Package 'oRion'

August 3, 2017

Type Package

Title Integrate R & Cheetah Mobile Orion

Version 0.0.1

Date 2016-03-29

Description Integrate R and Cheetah Mobile Orion platform.
All API calls are implemented as individual functions.

License MIT + file LICENSE

LazyData TRUE

RoxygenNote 6.0.1

Depends R (>= 3.0.0)

Imports jsonlite,
httr,
plyr,
methods

URL https://github.com/JohnCoene/oRion

BugReports https://github.com/JohnCoene/oRion/issues

Suggests testthat,
lintr

R topics documented:

`createAd` *Create an ad*

Description

Create an ad.

Usage

```
createAd(body)
```

Arguments

`body`	Body of request that includes ad settings. See examples and details.

Details

Valid ad settings (body):

- `adset_id` Adset id to which the ad belongs.
- `name` Ad name, unique.
- `icon_url` Logo image url. Must be larger than 84*84 px and have an aspect ratio of 1:1, in JPG or PNG format. File must not exceed 100KB.
- `title` Ad headline
- `desc` Description tells people a bit more about your Application. Make sure to clearly explain what you are promoting.
- `button_text` Button display text, use `buttonText` to generate otherwise, `1`: Download, `2`: Install, `3`: Check, `4`: Free, `5`: Play, `6`: Buy, `7`: More.
- `image_url` Image url. Must be larger than 600*314 px and have an aspect ratio of 1.9:1. Best is 1200*628 px, JPG or PNG format. File size must not exceed 500KB. Image should come from Orion CDN server, please upload image using `createImage`, see uploaded images with `listImages`.
- `video_url` URL to video
- `video_img_before_url` Image which will be show before video starts.
- `video_img_after_url` Image which will be show after video ends.

Author(s)

John Coene <jcoenep@gmail.com>

See Also

`orionOAuth`, `createAdset`, `createImage`, `helpers`

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# list adsets
adsets <- listAdsets()

# list images
images <- listImages()

# define ad settings
body <- list(
   adset_id = adsets$id[1],
   name = "testAd",
   icon_url = images$thumb[1],
   title = "Download oRion",
   desc = "Download oRion and interact with Orion from your R console",
```

```
    button_text = buttonText("install"),
    image_url = images$local_url[1])

# post ad
createAd(body = body)

## End(Not run)
```

`createAdset` *Create an adset*

Description

Create an adset.

Usage

```
createAdset(body)
```

Arguments

body	body of request that includes adset settings. See examples and details

Details

Valid adset settings (body):

- `campaign_id` The campaign id which the ad set is to belongs to
- `audience_create_way` `0`: Create audience targeting parameters directly. Audience should be along with this API if choose this audience_create_way. `1`: Create audience targeting by a template. Note: Once audience created, any changes made on the template will not affect this audience.
- `audience_template_id` Required when `audience_create_way`: `1` or `2`
- `bid_type` Use `bidType` to generate otherwise, `1`: Install (CPI) [Only for KA user], `3`: Click (CPC), `4` Impressions (CPM)
- `app_show_type` Use `appShowType` to generate otherwise, `50000`: Native News Feed, `50001`: Mini Native News Feed, `50003`: Video Ad, `50008`: App Locker Ad
- `name` Ad set name (unique)
- `unit_price` Unit Price. Amount you want to spend to pay per click (CPC) or per 1,000 impressions (CPM)
- `click_url` click_url, protocol is required (http://, https://).
- `deeplink` Deep link is the ability to link into a specific page or function inside of your app, making an app linkable just like a web site. If you are only targeting people who already installed your app, you do not need to add deferred deep linking. Used when the objective of campaign which ad set belonged to is "Get App Reengagement" (`objective = 4`)

- `imp_url` Impression Tracking. An Impression tracking is an optional URL that allows you to track how many people viewed your ads through third-party tracking system, and our system supported macro is Gaid, e.g. http://host/imp?gaid=gaid. Used when the bid type is "CPM" `bid_type = 4`
- `budget_lifetime` Your budget is the maximum amount you want to spend. If you choose lifetime, the amount you enter is the maximum you'll spend during the lifetime of your ad set. At least one of `budget_daily` and `budget_lifetime` is filled.
- `budget_daily` Your budget is the maximum amount you want to spend. If you choose Per Day, the amount you enter is the maximum you'll spend each day. At least one of "budget_daily" and "budget_lifetime" is filled.
- `start_time` Start time for the adset (i.e.: `2015-09-09 10:10`)
- `end_time` End time for the adset (i.e.: `2015-09-09 10:10`)
- `ad_scheduling` Your ad set will either run continuously within some days or within some hours range you select. (`JSON`) i.e.: `{1:[1,2],7:[1,2,23]}`
- `delivery_type` Use `deliveryType` to generate otherwise, `0`: Standard delivery, `1`: Accelerated delivery. Standard delivery is recommended and the preferred option for most advertisers. Accelerated delivery can be useful for promoting time-sensitive events and quickly reaching a target audience.
- `target_cpi` Target CPI. Only for KA user. `3`: Click (CPC)
- `freq_type` Frequency Capping type. Frequency capping controls frequency capping of the ad set per user.
- `freq_times` Frequency Capping type. Frequency capping controls frequency capping of the ad set per user.
- `country` Audience country code. Narrow your potential audience to the people in those countries. Using '|' separated if more than one country. i.e.: `US|ID|CN`. See `dictCountry`
- `language` Audience language code, i.e.: `en` see `dictLanguage`
- `gender` Use `gender` to generate otherwise, `0`: All, `1`: Male, `2` Female
- `age` Audience age. Use `age` to generate otherwise, `1`: 18-24, `2`: 25-30, `3`: 32-40, `4`: 41+.
- `interest` Audience interest. Reach users based on their specific interests. See `dictInterest`
- `behavior` Audience behavior. Reach users based on app usage and other behaviors. See `dictBehaviour`
- `device_brand` Audience device brand. See `dictDevice`
- `min_device_os` Min OS version of audience used, blank for unrestricted. See `dictOS`
- `max_device_os` Max OS version of audience used, blank for unrestricted. See `dictOS`
- `net_type` Use `netType` to generate otherwise, `0`: All, `1`: Wifi, `2`: 2G/3G/4G
- `dsp_url` Bid request url of DSP when this pre-targeting condition meets.

Author(s)

John Coene <jcoenep@gmail.com>

See Also

`orionOAuth`, `createCampaign`, `helpers`

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# list of campaigns
camps <- listCampaigns()

# list of audience templates
audiences <- listAudiences()

# create adset using random template, in random campaign
createAdset(body = list(
   name = "testAdset",
   bid_type = bidType("cpi"),
   unit_price = 1000,
   budget_lifetime = 10000,
   campaign_id = sample(camps$id, 1),
   audience_create_way = 2,
   app_show_type = appShowType("newsfeed"),
   audience_template_id = sample(audiences$id, 1),
   click_url = "http://app.adjust.io"))

## End(Not run)
```

