Package ‘RSiteCatalyst’

November 5, 2019

Type Package
Title R Client for Adobe Analytics API V1.4
Version 1.4.16
Date 2019-11-04
Author Willem Paling, Randy Zwitch & Jowanza Joseph
Maintainer Randy Zwitch <rzwitch+rsitecatalyst@gmail.com>
Depends R (>= 3.5)
Imports jsonlite (>= 1.0), httr (>= 1.0), plyr (>= 1.8), base64enc (>= 0.1), digest (>= 0.6), stringr (>= 1.0)
Suggests testthat (>= 2.0)
Description Functions for interacting with the Adobe Analytics API V1.4 (<https://api.omniture.com/admin/1.4/rest/>).
License MIT + file LICENSE
BugReports https://github.com/randyzwitch/RSiteCatalyst
RoxygenNote 6.0.1
NeedsCompilation no
Repository CRAN
Date/Publication 2019-11-05 05:20:02 UTC

R topics documented:

BuildClassificationValueSegment ............................................ 3
BuildRealTimeReportStructure .................................................. 4
CancelReport ........................................................................ 5
GetActivation ......................................................................... 6
GetAxleStartDate .................................................................... 7
GetBaseCurrency ...................................................................... 7
GetBaseURL ............................................................................. 8
GetBookmarks ........................................................................ 9
GetCalculatedMetrics ................................................................. 9
GetClassifications ................................................. 10
GetClickMapReporting ........................................ 11
GetCustomCalendar .......................................... 12
GetDashboards ................................................. 12
GetDataWarehouseDisplay .................................. 13
GetDefaultPage .............................................. 14
GetDiscoverEnabled ......................................... 15
GetEcommerce ................................................. 15
GetElements .................................................. 16
GetEvars ...................................................... 17
GetFeed ...................................................... 18
GetFeeds ..................................................... 18
GetFunctions ............................................... 19
GetGeoSegmentation ....................................... 20
GetGroup ..................................................... 21
GetGroups ................................................... 21
GetInternalURLFilters ..................................... 22
GetIPAddressExclusions .................................. 23
GetIPObfuscation ........................................... 24
GetKeyVisitors .............................................. 24
GetListVariables .......................................... 25
GetLocalization ............................................. 26
GetLogin ..................................................... 26
GetLogins ................................................... 27
GetMarketingChannelExpiration ....................... 28
GetMarketingChannelRules ................................ 28
GetMarketingChannels .................................... 29
GetMetrics ............................................... 30
GetMobileAppReporting ................................... 31
GetPaidSearchDetection .................................. 32
GetPermanentTraffic ....................................... 32
GetPreviousServerCalls .................................. 33
GetPrivacySettings .......................................... 34
GetProps .................................................... 35
GetQueue .................................................... 35
GetRealTimeReport .......................................... 36
GetRealTimeSettings ...................................... 37
GetReport ................................................... 38
GetReportDescription .................................... 39
GetReportsByIds ........................................... 40
GetReportSuiteGroups ...................................... 41
GetReportSuites ............................................ 41
GetScheduledSpike ........................................ 42
GetSegments ............................................... 43
GetSiteTitle ................................................ 43
GetSuccessEvents ....................................... 44
GetTemplate ............................................... 45
GetTimeStampEnabled .................................... 46
BuildClassificationValueSegment

**Build a Classification Value Segment**

**Description**

Function to build a classification value segment for use in segmenting reports.

**Usage**

```
BuildClassificationValueSegment(element, search.keywords, classification = "", search.type = "OR")
```

**Arguments**

- `element`: List of elements on which to base the segment
- `search.keywords`: List of search keyword vectors for each element (Use ^ to pin to start and $ to pin to end, or both to specify exact match)
- `classification`: (optional) Classification breakdown name for the element (defaults to the element name)
- `search.type`: How to combine the keywords list. This defaults to 'OR' if it is not specified.
BuildRealTimeReportStructure

Details

Function to build a classification value segment for use in segmenting reports.

Multiple segments can be combined in a list. Note that search can only be applied to a breakdown classification and not an element value.

Value

Segment definition for use with Queue* functions

Examples

```r
## Not run:
visitor_segment <- BuildClassificationValueSegment(element,
    search.keywords,
    classification,
    search.type)

## End(Not run)
```

BuildRealTimeReportStructure

Build Configuration for Real-Time Report

Description

Selects the metrics and elements (dimensions) on which you want Real-Time reports enabled. Use the returned list from this function as argument(s) in SaveRealTimeSettings.

Usage

```r
BuildRealTimeReportStructure(report.name = "", metric = "",
    elements = c(), min.granularity = "1", ui.report = TRUE)
```

Arguments

- `report.name`: Real-Time Report Name
- `metric`: Metric for Real-Time Report
- `elements`: Breakdowns for Real-Time Report
- `min.granularity`: Minimum Granularity for Report. Defaults to 1 minute.
- `ui.report`: Show report in Adobe Analytics web interface

Value

List
CancelReport

See Also

GetRealTimeSettings

SaveRealTimeSettings

Examples

## Not run:

```r
cancel_report <- function(id) {
  # ... implementation ...
}
```

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

---

**CancelReport**

**Cancel a Report in the Report Queue**

**Description**

Cancels a report in the Report Queue

**Usage**

`CancelReport(report.id)`

**Arguments**

- `report.id` Id of the report that you want to cancel

**Details**

Returns either a console message that no reports are queued or the reportID number that was cancelled
GetActivation

Value

Console message

Examples

## Not run:
CancelReport('12345678')

## End(Not run)

---

GetActivation  Get Activation Detail for a Report Suite(s)

Description

Get activation detail for the specified report suites.

Usage

GetActivation(reportsuite.ids)

Arguments

reportsuite.ids

Report suite id (or list of report suite ids)

Details

This function requires having a character vector with one or more valid Report Suites specified.

Value

Data frame

Examples

## Not run:
active <- GetActivation("your_report_suite")

active2 <- GetActivation(reportSuites$rsid)

## End(Not run)
**GetAxleStartDate**

**Get Cutover Date from SC14 to SC15 for a Report Suite(s)**

**Description**

Get cutover date from SC14 to SC15 for the specified report suites.

**Usage**

GetAxleStartDate(reportsuite.ids)

**Arguments**

- reportsuite.ids
  
  Report suite id (or list of report suite ids)

**Details**

This function requires having a character vector with one or more valid Report Suites specified.

**Value**

Data frame

**Examples**

```r
## Not run:
switch <- GetAxleStartDate("your_report_suite")

switch2 <- GetAxleStartDate(report_suites$rsid)
## End(Not run)
```

**GetBaseCurrency**

**Get Base Currency for a Report Suite(s)**

**Description**

Get base currency for the specified report suites.

**Usage**

GetBaseCurrency(reportsuite.ids)

**Arguments**

- reportsuite.ids
  
  Report suite id (or list of report suite ids)
GetBaseURL

Get Base URL for a Report Suite(s)

Description

Get base url for the specified report suites.

Usage

GetBaseURL(reportsuite.ids)

Arguments

reportsuite.ids

Report suite id (or list of report suite ids)

Details

This function requires having a character vector with one or more valid Report Suites specified.

Value

Data frame

Examples

## Not run:
url <- GetBaseURL("your_report_suite")

url2 <- GetBaseURL(report_suites$rsid)

## End(Not run)
GetBookmarks

Get Defined Bookmarks for a user

Description
Get defined bookmarks for a user.

Usage
GetBookmarks(folder.limit = "", folder.offset = "")

Arguments
folder.limit Max number of folders to return
folder.offset Offset of folders (i.e. start with other than first folder)

Details
This function’s arguments are both optional

Value
Data frame

Examples
## Not run:
bookmarks<- GetBookmarks()

bookmarks2 <- GetBookmarks('5', '1')

## End(Not run)

GetCalculatedMetrics

Get Calculated Metrics for a Report Suite(s)

Description
Get calculated metrics for the specified report suites.

Usage
GetCalculatedMetrics(reportsuite.ids)
GetClassifications

Get Classifications for Selected Report Suite Elements

Description

Retrieves a list of classifications (associated with the specified element) for each of the specified report suites.

Usage

GetClassifications(reportsuite.ids, elements = c())

Arguments

reportsuite.ids

Single report suite id or list of report suites

elements

Optional. List of existing elements you want to use in combination with an additional metric

Details

Retrieves a list of classifications (associated with the specified element) for each of the specified report suites. Function attempts to flatten classifications as best as possible; may return data frame having a nested list as a column if classification is sufficiently complex.

