Package 'RSauceLabs'

September 27, 2016

Type Package

Title R Wrapper for 'SauceLabs' REST API

Version 0.1.6

Description Retrieve, update, delete job information from <https://saucelabs.com/>. Poll the 'SauceLabs' services
current status and access supported platforms. Send and retrieve files from 'SauceLabs' and manage tunnels associated
with 'SauceConnect'.

Depends R (>= 3.0.0),httr,jsonlite,xml2,whisker,data.table

Suggests covr, testthat

Encoding UTF-8

License GPL-3

RoxygenNote 5.0.1

URL http://johndharrison.github.io/RSauceLabs/

URLNote https://github.com/johndharrison/RSauceLabs

BugReports https://github.com/johndharrison/RSauceLabs/issues

NeedsCompilation no

Author John Harrison [aut, cre]

Maintainer John Harrison <johndharrison0@gmail.com>

Repository CRAN

Date/Publication 2016-09-27 00:44:35

R topics documented:

`account` *Create an account object*

Description

Creates a wrapper around the httr `authenticate` function. Adds an "account" class.

Usage

```
account(user = Sys.getenv("SLUSER"), password = Sys.getenv("SLPASS"))
```

Arguments

`user`	The SauceLabs user. By default an environmental variable "SLUSER" is looked for.
`password`	The SauceLabs password for user. By default an environmental variable "SLPASS" is looked for.

Value

returns an object of class "account".

Examples

```
## Not run:
myAcc <- account()

## End(Not run)
```

---

changeAccessKey *Change access key*

Description

Change access key of an account

Usage

```
changeAccessKey(account, username = Sys.getenv("SLUSER"), ...)
```

Arguments

account	An object of class "account". An account object see account.
username	SauceLabs username
...	Additonal function arguments - Currently unused.

See Also

Other accountMethods: createUser, getListOfSiblingAccounts, getListOfSubAccounts, getSubAccountInformatio
getUserConcurrency, getUser

Examples

```
## Not run:
myAcc <- account()
appData <- getUser(myAcc)

createUser(myAcc, newUsername = "superstartester", password = "johndoe"
           , name = "John", email = "superstartester@example.com")
# $errors
# [1] "Subaccount capacity exhausted."
uC <- getUserConcurrency(myAcc)
#> rbindlist(uC$concurrency$self[c("allowed", "current")], fill = TRUE)
#manual mac overall real_device
#1:      5   5       5           0
#2:      0   0       0          NA
users <- getListOfSubAccounts(myAcc)
#> users$users_total
#[1] 1
siblings <- getListOfSiblingAccounts(myAcc)
#> getListOfSiblingAccounts(myAcc)
```

```
#list()
subAcc <- getSubAccountInformation(myAcc)

# change accesskey for a user
# changeAccessKey(myAcc, "rsaucelabs")

## End(Not run)
```

---

createUser *Create a sub account*

Description

Create a sub account

Usage

```
createUser(account, username = Sys.getenv("SLUSER"), newUsername, password,
  name, email, ...)
```

Arguments

account	An object of class "account". An account object see account.
username	SauceLabs username
newUsername	The username of the new user you wish to create
password	The password for the new user you wish to create
name	The name of the new user you wish to create
email	The email of the new user you wish to create
...	Additonal function arguments - Currently unused.

See Also

Other accountMethods: changeAccessKey, getListOfSiblingAccounts, getListOfSubAccounts, getSubAccountInformation, getUserConcurrency, getUser

Examples

```
## Not run:
myAcc <- account()
appData <- getUser(myAcc)

createUser(myAcc, newUsername = "superstartester", password = "johndoe"
           , name = "John", email = "superstartester@example.com")
# $errors
# [1] "Subaccount capacity exhausted."
uC <- getUserConcurrency(myAcc)
#> rbindlist(uC$concurrency$self[c("allowed", "current")], fill = TRUE)
```

```
#manual mac overall real_device
#1:      5   5       5           0
#2:      0   0       0          NA
users <- getListOfSubAccounts(myAcc)
#> users$users_total
#[1] 1
siblings <- getListOfSiblingAccounts(myAcc)
#> getListOfSiblingAccounts(myAcc)
#list()
subAcc <- getSubAccountInformation(myAcc)

# change accesskey for a user
# changeAccessKey(myAcc, "rsaucelabs")

## End(Not run)
```

---

deleteJob *Delete Job*

Description

Removes the job from the system with all the linked assets

Usage

```
deleteJob(account, username = Sys.getenv("SLUSER"), jobID, ...)
```

Arguments

account	An object of class "account". An account object see account.
username	SauceLabs username
jobID	Id of the job to delete
...	Additonal function arguments - Currently unused.