`createAudience` *Create an audience targeting template*

Description

Create an audience targeting template.

Usage

```
createAudience(body)
```

Arguments

body	body of request that includes audience targeting template settings. See examples and details.

Details

Valid audience target template settings (body):

- `audience_template_name` Audience targeting template name (unique).
- `audience_template_desc` Description for the audience targeting template.

- country Audience country code. Narrow your potential audience to the people in those countries. Using 'l' separated if more than one country. i.e.: US|ID|CN, typically returned by dictCountry
- language Audience language code, i.e.: en, typically returned by dictLanguage
- gender Use gender to generate otherwise, 0: All, 1: Male, 2 Female
- age Audience age. Use age to generate otherwise, 1: 18-24, 2: 25-30, 3: 32-40, 4: 41+.
- interest Audience interest. Reach users based on their specific interests, typically returned by dictInterest
- behavior Audience behavior. Reach users based on app usage and other behaviors, typically returned by dictBehaviour
- device_brand Audience device brand, typically returned by dictDevice
- min_device_os Min OS version of audience used, blank for unrestricted, typically returned by dictOS
- max_device_os Max OS version of audience used, blank for unrestricted, typically returned by dictOS
- net_type Use netType to generate otherwise, 0: All, 1: Wifi, 2: 2G/3G/4G
- dsp_url Bid request url of DSP when this pre-targeting condition meets.

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth, helpers, dictCountry, dictLanguage, dictInterest, dictBehaviour, dictDevice, dictOS

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# pick two random countries
locations <- paste0(sample(dictCountry()$code, 1), "|",
                    sample(dictCountry()$code, 1))

# create audience
createAudience(body = list(
   audience_template_name = "test",
   audience_template_desc = "test template",
   country = locations,
   language = sample(dictLanguage()$code, 1),
   net_type = netType("wifi")),
   interest = dictInterst()$pid[1])

# retrieve audience
```

```
aud <- listAudiences(n = 100)