Value

Data frame
GetClickMapReporting

Examples

## Not run:

classifications <- GetClassifications(c("prod", "dev"), "trackingcode")

## End(Not run)

GetClickMapReporting  Get Click Map Settings for a Report Suite(s)

Description

Get Click Map Settings for the specified report suites.

Usage

GetClickMapReporting(reportsuite.ids)

Arguments

reportsuite.ids

Report suite id (or list of report suite ids)

Details

This function requires having a character vector with one or more valid Report Suites specified.

Value

Data frame

Examples

## Not run:
cmsettings <- GetClickMapReporting("your_report_suite")

cmsettings2 <- GetClickMapReporting(report_suites$rsid)

## End(Not run)
### GetCustomCalendar  
*Get Custom Calendar for a Report Suite(s)*

**Description**

Get custom calendar for the specified report suites.

**Usage**

```r
GetCustomCalendar(reportsuite.ids)
```

**Arguments**

- `reportsuite.ids`  
  Report suite id (or list of report suite ids)

**Details**

This function requires having a character vector with one or more valid Report Suites specified.

**Value**

Data frame

**Examples**

```r
## Not run:
cal <- GetCustomCalendar("your_report_suite")
cal2 <- GetCustomCalendar(report_suites$rsid)
## End(Not run)
```

### GetDashboards  
*Get Defined Dashboards*

**Description**

Get defined dashboards

**Usage**

```r
GetDashboards(dashboard.limit = "", dashboard.offset = "")
```
GetDataWarehouseDisplay

Arguments

dashboard.limit  
  Limit number of dashboards returned  
dashboard.offset  
  Offset of dashboards (i.e. start with other than first dashboard)

Details

  This function’s arguments are both optional

Value

  List

Examples

  ## Not run:
  dash<- GetDashboards()
  dash2 <- GetBookmarks(‘5’, ’1’)
  ## End(Not run)

GetDataWarehouseDisplay

  Get Whether Data Warehouse is Enabled for a Report Suite(s)

Description

  Get whether Data Warehouse is enabled for the specified report suites.

Usage

  GetDataWarehouseDisplay(reportsuite.ids)

Arguments

  reportsuite.id(s)
  Report suite id (or list of report suite ids)

Details

  This function requires having a character vector with one or more valid Report Suites specified.

Value

  Data frame
Examples

```r
## Not run:  
url <- GetDataWarehouseDisplay("your_report_suite")

url2 <- GetDataWarehouseDisplay(report_suites$rsid)

## End(Not run)
```

GetDefaultPage

Get Default Page for a Report Suite(s)

Description

Get default page for the specified report suites.

Usage

`GetDefaultPage(reportsuite.ids)`

Arguments

- `reportsuite.ids`: Report suite id (or list of report suite ids)

Details

This function requires having a character vector with one or more valid Report Suites specified.

Value

Data frame

Examples

```r
## Not run:  
defpage <- GetDefaultPage("your_report_suite")

defpage2 <- GetDefaultPage(report_suites$rsid)

## End(Not run)
```
GetDiscoverEnabled  

Get Whether Discover is Enabled for a Report Suite(s)

Description
Get whether Discover is enabled for the specified report suites.

Usage
GetDiscoverEnabled(reportsuite.ids)

Arguments
reportsuite.ids
Report suite id (or list of report suite ids)

Details
This function requires having a character vector with one or more valid Report Suites specified.

Value
Data frame

Examples
## Not run:
discenable <- GetDiscoverEnabled("your_report_suite")
discenable2 <- GetDiscoverEnabled(report_suites$rsid)
## End(Not run)

GetEcommerce  

Get the Commerce Level for a Report Suite(s)

Description
Get the commerce level for each of the specified report suites.

Usage
GetEcommerce(reportsuite.ids)

Arguments
reportsuite.ids
Report suite id (or list of report suite ids)
Details

This function requires having a character vector with one or more valid Report Suites specified.

Value

Data frame

Examples

```r
## Not run:
ecom <- GetEcommerce("your_report_suite")
ecom2 <- GetEcommerce(report_suites$rsid)
## End(Not run)
```

GetElements

Get Valid Elements for a Report Suite

Description

Get valid elements for a report suite for the current user. This list is restricted by optionally specified existing elements, existing metrics and date granularity.

Usage

```r
GetElements(reportsuite.ids, metrics = c(), elements = c(),
            date.granularity = "", report.type = "")
```

Arguments

- `reportsuite.ids` 
  Single report suite id, or character vector of report suite ids
- `metrics` 
  List of existing metrics you want to use in combination with an additional element
- `elements` 
  List of existing elements you want to use in combination with an additional element
- `date.granularity` 
  Granularity that you want to combine with an additional metric
- `report.type` 
  If set to 'warehouse', the elements and metrics returned to use in combination with an additional element are supported in data warehouse reports.

Details

This function requires a character vector with one or more valid Report Suites specified.
GetEvars

Description
Get Commerce Variables (eVars) Associated with a Report Suite.

Usage
GetEvars(reportsuite.ids)

Arguments
reportsuite.ids
report suite id (or list of report suite ids)

Details
This function requires having a character vector with one or more valid Report Suites specified.

Value
Data frame

Examples
## Not run:
evars <- GetEvars("your_report_suite")
evars2 <- GetEvars(report_suites$rsid)
## End(Not run)
GetFeed

Get Data Feed Detail for a specific feed

Description

Returns structure of a data feed, including column header names

Usage

GetFeed(feed.id)

Arguments

feed.id  Data Feed ID

Details

This function requires a single data feed id (obtained from GetFeeds)

Value

Data frame

Examples

## Not run:

# Get info for feed #110980
feed <- GetFeed("110980")

## End(Not run)

GetFeeds

Get Data Feed Detail for a Report Suite(s)

Description

Returns a list of data feeds for the specified report suites, including delivery status. Note that the difference between start.time and end.time can be no more than 48 hours.

Usage

GetFeeds(reportsuite.ids, start.time = "", end.time = "", status = c())
GetFunctions

Arguments

 reportsuite.ids 
Report suite id (or list of report suite ids)

 start.time 
Beginning of time period you want to check

 end.time 
End of time period you want to check

 status 
Character vector/list of statuses to filter by (processing/delivered/upload_error/no_data)

Details

This function requires having a character vector with one or more valid Report Suites specified. All other function arguments are optional.

Value

Data frame

Examples

## Not run:

# Get info for all feeds for a report suite in past day
feeds <- GetFeeds("zwitchdev")

# Get info for all feeds for a 48-hour period
feeds2 <- GetFeeds("zwitchdev", "2014-12-02 05:00:00", "2014-12-03 05:00:00")

## End(Not run)

---

GetFunctions | Get Functions Defined in Adobe Analytics

Description

Requests the id and definitions of functions in Adobe Analytics.

Usage

GetFunctions()

Details

Returns descriptions/formulas available within Adobe Analytics such as median, pi, regression, etc.

Value

Data Frame
GetGeoSegmentation

Get the Geography Segmentation for a Report Suite(s)

Description

Get the geography segmentation for the requested report suites.

Usage

GetGeoSegmentation(reportsuite.ids)

Arguments

reportsuite.ids

Report suite id (or list of report suite ids)

Details

This function requires having a character vector with one or more valid Report Suites specified.

Value

Data frame

Examples

## Not run:
geoseg <- GetGeoSegmentation("your_report_suite")

geoseg2 <- GetGeoSegmentation(report_suites$rsid)

## End(Not run)
GetGroup

Retrieves Information About The Specified Permission Group.

Description

Retrieves information about the specified permission group.

Usage

GetGroup(group_name)

Arguments

group_name  The name of the group that you want to retrieve.

Details

This function requires having a group name specified.

Value

Data frame

Examples

## Not run:
  group <- GetGroup("group_name")

## End(Not run)

GetGroups

Get Defined User Groups for a Company

Description

Get defined user groups for a company.

Usage

GetGroups(group_search_field = "", group_search_value = "")
GetInternalURLFilters

Get Internal URL Filters for a Report Suite(s)

Arguments

- group_search_field
  - Optional. Field to search for a specific value
- group_search_value
  - Optional. Use with group_search_field to search for a specific value

Details

This function requires having a character vector with one or more valid Report Suites specified.