See Also

Other jobMethods: deleteJobAssets, getJobAssetFiles, getJobAssetNames, getJobs, stopJob, updateJob

Examples

```
## Not run:
myAcc <- account()
myJobs <- getJobs(myAcc)
#> myJobs$data[1,]
#id
#1: 4152e0a185f945bfa43e091eef1e7c30
myJobs <- getJobs(myAcc, getFullJobs = TRUE)
#> myJobs$data[1,.(id, browser)]
```

```
#id      browser
#1: 4152e0a185f945bfa43e091eef1e7c30 googlechrome
testId <- myJobs$data[1, id]

#> myJobs$data[1,.(build, passed)]
#build passed
#1:    24  FALSE

# update this job
updateJob(myAcc, jobID = testId, passed = TRUE, build = 20)
myJobs <- getJobs(myAcc, getFullJobs = TRUE)
#> myJobs$data[1,.(build, passed)]
#build passed
#1:    20   TRUE
# deleteJob(myAcc, jobID = testId)
stopJob(myAcc, jobID = testId)

jobAssets <- getJobAssetNames(myAcc, jobID = testId)
#> jobAssets[["selenium-log"]]
#[1] "selenium-server.log"

jobLog <- getJobAssetFiles(myAcc, jobID = testId)
# deleteJobAssets(myAcc, jobID = testId)

## End(Not run)
```

`deleteJobAssets` *Delete Job Assets*

Description

Delete all the assets captured during a test run. This includes the screencast recording, logs, and all screenshots.

Usage

```
deleteJobAssets(account, username = Sys.getenv("SLUSER"), jobID, ...)
```

Arguments

`account`	An object of class "account". An account object see `account`.
`username`	SauceLabs username
`jobID`	Id of the job to delete assests from
`...`	Additonal function arguments - Currently unused.

See Also

Other jobMethods: `deleteJob`, `getJobAssetFiles`, `getJobAssetNames`, `getJobs`, `stopJob`, `updateJob`

Examples

```
## Not run:
myAcc <- account()
myJobs <- getJobs(myAcc)
#> myJobs$data[1,]
#id
#1: 4152e0a185f945bfa43e091eef1e7c30
myJobs <- getJobs(myAcc, getFullJobs = TRUE)
#> myJobs$data[1,.(id, browser)]
#id      browser
#1: 4152e0a185f945bfa43e091eef1e7c30 googlechrome
testId <- myJobs$data[1, id]

#> myJobs$data[1,.(build, passed)]
#build passed
#1:    24  FALSE

# update this job
updateJob(myAcc, jobID = testId, passed = TRUE, build = 20)
myJobs <- getJobs(myAcc, getFullJobs = TRUE)
#> myJobs$data[1,.(build, passed)]
#build passed
#1:    20   TRUE
# deleteJob(myAcc, jobID = testId)
stopJob(myAcc, jobID = testId)

jobAssets <- getJobAssetNames(myAcc, jobID = testId)
#> jobAssets[["selenium-log"]]
#[1] "selenium-server.log"

jobLog <- getJobAssetFiles(myAcc, jobID = testId)
# deleteJobAssets(myAcc, jobID = testId)

## End(Not run)
```

`deleteTunnel` *Delete Tunnel*

Description

Shuts down a tunnel given its ID

Usage

```
deleteTunnel(account, username = Sys.getenv("SLUSER"), tunnelID, ...)
```

Arguments

account	An object of class "account". An account object see account.
username	SauceLabs username
tunnelID	id of the tunnel to shutdown
...	Additonal function arguments - Currently unused.

See Also

Other tunnelMethods: getTunnels, getTunnel

Examples

```
## Not run:
myAcc <- account()
myTunnels <- getTunnels(myAcc)
#> myTunnels
#list()
# A tunnel needs to be started with sauceConnect
# ./sc-4.3.16-linux/bin/sc -u seleniumPipes -k ##########-####-####-####-#######
# we start one
myTunnels <- getTunnels(myAcc)
#> myTunnels
#[[1]]
#[1] "cbfb1981c9dd45d1a1ecb9dc47de5ba4"
tunnelInfo <- getTunnel(myAcc, tunnelID = myTunnels[[1]])
#> tunnelInfo[c("status", "host", "owner", "id")]
#$status
#[1] "running"
#
#$host
#[1] "maki81013.miso.saucelabs.com"
#
#$owner
#[1] "seleniumPipes"
#
#$id
#[1] "cbfb1981c9dd45d1a1ecb9dc47de5ba4"
res <- deleteTunnel(myAcc, tunnelID = myTunnels[[1]])
#> res
#$jobs_running
#[1] 0
#
#$result
#[1] TRUE
#
#$id
#[1] "cbfb1981c9dd45d1a1ecb9dc47de5ba4"

#> getTunnels(myAcc)
#list()
```

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

`getAppiumEolDates` *Get Appium EOL dates*

Description

Get a list of Appium end-of-life dates. Dates are displayed in Unix time.

Usage

```
getAppiumEolDates(account, ...)
```

Arguments

`account`	An object of class "account". An account object see `account`.
`...`	Additonal function arguments - Currently unused.