## End(Not run)
```

`createCampaign` *Create a campaign.*

Description

Create a campaign

Usage

```
createCampaign(body)
```

Arguments

body	Body of request that includes campaigns settings. See examples and details.

Details

Valid campaign settings (body):

- `name` Campaign name should be unique under the same user account.
- `budget_type` can take either `daily` or `lifetime`, see `budgetType`.
- `budget_daily` Required when `budget_type` equals to `daily`
- `budget_lifetime` Required when `budget_type` equals to `lifetime`
- `pkg_name` App package name or website domain
- `objective` Use `objective` to generate, otherwise, `1`: Drive App Installs, `2`: Drive Mobile Site Traffic, `3`: Build Brand Awareness, `4`: Get App Re-engagement, `5`: Get Video Views
- `app_type` Use `appType` to generate, otherwise, `1`: Game, `2`: App
- `web_type` Use `webType` to generate, otherwise, if objective == `2`. `0`: None, `1`: Ordinary Website, `2`: E-commerce Website, `3`: Game Website
- `landing_page` Use `landingPage` to generate, otherwise, if objective == `3`. `1`: Redirect to Google Play, `2`: Website

Author(s)

John Coene <jcoenep@gmail.com>

See Also

`orionOAuth`, `helpers`

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# set body of request
camp <- list(name = "test",
             budget_type = "daily",
             budget_daily = "1000",
             pkg_name = "someting.test.com",
             objective = objective("installs"),
             app_type = "2",
             web_type = "1",
             landing_page = landingPage("googlePlay"))

# create campaign
createCampaign(body = camp)

## End(Not run)
```

`createImage` *Upload an image*

Description

Upload an image

Usage

```
createImage(file)
```

Arguments

`file`	Image data, supports JPEG, GIF and PNG.

Author(s)

John Coene <jcoenep@gmail.com>

See Also

`orionOAuth`

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# upload image
createImage(file = paste0(.libPaths(), "/png/img/Rlogo.png"))

## End(Not run)
```

`createObject` *Create any object*

Description

Create any object, see details.

Usage

```
createObject(object, body)
```

Arguments

`object`	Object to be created, see details for valid values.
`body`	Body of request that includes campaigns settings. See examples.

Details

Valid values for `object`:

- `audience`, see `createAudience` for details.
- `campaign`, see `createCampaign` for details.
- `adset`, see `createAdset` for details.
- `ad`, see `createAd` for details.

`createObject` can essentially replace any other `create` family functions. See examples.

Author(s)

John Coene <jcoenep@gmail.com>

See Also

`orionOAuth`, `createAudience`, `createCampaign`, `createAdset`, `createAd` and `helpers`

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# set body of request for campaign
camp <- list(name = "test",
             budget_type = "daily",
             budget_daily = "1000",
             pkg_name = "test",
             objective = objective("traffic"),
             app_type = appType("game"),
             web_type = webType("game"),
             landing_page = "2")

# create campaign
new_camp <- createObject(body = camp, object = "campaign")

# equivalent to
# new_camp <- createCampaign(body = camp)

showCampaign(campaign.id = new_camp$id)

## End(Not run)
```

`deleteAd` *Delete an ad*

Description

Delete an ad.

Usage

```
deleteAd(ad.id)
```

Arguments

`ad.id`	id of object to delete

Author(s)

John Coene <jcoenep@gmail.com>

See Also

`orionOAuth`, `createAd`

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# list adsets
adsets <- listAdsets()

# list images
images <- listImages()

# create an ad
# define ad settings
body <- list(
   adset_id = adsets$id[1],
   name = "My Ad",
   icon_url = images$thumb[1],
   title = "Ad Title",
   desc = "Ad description",
   button_text = buttonText("install"),
   image_url = images$local_url[1])

# post ad
ad <- createAd(body = body)

# delete the ad
deleteAd(ad.id = ad$id)

## End(Not run)
```

---

deleteAdset *Delete an adset*

Description

Delete an adset.

Usage

```
deleteAdset(adset.id)
```

Arguments

adset.id	id of object to delete

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth, createAdset

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# list of campaigns
camps <- listCampaigns()

# list of audience templates
audiences <- listAudiences()

# create adset using random template, in random campaign
adset <- createAdset(body = list(
   name = "My Adset",
   bid_type = bidType("CPI"),
   unit_price = 1000,
   budget_lifetime = 10000,
   campaign_id = sample(camps$id, 1),
   audience_create_way = 2,
   app_show_type = appShowType("newsfeed"),
   audience_template_id = sample(audiences$id, 1),
   click_url = "http://app.adjust.io"))

# delete random adset
deleteAdset(adset.id = adset$id)

## End(Not run)
```

---

deleteAudience *Delete an audience targeting template*

Description

Delete audience targeting template.

Usage

```
deleteAudience(audience.id)
```

Arguments

audience.id	id of object to delete

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth, createAudience

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# list audiences
audiences <- listAudiences()

# delete random aduience template
deleteAudience(audience.id = sample(audiences$id, 1))

## End(Not run)
```

---

deleteCampaign *Delete a campaign*

Description

Delete a campaign.

Usage

```
deleteCampaign(campaign.id)
```

Arguments

campaign.id	id of object to delete

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth, createCampaign

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# set body of request
camp <- list(name = "My Campaign",
             budget_type = "daily",
             budget_daily = "1000",
             pkg_name = "someting.test.com",
             objective = objective("installs"),
             app_type = "2",
             web_type = "1")

# create campaign
campaign <- createCampaign(body = camp)

# delete random campaign
deleteCampaign(campaign.id = campaign$id)

## End(Not run)
```

`deleteObject` *Delete any object*

Description

Delete campaign, adsets, ads or audience temapltes, see details and examples.