Value

Data frame

Examples

```r
## Not run:
groups <- GetGroups()

## End(Not run)
```

GetInternalURLFilters

Get internal url filters for the specified report suites.

Usage

```r
GetInternalURLFilters(reportsuite.ids)
```

Arguments

- reportsuite.ids
  - Report suite id (or list of report suite ids)

Details

This function requires having a character vector with one or more valid Report Suites specified.

Value

Data frame
## Not run:
calc <- GetInternalURLFilters("your_report_suite")

calc2 <- GetInternalURLFilters(reportSuites$rsid)

## End(Not run)

---

**GetIPAddressExclusions**

*Get the IP Address Exclusions for a Report Suite(s)*

### Description

Get the IP address exclusions for the requested report suites.

### Usage

```r
GetIPAddressExclusions(reportsuite.ids)
```

### Arguments

- **reportsuite.ids**
  - Report suite id (or list of report suite ids)

### Details

This function requires having a character vector with one or more valid Report Suites specified.

### Value

Data frame

### Examples

```r
## Not run:
ipexc <- GetIPAddressExclusions("your_report_suite")

ipexc2 <- GetIPAddressExclusions(reportSuites$rsid)

## End(Not run)
```
GetIPObfuscation  
Get IP Obfuscation Status for a Report Suite(s)

**Description**
Get IP Obfuscation status for the requested report suites.

**Usage**
GetIPObfuscation(reportsuite.ids)

**Arguments**
- reportsuite.ids
  Report suite id (or list of report suite ids)

**Details**
This function requires having a character vector with one or more valid Report Suites specified.

**Value**
Data frame

**Examples**

```r
## Not run:
ipobf <- GetIPObfuscation("your_report_suite")
ipobf2 <- GetIPObfuscation(report_suites$rsid)
## End(Not run)
```

GetKeyVisitors  
Get Key Visitors for a Report Suite(s)

**Description**
Get key visitors for the specified report suites.

**Usage**
GetKeyVisitors(reportsuite.ids)

**Arguments**
- reportsuite.ids
  Report suite id (or list of report suite ids)
GetListVariables

Details
This function requires having a character vector with one or more valid Report Suites specified.

Value
Data frame

Examples

```r
## Not run:
keyvisit<- GetKeyVisitors("your_report_suite")

keyvisit2 <- GetKeyVisitors(report_suites$rsid)

## End(Not run)
```

---

**GetListVariables**  
*Get List Variables for a Report Suite(s)*

Description
Get list variables for the specified report suites.

Usage
GetListVariables(reportsuite.ids)

Arguments

- **reportsuite.ids**
  - Report suite id (or list of report suite ids)

Details
This function requires having a character vector with one or more valid Report Suites specified.

Value
Data frame

Examples

```r
## Not run:
calc<- GetListVariables("your_report_suite")

calc2 <- GetListVariables(report_suites$rsid)

## End(Not run)
```
GetLocalization

Get Localization for a Report Suite(s)

Description
Get localization for the specified report suites.

Usage
GetLocalization(reportsuite.ids)

Arguments
reportsuite.ids
Report suite id (or list of report suite ids)

Details
This function requires having a character vector with one or more valid Report Suites specified.

Value
Data frame

Examples
## Not run:
local <- GetLocalization("your_report_suite")
local2 <- GetLocalization(report_suites$rsid)
## End(Not run)

GetLogin

Get Login Information for a Single Login

Description
Get login information for a single login

Usage
GetLogin(login)

Arguments
login
Login to get information about
GetLogins

Details
This function requires a character string as the input

Value
List

Examples

## Not run:
login_info<- GetLogin('login-name')

## End(Not run)

---

GetLogins  Get Logins for a Company

Description
Get logins for a company

Usage
GetLogins(login.search.field = "", login.search.value = "")

Arguments
- login.search.field
  - Search login, first_name, last_name or title
- login.search.value
  - Value to search for (case-insensitive)

Details
This function’s arguments are both optional

Value
Data frame

Examples

## Not run:
logins<- GetLogins()

logins2 <- GetLogins('last_name', 'zwitch')

## End(Not run)
GetMarketingChannelExpiration

Get Marketing Channel Expiration for a Report Suite(s)

Description
Get marketing channel expiration for the specified report suites.

Usage
GetMarketingChannelExpiration(reportsuite.ids)

Arguments
reportsuite.ids
Report suite id (or list of report suite ids)

Details
This function requires having a character vector with one or more valid Report Suites specified.

Value
Data frame

Examples
## Not run:
expire <- GetMarketingChannelExpiration("your_report_suite")

expire2 <- GetMarketingChannelExpiration(report_suites$rsid)

## End(Not run)

GetMarketingChannelRules

Get Marketing Channel Rules for a Report Suite(s)

Description
Get marketing channel rules for the specified report suites.

Usage
GetMarketingChannelRules(reportsuite.ids)
GetMarketingChannels

Get Defined Marketing Channels for a Report Suite(s)

Description

Get defined marketing channels for each of the specified report suites.

Usage

GetMarketingChannels(reportsuite.ids)

Arguments

reportsuite.ids

Report suite id (or list of report suite ids)

Details

This function requires having a character vector with one or more valid Report Suites specified.

Value

Data frame

Examples

## Not run:
expire <- GetMarketingChannelRules("your_report_suite")

dmch <- GetMarketingChannelRules(report_suites$rsid)

## End(Not run)
### Not run:
```r
mchan <- GetMarketingChannels("your_report_suite")
```
```r
mchan2 <- GetMarketingChannels(report_suites$rsid)
```
### End(Not run)

---

## GetMetrics

### Get Available Metrics within a Report Suite

### Description

Gets valid metrics for current user, valid with optionally specified existing metrics, elements and date granularity

### Usage

```r
GetMetrics(reportsuite.ids, metrics = c(), elements = c(),
  date.granularity = "", report.type = "")
```

### Arguments

- **reportsuite.ids**: Single report suite id, or character vector of report suite ids
- **metrics**: List of existing metrics you want to use in combination with an additional metric
- **elements**: List of existing elements you want to use in combination with an additional metric
- **date.granularity**: Granularity that you want to combine with an additional metric
- **report.type**: If set to 'warehouse', the elements and metrics returned to use in combination with an additional element are supported in data warehouse reports.

### Details

This function requires having a character vector with one or more valid Report Suites specified.

### Value

Data frame
Examples

```r
## Not run:
metrics.valid <- GetMetrics("your_report_suite",
    metrics=c("visitors","pageviews"),
    elements=c("page","geoCountry"),
    date.granularity="day",
    report.type=""))

metrics <- GetMetrics(report_suites$rsid)

## End(Not run)
```

---

### GetMobileAppReporting

*Get Mobile App Reporting Status for a Report Suite(s)*

**Description**

Get mobile app reporting status for the specified report suites.

**Usage**

```r
GetMobileAppReporting(reportsuite.ids)
```

**Arguments**

- **reportsuite.ids**
  
  Report suite id (or list of report suite ids)

**Details**

This function requires having a character vector with one or more valid Report Suites specified.

**Value**

Data frame

**Examples**

```r
## Not run:
mobile <- GetMobileAppReporting("your_report_suite")

mobile2 <- GetMobileAppReporting(report_suites$rsid)

## End(Not run)
```
GetPaidSearchDetection

*Get Paid Search Detection Parameters for a Report Suite(s)*

**Description**

Get paid search detection parameters for the specified report suites.

**Usage**

GetPaidSearchDetection(reportsuite.ids)

**Arguments**

- **reportsuite.ids**
  
  Report suite id (or list of report suite ids)

**Details**

This function requires having a character vector with one or more valid Report Suites specified.

**Value**

Data frame

**Examples**

```r
## Not run:
paidsearch <- GetPaidSearchDetection("your_report_suite")
paidsearch2 <- GetPaidSearchDetection(report_suites$rsid)
## End(Not run)
```

GetPermanentTraffic

*Get Permanent Traffic Setting for a Report Suite(s)*

**Description**

Get permanent traffic setting for the specified report suites.

**Usage**

GetPermanentTraffic(reportsuite.ids)
GetPreviousServerCalls

Arguments

- reportsuite.ids
  Report suite id (or list of report suite ids)

Details

This function requires having a character vector with one or more valid Report Suites specified.

