma-delimited list of IDs). This has filtered IDs from tags, approved, favorites and user specified IDs list.

toBeUsedInRsid	The report suite where the segment is intended to be used. This report suite will be used to determine things like compatibility and permissions.
locale	Locale - Default: "en_US"
name	Filter list to only include filters that contains the Name. Can only be a string value.
filterByModifiedAfter	Filter list to only include filters modified since this date. 'yyyy-mm-dd' format
cached	Return cached results. TRUE by default.
pagination	Return paginated results
limit	Number of results per page
page	Page number (base 0 - first page is "0")
sortDirection	Sort direction ('ASC' or 'DESC'). 'ASC' is default.
sortProperty	Property to sort by (name, modified_date, performanceScore, id is currently allowed). 'id' is default
debug	Used to help troubleshoot api call issues. Shows the call and result in the console

### Details

*Expansion* options can include the following: "compatibility", "definition", "internal", "modified", "isDeleted", "definitionLastModified", "createdDate", "recentRecordedAccess", "performanceScore", "owner", "dataId", "ownerFullName", "dataName", "sharesFullName", "approved", "favorite", "shares", "tags", "usageSummary", "usageSummaryWithRelevancyScore"

### Value

A data frame of company ids and company names

### Examples

```
## Not run:
cja_get_filters()

## End(Not run)
```

---

cja\_get\_me

*Get my information*

---

### Description

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

### Usage

```
cja_get_me(expansion = NULL, debug = FALSE)
```

**Arguments**

expansion	Comma-delimited list of additional metadata fields to include in the response. Options are 'admin'
debug	Used to help troubleshoot api call issues. Shows the call and result in the console

**Value**

A list of the current user metadata

**Examples**

```
## Not run:
cja_get_me()

## End(Not run)
```

---

cja_get_metrics	<i>Get a list of metrics in CJA</i>
-----------------	-------------------------------------

---

**Description**

Retrieves a list of metrics available in a specified dataview

**Usage**

```
cja_get_metrics(
  dataviewId = Sys.getenv("CJA_DATAVIEW_ID"),
  expansion = "description",
  includeType = NULL,
  locale = "en_US",
  debug = FALSE
)
```

**Arguments**

dataviewId	<i>Required</i> The id of the dataview for which to retrieve metrics. If an environment variable called CJA_DATAVIEW_ID exists in .Renvirom or elsewhere and no dataviewId argument is provided, then the CJA_DATAVIEW_ID value will be used. Use <a href="#">cja_get_dataviews()</a> to get a list of available dataviewId.
expansion	Comma-delimited list of additional segment metadata fields to include on response. See Details for all options available.
includeType	Include additional segments not owned by user. Options include: "shared" "templates" "deleted" "internal"
locale	Locale - Default: "en_US"
debug	Used to help troubleshoot api call issues. Shows the call and result in the console



**Details**

*Expansion* options can include the following: "approved" "favorite" "tags" "usageSummary" "usageSummaryWithRelevancyScore" "description" "sourceFieldId" "segmentable" "required" "hideFromReporting" "hidden" "includeExcludeSetting" "fieldDefinition" "bucketingSetting" "noValueOptionsSetting" "defaultmetricsort" "persistenceSetting" "storageId" "tableName" "dataSetIds" "dataSetType" "type" "schemaPath" "hasData" "sourceFieldName" "schemaType" "sourceFieldType" "fromGlobalLookup" "multiValued" "precision"

**Value**

A data frame of metrics in a specified dataview

**Examples**

```
## Not run:
cja_get_metrics(dataviewId = "dv_5f4f1e2572ea0000003ce262")

## End(Not run)
```

---

cja_get_projects	<i>Get a paginated list of projects in CJA</i>
------------------	--

---

**Description**

Retrieves a paginated list of projects, also known as Workspace Projects.

**Usage**

```
cja_get_projects(
  includeType = "all",
  expansion = "definition",
  locale = "en_US",
  filterByIds = NULL,
  pagination = "true",
  ownerId = NULL,
  limit = 10,
  page = 0,
  debug = FALSE
)
```

**Arguments**

includeType	Include additional filters not owned by user. Default is "all". Options include: "all" (default) "shared"
expansion	Comma-delimited list of additional segment metadata fields to include on response. See Details for all options available
locale	Locale - Default: "en_US"

filterByIds	Filter list to only include filters in the specified list (comma-delimited list of IDs). This has filtered Ids from tags, approved, favorites and user specified Ids list.
pagination	Return paginated results
ownerId	Filter list to only include filters owned by the specified imsUserId
limit	Number of results per page
page	Page number (base 0 - first page is "0")
debug	Used to help troubleshoot api call issues. Shows the call and result in the console

### Details

*expansion* options can include any of the following: "shares" "tags" "accessLevel" "modified" "externalReferences" "definition"

*includeType* options can include any of the following: "all", "shared"

### Value

A data frame of projects and corresponding metadata

### Examples

```
## Not run:
cja_get_projects()

## End(Not run)
```

---

cja\_get\_project\_config

*Get a project configuration in CJA*

---

### Description

Retrieves a project configuration JSON string.