Usage

```
deleteObject(object, id)
```

Arguments

`object`	Object to be deleted, see details for valid values.
`id`	id of object to delete

Details

Valid values for `object`:

- `audience`, see `deleteAudience` for details.
- `campaign`, see `deleteCampaign` for details.
- `adset`, see `deleteAdset` for details.
- `ad`, see `deleteAd` for details.

`deleteObject` can essentially replace any other `delete` family functions. See examples.

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth, deleteAudience, deleteCampaign, deleteAdset and deleteAd

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# list ads
ads <- listObjects(object = "ad")

# delete
# set seed for reproducibility
set.seed(19880525)

# delete random ad
deleteObject(object = "ad", id = sample(ads$id, 1))

# the above is equivalent to
# deleteAd(ad.id = sample(ads$id, 1))

## End(Not run)
```

---

dictBehaviour *List available behaviours*

Description

Fetch list of behaviours.

Usage

```
dictBehaviour()
```

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth, createAudience

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# dict behaviours
(behaviours <- dictBehaviour())

# create target audience template
createAudience(body = list(
               audience_template_name = "target game addicts",
               behavior = behaviours$code[1]
               ))

## End(Not run)
```

`dictCity` *List valid cities*

Description

Fetch list of cities.

Usage

```
dictCity(state.code)
```

Arguments

`state.code`	Code of country as returned by `dictState`, see examples.

Author(s)

John Coene <jcoenep@gmail.com>

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# dict states of random country
states <- dictState(country.code = "US")

# dict cities
cities <- dictCity(state.code = sample(states$code, 1))
```

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

---

dictCountry *List valid countries*

Description

Fetch list of countries.

Usage

```
dictCountry()
```

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth, createAudience

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# dict countries
countries <- dictCountry()

# create audience with country settings
createAudience(body = list(
               audience_template_name = "countries",
               country = paste0(countries$code[1:5], collapse="|")
               ))

## End(Not run)
```

dictDevice *List valid device models*

Description

Fetch list of device models.

Usage

```
dictDevice()
```

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth, createAudience

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# dict mobile devices
(devices <- dictDevice())

# create audience with device setting
createAudience(body = list(
               audience_template_name = "target specific device users",
               device_brand = sample(devices$id, 1)
               ))

## End(Not run)
```

dictInterest *List available interests*

Description

Fetch list of interests.

Usage

```
dictInterest()
```

Author(s)

John Coene <jcoenep@gmail.com>

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# dict interests
head(int <- dictInterest())

# filter transportation-related interests
trans <- int[grep("Transportation", int$value),]

# create target audience template
createAudience(body = list(
               audience_template_name = "target specific interests",
               interest = trans$pid[1]
               ))

## End(Not run)
```

`dictLanguage` *List valid languages*

Description

Fetch list of languages.

Usage

```
dictLanguage()
```

Author(s)

John Coene <jcoenep@gmail.com>

See Also

`orionOAuth`, `createAudience`

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# dict languages
head(lang <- dictLanguage())

# filter russian language code
russian <- subset(lang, value == "Russian")$code

# create audience
createAudience(body= list(
               audience_template_name = "target Russian speakers",
               language = russian
))

## End(Not run)
```

`dictOS` *List valid OS versions*

Description

Fetch list of OS versions.

Usage

```
dictOS()
```

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth, createAudience

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# dict os versions
versions <- dictOS()
```

```
# create audience
createAudience(body = list(
               audience_template_name = "target android >= 3.0",
               min_device_os = versions[versions$value == "3.0", ]
               ))

## End(Not run)
```

`dictState` *List valid states*

Description

Fetch list of states.

Usage

```
dictState(country.code)
```

Arguments

`country.code`	Code of country as returned by `dictCountry`, see examples.

Author(s)

John Coene <jcoenep@gmail.com>

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# dict countries
country <- dictCountry()

# dict states of random country
states <- dictState(country.code = sample(country$code, 1))

# dict US states
us <- dictState(country.code = "US")

# filter states starting with A (because why not?)
a_states <- us[grep("^A", us$value),]

# create audience
createAudience(body = list(
```

```
                audience_template_name = "A-states",
                country = "US",
                region = paste0(a_states$code, collapse="|")
))