Value

Data frame

Examples

```r
## Not run:
permtraf <- GetPermanentTraffic("your_report_suite")
permtraf2 <- GetPermanentTraffic(report_suites$rsid)
## End(Not run)
```
Examples

## Not run:
gpsc <- GetPreviousServerCalls("your_report_suite")

gpsc2 <- GetPreviousServerCalls(report_suites$rsid)

## End(Not run)

---

**GetPrivacySettings**  
*Get Privacy Settings for a Report Suite(s)*

Description

Get privacy settings for the specified report suites.

Usage

GetPrivacySettings(reportsuite.ids)

Arguments

reportsuite.ids

Report suite id (or list of report suite ids)

Details

This function requires having a character vector with one or more valid Report Suites specified.

Value

Data frame

Examples

## Not run:
privacy <- GetPrivacySettings("your_report_suite")

privacy2 <- GetPrivacySettings(c("your_dev_suite", "your_prod_suite"))

## End(Not run)
GetProps

Get Traffic Variables (props) Associated with a Report Suite

Description

Get Traffic Variables (props) Associated with a Report Suite(s).

Usage

GetProps(reportsuite.ids)

Arguments

reportsuite.ids

report suite id (or list of report suite ids)

Details

This function requires having a character vector with one or more valid Report Suites specified.

Value

Data frame

Examples

## Not run:
props <- GetProps("your_report_suite")
props2 <- GetProps(reportSuites$rsid)
## End(Not run)

GetQueue

Get Number/ID of Reports in Queue

Description

Requests the number of reports in the Report Queue, as well as the Report ID.

Usage

GetQueue()

Details

Returns either a message to the console that no reports are in the Queue or a list with the Report ID’s.
GetRealTimeReport

Value

Console message and/or list

Examples

```r
## Not run:
queue <- GetQueue()

## End(Not run)
```

---

GetRealTimeReport  Get Real-Time report

Description

Function to access the Adobe Analytics Real-Time API v1.4. This API provides the ability for reporting up to the most recent minute. This API is best used at 15-30 second intervals (or longer).

Usage

```r
GetRealTimeReport(reportsuite.ids, metrics, elements = c(),
                   date.granularity = 5, date.from = "1 hour ago", date.to = "now",
                   sort.algorithm = "mostpopular", floor.sensitivity = 0.25,
                   first.rank.period = 0, algorithm.argument = "linear",
                   everything.else = TRUE, selected = c()))
```

Arguments

- `reportsuite.ids`: Report Suite
- `metrics`: Report metric
- `elements`: Report breakdowns
- `date.granularity`: Report Granularity. Defaults to 5 minutes
- `date.from`: Report starting time. Defaults to "1 hour ago"
- `date.to`: Report end time. Defaults to "now"
- `sort.algorithm`: Sorting algorithm. Defaults to "mostpopular"
- `floor.sensitivity`: Floor sensitivity. Defaults to .25
- `first.rank.period`: First Ranking Period. Defaults to 0
- `algorithm.argument`: Ranking algorithm. Defaults to "linear"
- `everything.else`: Provide counts for elements not returned as 'top'
- `selected`: Selected items for a given element (only works for a single element)
GetRealTimeSettings

Details
The Real-Time API uses a concept of "relative dates". To get a feeling for what's possible for submitting to date.from and date.to parameters, see link at:

Value
Data frame

Examples
## Not run:
custom_report <- GetRealTimeReport('')

## End(Not run)

---

GetRealTimeSettings Get Current Settings for Real-Time Reports

Description
Get Current Settings for Real-Time Reports

Usage
GetRealTimeSettings(reportsuite.ids)

Arguments
reportsuite.ids

    Report Suite ID

Details
GetRealTimeSettings returns a Data Frame with the current set up of real-time reports within the Adobe Analytics Real-Time API.

To change configuration settings, use SaveRealTimeConfiguration function.

Value
Data Frame

See Also

SaveRealTimeSettings
Examples

```r
## Not run:
GetRealTimeSettings("your_report_suite")

## End(Not run)
```

### Description

Get a single report by report id, this allow asynchronous way of getting reports.

### Usage

```r
GetReport(report.id, interval.seconds = 10, max.attempts = 3,
           print.attempts = TRUE, format = "json", page = 0)
```

### Arguments

- `report.id`: report id that’s returned by QueueTrended and other functions while used with enqueueOnly parameter set to TRUE
- `interval.seconds`: How long to wait between attempts
- `max.attempts`: Number of API attempts before stopping
- `print.attempts`: Print each attempt for fetching data
- `format`: "csv" or "json"
- `page`: Page Number of Results (QueueDataWarehouse only)

### Details

This is a function for advanced users, after you’ve enqueued multiple reports and want to get one of them when it’s ready.

### Value

Data frame
GetReportDescription  

Description

Get report description for a specific bookmark_id

Usage

GetReportDescription(bookmark.id)

Arguments

bookmark.id  Bookmark ID

Details

Requires a single bookmark_id value, obtained from GetBookmarks()

Value

List

Examples

## Not run:
reportdesc <- GetReportDescription("28473595")

## End(Not run)
GetReportsByIds  

Get EnQueued Reports by report ID

Description

Get reports for report ids provided as a list. These reports are previously enqueued.

Usage

GetReportsByIds(report.ids, interval.seconds = 10, max.attempts = 300, print.attempts = TRUE)

Arguments

- report.ids: list of report ids that you’ve enqueued and want to retrieve the data for
- interval.seconds: How long to wait between attempts
- max.attempts: Number of API attempts before stopping
- print.attempts: Print each attempt to check if report is ready

Details

This is a function for advanced users, after you’ve enqueued multiple reports and want to get all of them when they’re ready.

Value

list of (report id and Data frame pairs)

Examples

## Not run:

reports <- GetReportsByIds(list(12345678,87654321),print.attempts=FALSE)

## End(Not run)


**GetReportSuiteGroups**

*Get Report Suite Groups for a specific report suite*

**Description**

Retrieves a list of permission groups assigned to the specified report suite

**Usage**

```r
GetReportSuiteGroups(reportsuite.id)
```

**Arguments**

- `reportsuite.id` Report Suite ID

**Details**

Requires a single report suite id

**Value**

Data frame

**Examples**

```r
## Not run:
rsg <- GetReportSuiteGroups("your-report-suite")

## End(Not run)
```

---

**GetReportSuites**

*Get Report Suites Associated with a Specific User/Company*

**Description**

Get Report Suites Associated with a Specific User/Company

**Usage**

```r
GetReportSuites()
```

**Details**

Returns a data frame containing the Report Suite ID and Site Title
### GetScheduledSpike

**Get Scheduled Traffic Spike Setting for a Report Suite(s)**

#### Description
Get scheduled traffic spikes for the specified report suites.

#### Usage

```r
GetScheduledSpike(reportsuite.ids)
```

#### Arguments
- `reportsuite.ids`
  Report suite id (or list of report suite ids)

#### Details
This function requires having a character vector with one or more valid Report Suites specified.

#### Value
Data frame

#### Examples
```r
## Not run:
spike <- GetScheduledSpike("your_report_suite")
spike2 <- GetScheduledSpike(report_suites$rsid)
## End(Not run)
```
GetSegments

Get Segments Defined within a Report Suite

Description
Get a data frame of segments for the specified report suites. Useful to find segment IDs for use in reporting helper functions or JSON report definitions.

Usage
GetSegments(reportsuite.ids)

Arguments
reportsuite.ids
Report suite id (or list of report suite ids)

Details
This function requires having a character vector with one or more valid Report Suites specified.

Value
Data frame

Examples
## Not run:
segments <- GetSegments("your_report_suite")
segments2 <- GetSegments(report_suites$rsid)
## End(Not run)

GetSiteTitle

Get Site Title for a Report Suite(s)

Description
Get site title for the specified report suites.

Usage
GetSiteTitle(reportsuite.ids)
**GetSuccessEvents**

**Arguments**

- `reportsuite.ids`  
  Report suite id (or list of report suite ids)

**Details**

This function requires having a character vector with one or more valid Report Suites specified.

**Value**

Data frame

**Examples**

```r
## Not run:
sitetitle <- GetSiteTitle("your_report_suite")
sitetitle2 <- GetSiteTitle(report_suites$rsid)
## End(Not run)
```

---

**Description**

Gets success event definitions for the specified report suite(s). Useful to audit or document a report suite or company in Adobe Analytics.