### Usage

```
cja_get_project_config(
  id = NULL,
  expansion = "definition",
  locale = "en_US",
  debug = FALSE
)
```

**Arguments**

id	(Required) The Project id for which to retrieve information
expansion	Comma-delimited list of additional segment metadata fields to include on response. See Details for all options available
locale	Locale - Default: "en_US"
debug	Used to help troubleshoot api call issues. Shows the call and result in the console

**Details**

*expansion* options can include any of the following: "shares" "tags" "accessLevel" "modified" "externalReferences" "definition"

**Value**

A project configuration list

**Examples**

```
## Not run:
cja_get_project_config(id = '6047e0a3de6aaaaac7c3accb')

## End(Not run)
```

---

filter_build	<i>Build the filter in CJA</i>
--------------	--------------------------------

---

**Description**

This function combines rules and/or containers and then makes the post call to create the filter in CJA.

**Usage**

```
filter_build(
  dataviewId = Sys.getenv("CJA_DATAVIEW_ID"),
  name = NULL,
  description = NULL,
  containers = NULL,
  rules = NULL,
  sequences = NULL,
  context = "hits",
  conjunction = "and",
  sequence = "in_order",
  sequence_context = "hits",
  exclude = FALSE,
  create_filter = FALSE,
```

```

    debug = FALSE,
    locale = "en_US",
    expansion = NULL
)

```

### Arguments

dataviewId	CJA data view id. If an environment variable called CJA_DATAVIEW_ID exists in .Renviron or elsewhere and no dataviewId argument is provided, then the CJA_DATAVIEW_ID value will be used. Use <code>cja_get_dataviews()</code> to get a list of available dataviewId. Required
name	This is the name of the new filter (required)
description	This is the description of the filter (required)
containers	List of the container(s) that make up the filter. Containers are list objects created using the <code>filter_con()</code> function.
rules	List of the rules to create a filter. Rules are list objects created using the <code>filter_rule()</code> function.
sequences	List of the predicate(s) and sequence container(s) that are combined to make a filter. Sequence containers are list objects created using the <code>filter_seq()</code> function.
context	Defines the level that the filter 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 filter 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_filter	Used to determine if the filter should be created in the UI or if the definition should be returned to be used in a freeform table API call as a global filter. Default is FALSE, which means the segment json string will be returned and the segment will not be created in the UI.
debug	This enables the api call information to show in the console for help with debugging issues. default is FALSE
locale	Locale. Default "en_US"
expansion	Comma-delimited list of additional filter metadata fields to include on response. See Detail section for available options

## Details

**Context** The rules in a filter have a context that specify the level of operation. The context can be visitors, visits or hits. As an example, let's build a filter 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 filter 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. the context is set to visits, the filter 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 filter 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.

**Expansion** Available option include the following: "compatibility" "definition" "internal" "modified" "isDeleted" "definitionLastModified" "createdDate" "recentRecordedAccess" "performanceScore" "owner" "dataId" "ownerFullName" "dataName" "sharesFullName" "approved" "favorite" "shares" "tags" "usageSummary" "usageSummaryWithRelevancyScore"

## Value

If the filter validates it will return a data frame of the newly created filter id along with some other basic meta data. If it returns and error then the error response will be returned to help understand what needs to be corrected. If the argument `create_filter` is set to `FALSE`, the json string will be returned in list format.