## End(Not run)
```

---

getReport *Get data*

Description

Fetch data.

Usage

```
getReport(column = c("impression", "click"), group.by = c("datetime",
  "campaign"), action = "advertiser", filter = NULL, start,
  end = Sys.Date())
```

Arguments

column	Variables to retrieve, see details for valid values. Defaults to c("impression", "click"). Must pass at least 2 values (vector of length >= 2). Required.
group.by	How to breakdown the data, see @details for valid values. Defaults to c("datetime", "campaign"). Must pass at least 2 values (vector of length >= 2). Required.
action	target node, currently only supports "advertiser" (default).
filter	Defaults to NULL. Optional.
start	Start date (YYYY-MM-DD). Required.
end	End date (YYYY-MM-DD), defaults to Sys.Date(). Required.

Details

The official documentation can be found here.

Valid values for column:

- impression
- click
- conversion
- revenue
- ctr
- cpm

Valid values for group.by:

- age
- gender
- location
- brand
- ad
- adset
- campaign
- videotype

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# get daily campaign impressions and conversions for the past 7 days
dat <- getReport(column = c("impression", "conversion"),
                 group.by = c("datetime", "campaign"),
                 start = Sys.Date() - 7)

## End(Not run)
```

---

helpers *Get the settings right*

Description

These functions attempt to facilitate setting up campaigns, adsets, ads and audience targeting templates by allowing to pass more readable settings

Usage

```
budgetType(x)

objective(x)

appType(x)

webType(x)

landingPage(x)

bidType(x)

appShowType(x)

deliveryType(x)

gender(x)

age(x)

netType(x)

buttonText(x)

switchIt(x)

audienceCreation(x)
```

Arguments

x	Setting to convert

Value

Returns correct value to pass to API.

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth, createCampaign, createAdset, createAd and createAudience

Examples

```
## Not run:
# authenticate
```

```
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# settings with helpers
settings  <- list(name = "test",
                  budget_type = budgetType("daily"),
                  budget_daily = "1000",
                  pkg_name = "test.com",
                  objective = objective("installs"),
                  app_type = appType("game"),
                  web_type = webType("ecommerce"),
                  landing_page = landingPage("googlePlay"))

# settings WITHOUT helpers
nohelpers  <- list(name = "test",
                   budget_type = "daily",
                   budget_daily = "1000",
                   pkg_name = "test.com",
                   objective = "1",
                   app_type = "1",
                   web_type = "2",
                   landing_page = "1")

identical(settings, nohelpers)

# create campaign
createCampaign(body = settings)

## End(Not run)
```

`listAds` *List ads*

Description

List ads under the authenticated account.

Usage

```
listAds(n = 50)
```

Arguments

`n`	Number of ads to retrieve, defaults to `50`

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

(ads <- listAds(n = 100))

## End(Not run)
```

---

listAdsets *List adsets*

Description

List of adsets under the authenticated account.

Usage

```
listAdsets(n = 50)
```

Arguments

n	Number of adsets to retrieve, defaults to 50

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

(adsets <- listAdsets(n = 50))

## End(Not run)
```

listAudiences *List audience targeting templates*

Description

List of audience templates under the authenticated account.

Usage

```
listAudiences(n = 50)
```

Arguments

n	Number of audience templates to retrieve, defaults to 50

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

(aud <- listAudiences(n = 50))

## End(Not run)
```

listCampaigns *List campaigns*

Description

List of campaigns under the authenticated account.

Usage

```
listCampaigns(n = 50)
```

Arguments

n	Number of campaigns to retrieve, defaults to 50

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

head(camps <- listCampaigns(n = 100))

## End(Not run)
```

---

listImages *List images*

Description

List of images under the authenticated account.

Usage

```
listImages(n = 50)
```

Arguments

n	Number of images to retrieve, defaults to 50

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

(images <- listImages())

## End(Not run)
```

`listObjects` *List any object*

Description

List any objects from the API, see details for valid values.

Usage

```
listObjects(object, n = 50)
```

Arguments

`object`	Nodes to target. See details
`n`	Number of objects to retrieve, defaults to `50`

Details

Valid values for `objects`:

- `image`, see `listImages` for details.
- `audience`, see `listAudiences` for details.
- `campaign`, see `listCampaigns` for details.
- `adset`, see `listAdsets` for details.
- `ad`, see `listAds` for details.

`listObject` can essentially replace any other `list` family functions. See examples.

Author(s)

John Coene <jcoenep@gmail.com>

See Also

`orionOAuth`, `listImages`, `listAudiences`, `listCampaigns`, `listAdsets` and `listAds`

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# list campaigns with listCampaigns
camps <- listCampaigns()

# list campaigns with listObjects
obj <- listObjects(object = "campaign")

# identical results
identical(camps, obj)

## End(Not run)
```

---

map *Visualise your account*

Description

Return edge list of objects in the authenticated user's account, to help one visualise campaigns, adsets, ads and audience templates.

Usage

```
map(campaigns, adsets, ads, audiences)
```

Arguments

campaigns	Data frame of campaigns as returned by listCampaigns
adsets	Data frame of campaigns as returned by listAdsets
ads	Data frame of campaigns as returned by listAds
audiences	Data frame of campaigns as returned by listAudiences

Details

Maps the network of object under the authenticated user, follows the typical advertising structure.