**Usage**

```r
GetSuccessEvents(reportsuite.ids)
```

**Arguments**

- `reportsuite.ids`  
  report suite id (or list of report suite ids)

**Details**

This function requires having a character vector with one or more valid Report Suites specified.

**Value**

Data frame
### GetTemplate

**Get Template a Report Suite is Based On**

**Description**

Get template a report suite is based on for the specified report suites.

**Usage**

```r
GetTemplate(reportsuite.ids)
```

**Arguments**

- `reportsuite.ids`
  
  Report suite id (or list of report suite ids)

**Details**

This function requires having a character vector with one or more valid Report Suites specified.

**Value**

Data frame

**Examples**

```r
## Not run:
privacy <- GetPrivacySettings("your_report_suite")

privacy2 <- GetPrivacySettings(c("your_dev_suite", "your_prod_suite"))

## End(Not run)
```
GetTimeStampEnabled  
*Get Time Stamp Enabled for a Report Suite(s)*

**Description**
Get whether Time Stamp functionality enabled for the specified report suites.

**Usage**

```
GetTimeStampEnabled(reportsuite.ids)
```

**Arguments**

- `reportsuite.ids`
  
  Report suite id (or list of report suite ids)

**Details**
This function requires having a character vector with one or more valid Report Suites specified.

**Value**
Data frame

**Examples**
```r
## Not run:
tse <- GetTimeStampEnabled("your_report_suite")
tse2 <- GetTimeStampEnabled(report_suites$rsid)
## End(Not run)
```

GetTimeZone  
*Get Time Zone for a Report Suite(s)*

**Description**
Get Time Zone for the specified report suites.

**Usage**

```
GetTimeZone(reportsuite.ids)
```

**Arguments**

- `reportsuite.ids`
  
  Report suite id (or list of report suite ids)
GetTrackingServer

Details

This function requires having a character vector with one or more valid Report Suites specified.

Value

Data frame

Examples

```r
## Not run:
timezone <- GetTimeZone("your_report_suite")
timezone2 <- GetTimeZone(report_suites$rsid)
## End(Not run)
```

---

GetTrackingServer  Get Tracking Server Associated with a Namespace (Company)

Description

Get tracking server associated with a namespace (company).

Usage

```r
GetTrackingServer(reportsuite.id)
```

Arguments

- `reportsuite.id`: report suite id

Details

This function requires having a character string with a valid Report Suite specified. You can specify any report suite you want, as all report suites have same tracking server.

Value

Data frame

Examples

```r
## Not run:
ts <- GetTrackingServer("your_report_suite")
## End(Not run)
```
GetTransactionEnabled  *Get Whether Transaction Storage for a Report Suite(s)*

**Description**

Get whether transaction storage is enabled for the specified report suites.

**Usage**

GetTransactionEnabled(reportsuite.ids)

**Arguments**

- `reportsuite.ids`
  
  Report suite id (or list of report suite ids)

**Details**

This function requires having a character vector with one or more valid Report Suites specified.

**Value**

Data frame

**Examples**

```r
## Not run:
trans <- GetTransactionEnabled("your_report_suite")
trans2 <- GetTransactionEnabled(report_suites$rsid)
## End(Not run)
```

GetUniqueVisitorVariable  *Get Whether Unique Visitor Variable Enabled for a Report Suite(s)*

**Description**

Get whether unique visitor variable is enabled for the specified report suites.

**Usage**

GetUniqueVisitorVariable(reportsuite.ids)
GetVersionAccess

Arguments

  reportsuite.ids
    Report suite id (or list of report suite ids)

Details

  This function requires having a character vector with one or more valid Report Suites specified.

Value

  Data frame

Examples

  ## Not run:
  uniq <- GetUniqueVisitorVariable("your_report_suite")

  uniq2 <- GetUniqueVisitorVariable(report_suites$rsid)

  ## End(Not run)

GetVersionAccess

Get Products/Versions associated with a specific company

Description

  Get Products/Versions associated with a specific company

Usage

  GetVersionAccess()

Details

  Returns a Data Frame of Adobe Analytics products

Value

  Data Frame

Examples

  ## Not run:
  versions <- GetVersionAccess()

  ## End(Not run)
GetVideoSettings  
*Get Video Settings for a Report Suite(s)*

**Description**
Get video settings for the specified report suites.

**Usage**
GetVideoSettings(reportsuite.ids)

**Arguments**
- **reportsuite.ids**
  Report suite id (or list of report suite ids)

**Details**
This function requires having a character vector with one or more valid Report Suites specified.

**Value**
Data frame

**Examples**
```r
## Not run:
vidsettings<- GetVideoSettings("your_report_suite")
vidsettings2 <- GetVideoSettings(report_suites$rsid)
## End(Not run)
```

GetVirtualReportSuiteSettings  
*Get Virtual Report Suite Settings*

**Description**
Get Virtual Report Suite definitions.

**Usage**
GetVirtualReportSuiteSettings(reportsuite.ids)
QueueDataWarehouse

Arguments

  reportsuite.id
      report suite id (or list of report suite ids)

Details

  This function requires having a character vector with one or more valid Report Suites specified.

Value

  Data frame

Examples

  ## Not run:
  virtualsettings <- GetVirtualReportSuiteSettings("your_report_suite")

  virtualsettings2 <- GetVirtualReportSuiteSettings(report_suites$rsid)

  ## End(Not run)

QueueDataWarehouse  Queue a DataWarehouse Report

Description

  A QueueDataWarehouse report is a report where metrics are retrieved, broken down by an unlimited
  number of elements such as page, eVar, prop, etc, and with or without temporal aggregation.

Usage

  QueueDataWarehouse(reportsuite.id, date.from, date.to, metrics, elements,
                     date.granularity = "day", segment.id = "", data.current = TRUE,
                     expedite = FALSE, interval.seconds = 5, max.attempts = 120,
                     validate = TRUE, enqueueOnly = TRUE, ftp = "",
                     classification = c())

Arguments

  reportsuite.id  Report suite id
  date.from      Start date for the report (YYYY-MM-DD)
  date.to        End date for the report (YYYY-MM-DD)
  metrics        List of metrics to include in the report
  elements       List of elements to include in the report
QueueDataWarehouse

date.granularity
  Time granularity of the report (year/month/week/day/hour), default to 'day'.
  Pass NULL if you do not want any time granularity.
segment.id
  Id of Adobe Analytics segment to retrieve the report for
data.current
  TRUE or FALSE - whether to include current data for reports that include today's date
expedite
  Set to TRUE to expedite the processing of this report
interval.seconds
  How long to wait between attempts
max.attempts
  Number of API attempts before stopping
validate
  whether to submit report definition for validation before requesting the data.
enqueueOnly
  only enqueue the report, don’t get the data. returns report id, which you can later use to get the data
ftp
  FTP client parameters, only used if enqueueOnly=TRUE. Double check ftp parameters before requesting a long report.
classification
  List of SAINT classifications for each element. If supplied, must be a character vector of length equal to elements.

Details

The QueueDataWarehouse function allows to access to Data Warehouse data and returns either json or sends a csv to a ftp server.

Because of the Reporting API structure, this function requests the report, then, if enqueueOnly=FALSE, checks the reporting queue to see if the report is completed, and when the report returns as "done" pulls the report from the API (if ftp is not defined). This checking process will occur up to the specified number of times (default 120), with a delay between status checks (default 5 seconds). If the report does not return as "done" or a "delivery_complete" after the number of tries have completed, the function will return an error message. When enqueueOnly=TRUE and no ftp server is set, the report can be retrieved with Report.Get using the reportId returned by the QueueDataWarehouse function.

Data Warehouse documentation: https://docs.adobe.com/content/help/en/analytics/export/data-warehouse/data-warehouse.html
Data Warehouse (1.4) API documentation: https://github.com/AdobeDocs/analytics-1.4-apis/blob/master/docs/reporting-api/data_warehouse.md

Multiple segments are supported as long as they are compatible with Data Warehouse.

Classifications are supported, and if supplied, must be equal in length to elements. Pad out classifications with either a blank string or NA if and as necessary.

The following element properties are not supported in Data Warehouse reports: - selected - search - top - startingWith - sortBy

Calculated metrics are not supported.

Results for data warehouse reports can be accessed in two ways: directly through the API and through FTP delivery. Email delivery is not supported.