---

filter_con	<i>Create the filter container</i>
------------	------------------------------------

---

## Description

This function combines rules into a container

## Usage

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

## Arguments

context	Defines the level that the filter logic should operate on. Valid values are visitors, visits, and hits. See Details
conjunction	This defines the relationship of the rules. The two options are "and" (default) and "or".

rules	List of rules and/or containers. Must be wrapped in a list() function. Adding a container list item will nest it within a containers.
exclude	Exclude the entire container

### Details

**Context** The rules in a filter have a context that specify the level of operation. The context can be visitors, visits or hits. As an example, let's build a filter 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 filter 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 filter 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 filter 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

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

---

filter_rule	<i>Create the filter rule</i>
-------------	-------------------------------

---

### Description

This function creates the simple rule of a filter

### Usage

```
filter_rule(
  dimension = NULL,
  metric = NULL,
  verb = NULL,
  object = NULL,
  description = NULL,
  is_distinct = FALSE,
  attribution = "repeating",
  attribution_context = "visitors",
  validate = FALSE,
  dataviewId = Sys.getenv("CJA_DATAVIEW_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 <code>filter_verbs()</code> 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 filter 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.
dataviewId	CJA data view id. Required if the argument validate is set to TRUE. If an environment variable called CJA_DATAVIEW_ID exists in .Renviron or elsewhere and no dataviewId argument is provided, then the CJA_DATAVIEW_ID value will be used. Use <code>cja_get_dataviews()</code> to get a list of available dataviewId.

**Details**

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

- *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 filter

---

 filter\_seq

 Create the filter sequence container
 

---

### Description

This function combines rules into a sequence container

### Usage

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

### Arguments

context	Defines the level that the filter logic should operate on. Valid values for sequential filters is visitors and visits. See Details
rules	List of rules created using <code>filter_rule()</code> function. Must wrapped in a list() function.
sequence	How should the sequence of items be considered. Options: <code>in_order</code> (default), <code>before</code> , <code>after</code> , <code>and</code> , or
exclude	Excludes the entire sequence container which will include all rules.
exclude_checkpoint	Which checkpoints (rules) should be excluded. Example <code>c(1, 4)</code> . See Details

### Details

#### Context

The rules in a filter have a context that specify the level of operation. The context can be visitors, visits or hits. As an example, let's build a filter 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 filter 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. the context is set to visits, the filter 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 filter 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 filters can be difficult to get right. Referencing this article can help: <https://experienceleague.adobe.com/docs/analytics-platform/using/cja-components/cja-filters/filters-overview.html?lang=en>

### Value

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

---

filter_then	<i>Create the filter sequence then object</i>
-------------	---

---

### Description

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

### Usage

```
filter_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 filter\_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 filter\_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 filter

---

filter_val	<i>Validate a filter in CJA</i>
------------	---------------------------------

---

**Description**

Returns a filter validation for a filter contained in a json string object.

**Usage**

```
filter_val(filter_body = NULL, debug = FALSE)
```

**Arguments**

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

**Value**

A validation True or False response

---

filter_verbs	<i>Verbs available to be used in filter rules.</i>
--------------	--

---

**Description**

A dataset containing the list of available verbs which can be used in filters.

**Usage**

```
filter_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://experienceleague.adobe.com/docs/analytics-platform/using/cja-components/cja-filters/operators.html?lang=en>

# Index

- \* **auth**
  - cja\_auth, 4
- \* **datasets**
  - filter\_verbs, 34
- \* **options**
  - cja\_auth\_with, 6

annotation\_build, 3  
auth\_jwt (cja\_auth), 4  
auth\_oauth (cja\_auth), 4

cja\_auth, 4  
cja\_auth(), 6  
cja\_auth\_name (cja\_auth\_with), 6  
cja\_auth\_path (cja\_auth\_with), 6  
cja\_auth\_with, 6  
cja\_auth\_with(), 5  
cja\_freeform\_table, 7  
cja\_get\_annotations, 11  
cja\_get\_audit\_logs, 13  
cja\_get\_audit\_logs\_search, 14  
cja\_get\_calculatedmetrics, 15  
cja\_get\_calculatedmetrics(), 8  
cja\_get\_dataviews, 15, 17  
cja\_get\_dataviews(), 4, 7, 10, 20, 24, 28, 31  
cja\_get\_dateranges, 19  
cja\_get\_dimensions, 20  
cja\_get\_dimensions(), 7, 10  
cja\_get\_filter, 21  
cja\_get\_filters, 22  
cja\_get\_filters(), 8, 10  
cja\_get\_me, 23  
cja\_get\_me(), 10  
cja\_get\_metrics, 17, 24  
cja\_get\_metrics(), 8, 10  
cja\_get\_project\_config, 26  
cja\_get\_projects, 25  
cjar (cjar-package), 2  
cjar-package, 2  
filter\_build, 27  
filter\_con, 29  
filter\_con(), 28  
filter\_rule, 30  
filter\_seq, 32  
filter\_then, 33  
filter\_val, 34  
filter\_verbs, 34  
filter\_verbs(), 31