- 1User
- 2Campaigns
- 3Adsets
- 4Ads
- 4Audiences

Under the uaudthenticated user are campaigns, under campaigns lie adsets to which audience targeting templates and ads belong.

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth, listAudiences, listCampaigns, listAdsets, listAds

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# list required objects
ads <- listAds(n = 500)
adsets <- listAdsets(n = 500)
campaigns <- listCampaigns(n = 500)
audiences <- listAudiences(n = 500)

network <- map(campaigns, adsets, ads, audiences)

# plot with igraph
# plot ids
g <- igraph::graph.data.frame(network[,1:2], directed = TRUE)
plot(g)

# plot names
g <- igraph::graph.data.frame(network[,3:4], directed = TRUE)
plot(g)

# plot with networkd3
networkD3::simpleNetwork(Data = network, Source = "source.name",
                         Target = "target.name")

## End(Not run)
```

---

orionOAuth *orionAuthenticate*

Description

Authenticate the Orion API.

Usage

```
orionOAuth(client.id, client.secret, grant.type = "client_credentials",
  save = FALSE)
```

Arguments

client.id	Your client_id
client.secret	Your client_secret
grant.type	Currently only supports client_credentials (default)
save	If TRUE will save the token for future sessions, default to FALSE

Details

Please see the official documentation to apply for the API and get your client_id and client_secret: api.ori.cmcm.com/doc/#api-Auth-access_token. If the token is stored (save = TRUE) then orionOAuth does not need to be run in future sessions, see example.

Author(s)

John Coene <jcoenep@gmail.com>

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# test
camp_oauth <- listCampaigns()

# authenticate and save
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0",
           save = TRUE)

# detach and unload package
detach("package:oRion", unload = TRUE)

# no oauth
library(oRion)
camp_noauth <- listCampaigns()

identical(camp_oauth, camp_noauth)

## End(Not run)
```

showAd *Show an ad settings*

Description

Retrieve settings of a specfic ad.

Usage

```
showAd(ad.id)
```

Arguments

ad.id	id of ad to retrieve

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth, listAds

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# list ads
ads <- listAds(n = 100)

# fetch random adset settings
(ad <- showAd(ad.id = sample(ads$id, 1)))

## End(Not run)
```

showAdset *Show an adset settings*

Description

Retrieve settings of a specfic adset.

Usage

```
showAdset(adset.id)
```

Arguments

adset.id	id of adset to retrieve

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth, listAdsets

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# list adsets
adsets <- listAdsets(n = 100)

# fetch random adset settings
(adset <- showAdset(adset.id = sample(adsets$id, 1)))

## End(Not run)
```

showAdsetAudience *Show an adset audience targeting template*

Description

Retrieve the audience templates under a specific adset.

Usage

```
showAdsetAudience(adset.id)
```

Arguments

adset.id	id of adset's template to retrieve

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth, showAdset, showAudience, listAdsets, listAudiences

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# list 50 adsets
adsets <- listAdsets(n = 50)

# show audience template of random adset
audience <- showAdsetAudience(adset.id = sample(adsets$id, 1))

## End(Not run)
```

showAudience *Show an audience targeting template*

Description

Retrieve settings of a specfic audience targeting template.

Usage

```
showAudience(audience.id)
```

Arguments

audience.id	id of audience template to retrieve

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth, listAudiences

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# list ads
audiences <- listAudiences(n = 100)

# fetch random adset settings
(audience <- showAudiences(audience.id = sample(audiences$id, 1)))

## End(Not run)
```

showCampaign *Show a campaign settings*

Description

Retrieve settings of a specfic campaign.

Usage

```
showCampaign(campaign.id)
```

Arguments

campaign.id	id of campaign to retrieve

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth, listCampaigns

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# list campaigns
camps <- listCampaigns(n = 100)

# fetch random campaign settings
(camp <- showCampaign(campaign.id = sample(camps$id, 1)))

## End(Not run)
```

showObject *Show any object settings*

Description

Retrieve settings of an object, see details.

Usage

```
showObject(object, id)
```

Arguments

object	Object to retrieve, see details for valid values.
id	id of object to retrieve

Details

Valid values for object:

- audience see showAudience for details.
- campaign see showCampaign for details.
- adset see showAdset for details.
- ad see showAd for details.

showObject can essentially replace any other show family functions a the execption of showAdsetAudience. See examples.

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth, showAudience, showCampaign, showAdset and showAd

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# list ads
ads <- listObjects(object = "ad", n = 100)

# fetch first ad
ad <- showObject(object = "ad", id = ads$id[1])
```

```
# list campaigns
campaigns <- listObjects(object = "campaign")

# fetch random campaign
campaign <- showObject(id = campaigns$id[1])

# equivalent to
campaign2 <- showCampaign(campaign.id = campaigns$id[1])

identical(campaign, campaign2)

## End(Not run)
```

`updateAd` *Update an ad*

Description

Update ad settings, see details and example.