All data warehouse results are paged in chunks of 20 MB. Add "page": to Report.Get to determine the page returned. If no page is specified then the first page is returned.
QueueFallout

Value

Data frame or report id, if enqueueOnly is TRUE

Examples

```r
## Not run:
report.data <- QueueDataWarehouse("your_report_suite",
  "2014-01-01",
  "2014-01-07",
  c("visits", "pageviews","event10"),
  c("page","geoCountry","geoCity"),
enqueueOnly=TRUE,
ftp = list(host = "myftpserver.com",
  port = "21",
  directory = "/fromDW/",
  username = "memyselfandirene",
  password = "valkilmer",
  filename = "myreport.csv")
)
```

## End(Not run)

---

QueueFallout  Run a Fallout Report

Description

A QueueFallout Report is a report that shows how visitors drop out as part of a specified path.

Usage

```
QueueFallout(reportsuite.id, date.from, date.to, metrics, element, checkpoints,
  segment.id = "", expedite = FALSE, interval.seconds = 5,
  max.attempts = 120, validate = TRUE, enqueueOnly = FALSE)
```

Arguments

- **reportsuite.id**: Report suite id
- **date.from**: Start date for the report (YYYY-MM-DD)
- **date.to**: End date for the report (YYYY-MM-DD)
- **metrics**: List of metrics to include in the report
- **element**: Single pathed element (usually 'page')
- **checkpoints**: Character vector of checkpoints in the fallout path (e.g. c("Home","Contact","Thank You"))
- **segment.id**: Id(s) of Adobe Analytics segment to retrieve the report for
- **expedite**: Set to TRUE to expedite the processing of this report
QueueOvertime

### interval.seconds
How long to wait between attempts

### max.attempts
Number of API attempts before stopping

### validate
whether to submit report definition for validation before requesting the data.

### enqueueOnly
only enqueue the report, don’t get the data. returns report id, which you can later use to get the data.

**Details**

Because of the Reporting API structure, this function first requests the report, then checks the reporting queue to see if the report is completed, and when the report returns as "done" pulls the report from the API. This checking process will occur up to the specified number of times (default 120), with a delay between status checks (default 5 seconds). If the report does not return as "done" after the number of tries have completed, the function will return an error message.

**Value**

Data frame or report id, if enqueueOnly is TRUE

**Examples**

```r
## Not run:
falloutpattern <- c("Home Page","Contact Page","Login Page")
queue_fallout_pages <- QueueFallout("your_report_suite",
  "2014-04-01",
  "2014-04-20",
  metric="pageviews",
  element="page",
  falloutpattern
)
queued_report_id <- QueueFallout("your_report_suite",
  "2014-04-01",
  "2014-04-20",
  metric="pageviews",
  element="page",
  falloutpattern,
  enqueueOnly=TRUE
)
```

## End(Not run)
QueueOvertime

Description
A QueueOvertime report is a report where the only granularity allowed is time. This report allows for a single report suite, time granularity, multiple metrics, and a single segment. It is similar to the "Key Metrics" report or a Custom Event report within the Adobe Reports & Analytics interface. To get a summary report with no time granularity (i.e. a single row), pass an empty string to the date.granularity function parameter.

Usage
QueueOvertime(reportsuite.id, date.from, date.to, metrics,
   date.granularity = "day", segment.id = "", segment.inline = "",
   anomaly.detection = FALSE, data.current = FALSE, expedite = FALSE,
   interval.seconds = 5, max.attempts = 120, validate = TRUE,
   enqueueOnly = FALSE)

Arguments
reportsuite.id Report suite id
date.from Start date for the report (YYYY-MM-DD)
date.to End date for the report (YYYY-MM-DD)
metrics List of metrics to include in the report
date.granularity Time granularity of the report (year/month/week/day/hour"), default to 'day'
segment.id Id(s) of Adobe Analytics segment to retrieve the report for
segment.inline Inline segment definition
anomaly.detection Set to TRUE to include forecast data (only valid for day granularity with small date ranges)
data.current TRUE or FALSE - Whether to include current data for reports that include today's date
expedite Set to TRUE to expedite the processing of this report
interval.seconds How long to wait between attempts
max.attempts Number of API attempts before stopping
validate whether to submit report definition for validation before requesting the data.
enqueueOnly only enqueue the report, don't get the data. returns report id, which you can later use to get the data

Details
Because of the Reporting API structure, this function first requests the report, then checks the reporting queue to see if the report is completed, and when the report returns as "done" pulls the report from the API. This checking process will occur up to the specified number of times (default 120), with a delay between status checks (default 5 seconds). If the report does not return as "done" after the number of tries have completed, the function will return an error message.
QueuePathing

Value

Data frame

Examples

## Not run:

```r
overtime1 <- QueueOvertime("your_report_suite",
  date.from = "2014-04-01",
  date.to = "2014-04-20",
  metrics = c("pageviews", "visits", "bounces"),
  date.granularity = "day")

overtime2 <- QueueOvertime("your_report_suite",
  date.from = "2014-04-01",
  date.to = "2014-04-20",
  metrics = c("pageviews", "visits", "bounces"),
  date.granularity = "day",
  segment.id = "5433e4e4b02df70be4ac63",
  anomaly.detection = TRUE,
  interval.seconds = 10,
  max.attempts = 20)

overtime3 <- QueueOvertime("your_report_suite",
  date.from = "2014-04-01",
  date.to = "2014-04-20",
  metrics = c("pageviews", "visits", "bounces"),
  date.granularity = "")

enqueued.report.id <- QueueOvertime("your_report_suite",
  date.from = "2014-04-01",
  date.to = "2014-04-20",
  metrics = c("pageviews", "visits", "bounces"),
  date.granularity = "",
  enqueueOnly=TRUE)
```

## End(Not run)

---

QueuePathing  

Run a Pathing report

Description

A QueuePathing Report is a report that shows how often visitors go from Page A to Page B to Page C on site.
QueuePathing

Usage

QueuePathing(reportsuite.id, date.from, date.to, metric, element, pattern,
    top = 1000, start = 1, segment.id = "", expedite = FALSE,
    interval.seconds = 5, max.attempts = 120, validate = TRUE,
    enqueueOnly = FALSE)

Arguments

  reportsuite.id     Report suite id
  date.from          Start date for the report (YYYY-MM-DD)
  date.to            End date for the report (YYYY-MM-DD)
  metric             Single metric to include in the report (usually 'pageviews')
  element            Single pathed element (usually 'page')
  pattern            Character vector of items in the path (up to 3) use ":::anything::" as a wildcard.
                     For example c("Home","::anything::","::anything::") will return all paths that
                     start with the home page, c("::anything::","Home","::anything::") will return
                     the previous and next pages from the home page, and c("::anything::","::anything::","Home")
                     will return the two previous pages leading to the home page.
  top                Number of rows to return (defaults to 1000)
  start              Start row if you do not want to start at #1
  segment.id         Id(s) of Adobe Analytics segment to retrieve the report for
  expedite           Set to TRUE to expedite the processing of this report
  interval.seconds   How long to wait between attempts
  max.attempts       Number of API attempts before stopping
  validate           whether to submit report definition for validation before requesting the data.
  enqueueOnly        only enqueue the report, don’t get the data. returns report id, which you can later
                     use to get the data

Details

Because of the Reporting API structure, this function first requests the report, then checks the
reporting queue to see if the report is completed, and when the report returns as "done" pulls the
report from the API. This checking process will occur up to the specified number of times (default
120), with a delay between status checks (default 5 seconds). If the report does not return as "done"
after the number of tries have completed, the function will return an error message.

Value

Data frame
QueueRanked

Examples

```r
## Not run:
pathpattern <- c("Home Page", "::anything::", "::anything::", "::anything::")
queue_pathing_pages <- QueuePathing("your_report_suite",
  "2014-04-01",
  "2014-04-20",
  metric="pageviews",
  element="page",
  pathpattern
)

enqueued.report.id <- QueuePathing("your_report_suite",
  "2014-04-01",
  "2014-04-20",
  metric="pageviews",
  element="page",
  pathpattern,
  enqueueOnly=TRUE
)
```

## End(Not run)

---

QueueRanked  
Run a Ranked Report

**Description**

A QueueRanked report is a report that shows the ranking of values for one or more elements relative to a metric, aggregated over the time period selected.