Usage

```
updateAd(ad.id, body)
```

Arguments

`ad.id`	id of ad to update
`body`	Body of request that includes ad settings. See examples and details.

Details

Valid values for `body`:

- `switch` Ad switch, use `switchIt` otherwise, `1`: On, `0`: Off.
- `name` Ad name, unique.
- `icon_url` Logo image url. Must be larger than 84*84 px and have an aspect ratio of 1:1, in JPG or PNG format. File must not exceed 100KB.
- `title` Ad headline
- `desc` Description tells people a bit more about your Application. Make sure to clearly explain what you are promoting.
- `button_text` Button display text, use `buttonText` to generate otherwise, `1`: Download, `2`: Install, `3`: Check, `4`: Free, `5`: Play, `6`: Buy, `7`: More.
- `image_url` Image url. Must be larger than 600*314 px and have an aspect ratio of 1.9:1. Best is 1200*628 px, JPG or PNG format. File size must not exceed 500KB. Image should come from Orion CDN server, please upload image using `createImage`.

- `video_url` URL to video
- `video_img_before_url` Image which will be show before video starts.
- `video_img_after_url` Image which will be show after video ends.

Author(s)

John Coene <jcoenep@gmail.com>

See Also

`orionOAuth`, `listAds`, `createAd`, `helpers`

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# list ads
ads <- listAds()

# update ad name
update <- updateAd(ad.id = ad$id[1], body = list(
               switch = switch("off"),
               name = "New Ad Name - Paused"))

# show updated campaign
showAd(ad.id = ads$id[1])

## End(Not run)
```

`updateAdset` *Update an adset*

Description

Update adset settings, see details and example.

Usage

```
updateAdset(adset.id, body)
```

Arguments

`adset.id`	id of adset to update
`body`	Body of request that includes adset settings. See examples and details.

Details

Valid values for body:

- switch Ad switch, use switchIt otherwise, 1: On, 0: Off.
- name Ad set name (unique)
- unit_price Unit Price. Amount you want to spend to pay per click (CPC) or per 1,000 impressions (CPM)
- click_url click_url, protocol is required (http://, https://).
- deeplink Deep link is the ability to link into a specific page or function inside of your app, making an app linkable just like a web site. If you are only targeting people who already installed your app, you do not need to add deferred deep linking. Used when the objective of campaign which ad set belonged to is "Get App Reengagement" (objective = 4)
- imp_url Impression Tracking. An Impression tracking is an optional URL that allows you to track how many people viewed your ads through third-party tracking system, and our system supported macro is Gaid, e.g. http://host/imp?gaid=gaid. Used when the bid type is "CPM" bid_type = 4
- budget_lifetime Your budget is the maximum amount you want to spend. If you choose lifetime, the amount you enter is the maximum you'll spend during the lifetime of your ad set. At least one of budget_daily and budget_lifetime is filled.
- budget_daily Your budget is the maximum amount you want to spend. If you choose Per Day, the amount you enter is the maximum you'll spend each day. At least one of "budget_daily" and "budget_lifetime" is filled.
- start_time Start time for the adset (i.e.: 2015-09-09 10:10)
- end_time End time for the adset (i.e.: 2015-09-09 10:10)
- ad_scheduling Your ad set will either run continuously within some days or within some hours range you select. (JSON) i.e.: {1:[1,2],7:[1,2,23]}
- delivery_type Use deliveryType to generate otherwise, 0: Standard delivery, 1: Accelerated delivery. Standard delivery is recommended and the preferred option for most advertisers. Accelerated delivery can be useful for promoting time-sensitive events and quickly reaching a target audience.
- target_cpi Target CPI. Only for KA user. 3: Click (CPC)
- freq_type Frequency Capping type. Frequency capping controls frequency capping of the ad set per user.
- freq_times Frequency Capping type. Frequency capping controls frequency capping of the ad set per user.
- country Audience country code. Narrow your potential audience to the people in those countries. Using 'I' separated if more than one country. i.e.: US|ID|CN. See dictCountry
- language Audience language code, i.e.: en see dictLanguage
- gender Use gender to generate otherwise, 0: All, 1: Male, 2 Female
- age Use age to generate otherwise, Audience age. 1: 18-24, 2: 25-30, 3: 32-40, 4: 41+.
- interest Audience interest. Reach users based on their specific interests. See dictInterest
- behavior Audience behaviour. Reach users based on app usage and other behaviors. See dictBehaviour

- device_brand Audience device brand. See dictDevice
- min_device_os Min OS version of audience used, blank for unrestricted. See dictOS
- max_device_os Max OS version of audience used, blank for unrestricted. See dictOS
- net_type Use netType to generate otherwise, 0: All, 1: Wifi, 2: 2G/3G/4G
- dsp_url Bid request url of DSP when this pre-targeting condition meets.

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth, listAdsets, createAdset, helpers, dictCountry, dictLanguage, dictInterest, dictBehaviour, dictDevice, dictOS

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# list adsets
adsets <- listAdsets()

# update adset name
update <- updateCampaign(adset.id = adsets$id[1], body = list(
               switch = switchIt("on"),
               name = "New adset Name - Running"))

# show updated campaign
showAdset(adset.id = adsets$id[1])

## End(Not run)
```

---

updateAudience *Update an audience targeting template*

Description

Update audience targeting template settings, see details and example.

Usage

```
updateAudience(audience.id, body)
```

Arguments

audience.id	id of audience to update
body	Body of request that includes audience template settings. See examples and details.

Details

Valid values for body:

- audience_template_name Audience targeting template name (unique).
- audience_template_desc Description for the audience targeting template.
- country Audience country code. Narrow your potential audience to the people in those countries. Using '|' separated if more than one country. i.e.: US|ID|CN, typically returned by dictCountry
- language Audience language code, i.e.: en, typically returned by dictLanguage
- gender Use gender to generate otherwise, 0: All, 1: Male, 2 Female
- age Audience age. Use age to generate otherwise, 1: 18-24, 2: 25-30, 3: 32-40, 4: 41+.
- interest Audience interest. Reach users based on their specific interests, typically returned by dictInterest
- behavior Audience behavior. Reach users based on app usage and other behaviors, typically returned by dictBehaviour
- device_brand Audience device brand, typically returned by dictDevice
- min_device_os Min OS version of audience used, blank for unrestricted, typically returned by dictOS
- max_device_os Max OS version of audience used, blank for unrestricted, typically returned by dictOS
- net_type Use netType to generate otherwise, 0: All, 1: Wifi, 2: 2G/3G/4G
- dsp_url Bid request url of DSP when this pre-targeting condition meets.

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth, listAudiences, createAudience, helpers

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# list audiences
audiences <- listAudiences()
```

```
# update audience name
update <- updateAudience(audience.id = audiences$id[1], body = list(
               audience_template_name = "New Audienced Name",
               audience_template_desc = "New Description"))

# show updated campaign
showAudience(audience.id = audiences$id[1])

## End(Not run)
```

`updateCampaign` *Update a campaign*

Description

Update campaign settings, see details and example.

Usage

```
updateCampaign(campaign.id, body)
```

Arguments

`campaign.id`	id of campaign to update
`body`	Body of request that includes campaigns settings. See examples and details.

Details

Valid values for `body`:

- `switch` Ad switch, use `switchIt` otherwise, `1`: On, `0`: Off.
- `name` Campaign name should be unique under the same user account.
- `budget_type` can take either `daily` or `lifetime`, see `budgetType`.
- `budget_daily` Required when `budget_type` equals to `daily`
- `budget_lifetime` Required when `budget_type` equals to `lifetime`

Author(s)

John Coene <jcoenep@gmail.com>

See Also

`orionOAuth`, `listCampaigns`, `createCampaign`, `helpers`

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# list campaigns
camps <- listCampaign()

# update campaign name
update <- updateCampaign(campaign.id = camps$id[1], body = list(
                         switch = 0,
                         name = "New campaign Name"))

# show updated campaign
showCampaign(campaign.id = camps$id[1])

## End(Not run)
```

`updateObject` *Update any object*

Description

Update any object, see details and example.

Usage

```
updateObject(object, body, id)
```

Arguments

`object`	Object to be updated, see details for valid values.
`body`	Body of request that includes campaigns settings. See examples.
`id`	id of object to update

Details

Valid values for `object`:

- `audience` see `updateAudience` for details.
- `campaign` see `updateCampaign` for details.
- `adset` see `updateAdset` for details.
- `ad` see `updateAd` for details.

`upodateObject` can essentially replace any other `update` family functions. See examples.

Author(s)

John Coene <jcoenep@gmail.com>

See Also

orionOAuth, updateAudience, updateCampaign, updateAdset, updateAd and helpers

Examples

```
## Not run:
# authenticate
orionOAuth(client.id = 0000,
           client.secret = "0x00000000x00x0x000xxx0000x0xx0")

# list current adsets
adsets <- listAdsets()

# new settings
settings <- list(switch = switchIt("on"), name = "New Adset Name")

# update adset
update <- updateObject(object = "adset", adset.id = adsets$id[1],
                       body = settings)

# equivalent to
# update <- updateAdset(adset.id = adsets$id[1], body = settings)

# fetch updated list
adsets <- listAdsets()

## End(Not run)
```

Index