**Usage**

```r
QueueRanked(reportsuite.id, date.from, date.to, metrics, elements, top = 10,
  start = 1, selected = c(), search = c(), search.type = "or",
  segment.id = "", segment.inline = "", classification = c(),
  data.current = FALSE, expedite = FALSE, interval.seconds = 5,
  max.attempts = 120, validate = TRUE, enqueueOnly = FALSE)
```

**Arguments**

- `reportsuite.id`  Report suite id
- `date.from`  Start date for the report (YYYY-MM-DD)
- `date.to`  End date for the report (YYYY-MM-DD)
- `metrics`  List of metrics to include in the report
- `elements`  List of elements to include in the report
- `top`  List of numbers to limit the number of rows to include (top X). eg. c(10,5)
QueueRanked

start
Start row if you do not want to start at #1 - only applies to the first element.

selected
List of specific items (of the first element) to include in the report - e.g. c("www:home","www:search","www:about"), this only works for the first element (API limitation).

search
List of keywords for the first specified element - e.g. c("contact","about","shop"). search overrides anything specified using selected

search.type
String specifying the search type: 'and', or, 'or' 'not' (defaults to 'or')

segment.id
Id(s) of Adobe Analytics segment to retrieve the report for

segment.inline
Inline segment definition

classification
SAINT classification to use in place of first element. Need to specify element AND classification.

data.current
TRUE or FALSE - whether to include current data for reports that include today's date

expedite
Set to TRUE to expedite the processing of this report

interval.seconds
How long to wait between attempts

max.attempts
Number of API attempts before stopping

validate
whether to submit report definition for validation before requesting the data.

tenqueueOnly
only enqueue the report, don’t get the data. returns report id, which you can later use to get the data

Details

The QueueRanked function returns a data frame equivalent to pulling a Ranked report in Adobe Reports & Analytics. Correlations & Sub-Relations are supported.

Because of the Reporting API structure, this function first requests the report, then checks the reporting queue to see if the report is completed, and when the report returns as "done" pulls the report from the API. This checking process will occur up to the specified number of times (default 120), with a delay between status checks (default 5 seconds). If the report does not return as "done" after the number of tries have completed, the function will return an error message.

Note: Because of the multiple argument types ("top" and "start" OR "selected"), keyword arguments are generally needed towards the end of the function call instead of just positional arguments.

Value
Data frame

Examples

## Not run:

ranked1 <- QueueRanked("your_report_suite",
    date.from = "2014-04-01",
    date.to = "2014-04-20",
    metrics = "pageviews",
    elements = c("sitesection", "page")
)
QueueSummary

**Run a Summary Report**

**Description**

A QueueSummary report is a summary report of metrics for one or more report suites for a given time period. Time period in the date parameter can be specified as year only ("2015"), year-month ("2015-04") or year-month-day ("2015-04-20"); alternatively, date.to and date.from are available for custom date ranges.

**Usage**

QueueSummary(reportsuite.ids, date = "", metrics, interval.seconds = 5, max.attempts = 120, validate = TRUE, date.from = "", date.to = "", enqueueOnly = FALSE)

**Arguments**

- **reportsuite.ids**
  - Report suite ids
- **date**
  - Time period for the report (see Description)
- **metrics**
  - List of metrics to include in the report
- **interval.seconds**
  - How long to wait between attempts
- **max.attempts**
  - Number of API attempts before stopping
- **validate**
  - whether to submit report definition for validation before requesting the data.
QueueTrended

Details

The QueueSummary function returns a data frame containing a metric summary for the time period selected.

Because of the Reporting API structure, this function first requests the report, then checks the reporting queue to see if the report is completed, and when the report returns as "done" pulls the report from the API. This checking process will occur up to the specified number of times (default 120), with a delay between status checks (default 5 seconds). If the report does not return as "done" after the number of tries have completed, the function will return an error message.

Value

Data frame

Examples

```r
## Not run:
aa <- QueueSummary("zwitchdev", "2015", c("pageviews", "visits"))
bb <- QueueSummary("zwitchdev", ", ", c("pageviews", "visits"),
  date.from = "2016-01-01", date.to="2016-01-15")
enqueued.reprot.id <- QueueSummary("zwitchdev", ", ", c("pageviews", "visits"),
  date.from = "2016-01-01", date.to="2016-01-15",
  enqueueOnly=TRUE)

## End(Not run)
```

QueueTrended

Run a Trended Report

Description

A QueueTrended report is a report where a single metric is retrieved, broken down by an element such as page, eVar, prop, etc. and with a time component. Within the 'element' list, either the "Top X" number of elements can be received or you can specify the specific elements you are interested in (such as 3 specific page names).
QueueTrended(reportsuite.id, date.from, date.to, metrics, elements, top = 0, start = 0, selected = c(), search = c(), search.type = "or", date.granularity = "day", segment.id = "", segment.inline = "", classification = character(0), anomaly.detection = FALSE, data.current = FALSE, expedite = FALSE, interval.seconds = 5, max.attempts = 120, validate = TRUE, enqueueOnly = FALSE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reportsuite.id</td>
<td>Report suite id</td>
</tr>
<tr>
<td>date.from</td>
<td>Start date for the report (YYYY-MM-DD)</td>
</tr>
<tr>
<td>date.to</td>
<td>End date for the report (YYYY-MM-DD)</td>
</tr>
<tr>
<td>metrics</td>
<td>List of metrics to include in the report</td>
</tr>
<tr>
<td>elements</td>
<td>List of elements to include in the report</td>
</tr>
<tr>
<td>top</td>
<td>List of numbers to limit the number of rows to include (top X). eg. c(10,5)</td>
</tr>
<tr>
<td>start</td>
<td>Start row if you do not want to start at #1</td>
</tr>
<tr>
<td>selected</td>
<td>List of specific items (of the first element) to include in the report - e.g. c(&quot;www:home&quot;,&quot;www:search&quot;,&quot;www&quot;)</td>
</tr>
<tr>
<td>search</td>
<td>List of keywords for the first specified element - e.g. c(&quot;contact&quot;,&quot;about&quot;,&quot;shop&quot;). search overrides anything specified using selected</td>
</tr>
<tr>
<td>search.type</td>
<td>String specifying the search type: 'and', or, 'or' 'not' (defaults to 'or')</td>
</tr>
<tr>
<td>date.granularity</td>
<td>Time granularity of the report (year/month/week/day/hour), default to 'day'</td>
</tr>
<tr>
<td>segment.id</td>
<td>Id(s) of Adobe Analytics segment to retrieve the report for</td>
</tr>
<tr>
<td>segment.inline</td>
<td>Inline segment definition</td>
</tr>
<tr>
<td>classification</td>
<td>SAINT classification to use in place of first element. Need to specify elementAND classification.</td>
</tr>
<tr>
<td>anomaly.detection</td>
<td>Set to TRUE to include forecast data (only valid for day granularity with small date ranges)</td>
</tr>
<tr>
<td>data.current</td>
<td>TRUE or FALSE - whether to include current data for reports that include today’s date</td>
</tr>
<tr>
<td>expedite</td>
<td>Set to TRUE to expedite the processing of this report</td>
</tr>
<tr>
<td>interval.seconds</td>
<td>How long to wait between attempts</td>
</tr>
<tr>
<td>max.attempts</td>
<td>Number of API attempts before stopping</td>
</tr>
<tr>
<td>validate</td>
<td>whether to submit report definition for validation before requesting the data.</td>
</tr>
<tr>
<td>enqueueOnly</td>
<td>only enqueue the report, don’t get the data. returns report id, which you can later use to get the data</td>
</tr>
</tbody>
</table>
Details

The QueueTrended report is analogous to pulling a "trended" report within Adobe Reports & Analytics, but without the limitation of only 5 elements as in the Adobe Reports & Analytics interface.

Because of the Reporting API structure, this function first requests the report, then checks the reporting queue to see if the report is completed, and when the report returns as "done" pulls the report from the API. This checking process will occur up to the specified number of times (default 120), with a delay between status checks (default 5 seconds). If the report does not return as "done" after the number of tries have completed, the function will return an error message.

Note: Because of the multiple argument type ("top" and "start" OR "selected"), keyword arguments are generally needed towards the end of the function call instead of just positional arguments.

Value

Data frame or report id, if enqueueOnly is TRUE

Examples

```r
## Not run:
report.data <- QueueTrended("your_report_suite",
    "2014-01-01",
    "2014-01-07",
    c("visits","uniquevisitors","pageviews","event10"),
    c("page","geoCountry","geoCity")
)
## End(Not run)
```

RSiteCatalyst  R Client for Adobe Analytics API V1.4

Description

This package contains an "analyst’s toolbox" of functions for accessing the Adobe Analytics Reporting API v1.4. These functions allow the user to authenticate, get metadata about report suites (eVars, props, events, segments, etc.), and create reports using Adobe Analytics data.

This package is not intended for Adobe Analytics administration.

Details

Package: RSiteCatalyst
Type: Package
Version: 1.4.15
Date: 2018-04-21
License: MIT + file LICENSE
Author(s)
Willem Paling, Randy Zwitch, Jowanza Joseph

References
Official Adobe Analytics API documentation:
For support & bugs:
https://github.com/randyzwitch/RSiteCatalyst

SaveRealTimeSettings  Save Configuration for Real-Time Report

Description
Sets the metrics and elements (dimensions) on which you want real time reports enabled via list objects created by BuildRealTimeReportStructure. Realtime configuration changes take 15 minutes to be reflected in reports.

Usage
SaveRealTimeSettings(reportsuite.ids = "", report1 = list(), report2 = list(), report3 = list())

Arguments

reportsuite.ids  Report Suite ID
report1  Real Time Report 1
report2  Real Time Report 2
report3  Real Time Report 3

Details
SaveRealTimeSettings should be called each time you want to modify the structure of your real-time reports. If you are unsure of your current setup of your real-time reports, use GetRealTimeSettings to find out your current setup.

WARNING: This function allows you to change the settings in your Adobe Analytics UI for all users, so be sure this is what you want to do. Additionally, submitting this function with only one report will mean other reports are deleted, you’re NOT just changing a single report.

NOTE: If the ui_report parameter is set to false, you must save at least one element and one metric or the configuration will be invalid, even though an error does not occur. If the ui_report parameter is set to true, you must save three elements and one metric or you will receive an error.

Changes can take up to 15 minutes to be reflected.
SCAuth

Store Credentials for the Adobe Analytics API

Description
SCAuth

Usage
SCAuth(key, secret, company = "", token.file = "", auth.method = "legacy", debug.mode = FALSE, endpoint = "", locale = "en_US")

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>Client id from your app in the Adobe Marketing cloud Dev Center OR if you are using auth.method='legacy', then this is the API username (username:company)</td>
</tr>
<tr>
<td>secret</td>
<td>Secret from your app in the Adobe Marketing cloud Dev Center OR if you are using auth.method='legacy', then this is the API shared secret</td>
</tr>
<tr>
<td>company</td>
<td>Your company (only required if using OAUTH2 AUTH method)</td>
</tr>
<tr>
<td>token.file</td>
<td>If you would like to save your OAUTH token and other auth details for use in future sessions, specify a file here. The method checks for the existence of the file and uses that if available.</td>
</tr>
<tr>
<td>auth.method</td>
<td>Defaults to legacy, can be set to 'OAUTH2' to use the newer OAUTH method.</td>
</tr>
<tr>
<td>debug.mode</td>
<td>Set global debug mode</td>
</tr>
<tr>
<td>endpoint</td>
<td>Set Adobe Analytics API endpoint rather than let RSiteCatalyst decide (not recommended)</td>
</tr>
<tr>
<td>locale</td>
<td>Set encoding for reports (defaults to en_US)</td>
</tr>
</tbody>
</table>

See Also
GetRealTimeSettings
BuildRealTimeReportStructure

Examples
## Not run:
saverealt ime <- SaveRealTimeSettings("your-report-suite", report1, report2, report3)

## End(Not run)
SubmitJsonQueueReport

**Details**

Authorise and store credentials for the Adobe Analytics API

**Value**

Global credentials list 'SC.Credentials' in AdobeAnalytics (hidden) environment

**References**

The list of locale values can be obtained from the Adobe Analytics documentation:
https://marketing.adobe.com/developer/documentation/analytics-reporting-1-4/r-reportdescriptionlocale

**Examples**

```r
## Not run:
#Legacy authentication
SCAuth("key", "secret")

## End(Not run)
```

---

SubmitJsonQueueReport  *Create Queue Report from JSON*

**Description**

Generic interface to validate, queue and retrieve a report from the API

**Usage**

```
SubmitJsonQueueReport(report.description, interval.seconds = 5,
max.attempts = 120, validate = TRUE, enqueueOnly = FALSE,
format = "json")
```

**Arguments**

- `report.description`  
  JSON report description
- `interval.seconds`  
  How long to wait between attempts
- `max.attempts`  
  Number of API attempts before stopping
- `validate`  
  whether to submit report definition for validation before requesting the data.
- `enqueueOnly`  
  only enqueue the report, don’t get the data. returns report id, which you can later use to get the data
- `format`  
  "csv" or "json"
ViewProcessingRules

Details

This is a function for advanced users, for the case where a user feels that submitting a JSON request would be easier than using one of the pre-defined functions from RSiteCatalyst.

Value

Data frame or report id, if enqueueOnly is TRUE.

Examples

```r
## Not run:
custom_report <- SubmitJsonQueueReport('valid Adobe Analytics API JSON string')
## End(Not run)
```

ViewProcessingRules  View Processing Rules

Description

Get Processing Rules with title and actions.

Usage

```r
ViewProcessingRules(reportsuite.ids)
```

Arguments

- `reportsuite.ids`:
  Report suite id (or list of report suite ids).

Details

This function requires having a character vector with one or more valid Report Suites specified.

Value

Data frame.
Examples

## Not run:

```r
pr <- ViewProcessingRules("your-report-suite")
pr <- ViewProcessingRules(c("your-report-suite", "your-report-suite2"))
```

## End(Not run)
Index

*Topic **BuildRealTimeReportStructure**
  BuildRealTimeReportStructure, 4

*Topic **GetRealTimeSettings**
  GetRealTimeSettings, 37

*Topic **SaveRealTimeSettings**
  SaveRealTimeSettings, 64

BuildClassificationValueSegment, 3
BuildRealTimeReportStructure, 4, 65

CancelReport, 5

GetActivation, 6
GetAxleStartDate, 7
GetBaseCurrency, 7
GetBaseURL, 8
GetBookmarks, 9
GetCalculatedMetrics, 9
GetClassifications, 10
GetClickMapReporting, 11
GetCustomCalendar, 12
GetDashboards, 12
GetDataWarehouseDisplay, 13
GetDefaultPage, 14
GetDiscoverEnabled, 15
GetEcommerce, 15
GetElements, 16
GetEvars, 17
GetFeed, 18
GetFeeds, 18
GetFunctions, 19
GetGeoSegmentation, 20
GetGroup, 21
GetGroups, 21
GetInternalURLFilters, 22
GetIPAddressExclusions, 23
GetIPObfuscation, 24
GetKeyVisitors, 24
GetListVariables, 25
GetLocalization, 26
GetLogin, 26
GetLogins, 27
GetMarketingChannelExpiration, 28
GetMarketingChannelRules, 28
GetMarketingChannels, 29
GetMetrics, 30
GetMobileAppReporting, 31
GetPaidSearchDetection, 32
GetPermanentTraffic, 32
GetPreviousServerCalls, 33
GetPrivacySettings, 34
GetProps, 35
GetQueue, 35
GetRealTimeReport, 36
GetRealTimeSettings, 5, 37, 65
GetReport, 38
GetReportDescription, 39
GetReportsByIds, 40
GetReportSuiteGroups, 41
GetReportSuites, 41
GetScheduledSpike, 42
GetSegments, 43
GetSiteTitle, 43
GetSuccessEvents, 44
GetTemplate, 45
GetTimeStampEnabled, 46
GetTimeZone, 46
GetTrackingServer, 47
GetTransactionEnabled, 48
GetUniqueVisitorVariable, 48
GetVersionAccess, 49
GetVideoSettings, 50
GetVirtualReportSuiteSettings, 50

QueueDataWarehouse, 51
QueueFallout, 53
QueueOvertime, 54
QueuePathing, 56
QueueRanked, 58
QueueSummary, 60
QueueTrended, 61

RSiteCatalyst, 63
RSiteCatalyst-package (RSiteCatalyst), 63

SaveRealTimeSettings, 5, 37, 64
SCAuth, 65
SubmitJsonQueueReport, 66

ViewProcessingRules, 67