See Also

Other infoMethods: `getSauceLabsStatus`, `getSupportedPlatforms`

Examples

```
## Not run:
myAcc <- account()
getSauceLabsStatus(myAcc)
#$wait_time
#[1] 1580.536
#
#$service_operational
#[1] TRUE
#
#$status_message
#[1] "Basic service status checks passed."
supportedPlatforms <- getSupportedPlatforms(myAcc)
supportedPlatforms[os == "Linux" & api_name == "chrome" & short_version > 44
                   , .(api_name, long_version)]
#api_name  long_version
#1:   chrome 45.0.2454.85.
#2:   chrome  46.0.2490.71
#3:   chrome  47.0.2526.73
#4:   chrome  48.0.2564.97
getAppiumEolDates(myAcc)
#$`1.4.0`
#[1] "2016-04-09 PDT"
#
#$`1.4.3`
```

```
#[1] "2016-04-09 PDT"
#....

## End(Not run)
```

`getJobAssetFiles` *Get Job Asset Files*

Description

Download job assets. After a job completes, all assets created during the job are available via this API. These include the screencast recording, logs, and screenshots taken on crucial steps. The job assests will be deleted from the test page after 30 days. Thus, after 30 days all your test commands, logs, screenshots and the screencast recording will be gone. This is the reason why we strongly recommend to download your job assets if this is an information that you must keep in your records.

Usage

```
getJobAssetFiles(account, username = Sys.getenv("SLUSER"), jobID,
  fileName = "selenium-server.log", ...)
```

Arguments

`account`	An object of class "account". An account object see `account`.
`username`	SauceLabs username
`jobID`	Id of the job to get assets from
`fileName`	Accepted Values for fileName "selenium-server.log" "video.flv" "XXXXscreen-shot.png" (where XXXX is a number between 0000 and 9999) "final_screenshot.png"
`...`	Additonal function arguments - Currently unused.

See Also

Other jobMethods: `deleteJobAssets`, `deleteJob`, `getJobAssetNames`, `getJobs`, `stopJob`, `updateJob`

Examples

```
## Not run:
myAcc <- account()
myJobs <- getJobs(myAcc)
#> myJobs$data[1,]
#id
#1: 4152e0a185f945bfa43e091eef1e7c30
myJobs <- getJobs(myAcc, getFullJobs = TRUE)
#> myJobs$data[1,.(id, browser)]
#id      browser
#1: 4152e0a185f945bfa43e091eef1e7c30 googlechrome
testId <- myJobs$data[1, id]
```

```
#> myJobs$data[1,.(build, passed)]
#build passed
#1:    24  FALSE

# update this job
updateJob(myAcc, jobID = testId, passed = TRUE, build = 20)
myJobs <- getJobs(myAcc, getFullJobs = TRUE)
#> myJobs$data[1,.(build, passed)]
#build passed
#1:    20   TRUE
# deleteJob(myAcc, jobID = testId)
stopJob(myAcc, jobID = testId)

jobAssets <- getJobAssetNames(myAcc, jobID = testId)
#> jobAssets[["selenium-log"]]
#[1] "selenium-server.log"

jobLog <- getJobAssetFiles(myAcc, jobID = testId)
# deleteJobAssets(myAcc, jobID = testId)

## End(Not run)
```

`getJobAssetNames` *Get Job Asset Names*

Description

Get details about the static assets collected for a specific job

Usage

```
getJobAssetNames(account, username = Sys.getenv("SLUSER"), jobID, ...)
```

Arguments

`account`	An object of class "account". An account object see `account`.
`username`	SauceLabs username
`jobID`	Id of the job to stop
`...`	Additonal function arguments - Currently unused.

See Also

Other jobMethods: `deleteJobAssets`, `deleteJob`, `getJobAssetFiles`, `getJobs`, `stopJob`, `updateJob`

Examples

```
## Not run:
myAcc <- account()
myJobs <- getJobs(myAcc)
#> myJobs$data[1,]
#id
#1: 4152e0a185f945bfa43e091eef1e7c30
myJobs <- getJobs(myAcc, getFullJobs = TRUE)
#> myJobs$data[1,.(id, browser)]
#id      browser
#1: 4152e0a185f945bfa43e091eef1e7c30 googlechrome
testId <- myJobs$data[1, id]

#> myJobs$data[1,.(build, passed)]
#build passed
#1:    24  FALSE

# update this job
updateJob(myAcc, jobID = testId, passed = TRUE, build = 20)
myJobs <- getJobs(myAcc, getFullJobs = TRUE)
#> myJobs$data[1,.(build, passed)]
#build passed
#1:    20   TRUE
# deleteJob(myAcc, jobID = testId)
stopJob(myAcc, jobID = testId)

jobAssets <- getJobAssetNames(myAcc, jobID = testId)
#> jobAssets[["selenium-log"]]
#[1] "selenium-server.log"

jobLog <- getJobAssetFiles(myAcc, jobID = testId)
# deleteJobAssets(myAcc, jobID = testId)

## End(Not run)
```

`getJobs` *Get Jobs*

Description

List recent jobs belonging to a specific user

Usage

```
getJobs(account, username = Sys.getenv("SLUSER"), limit = 100L,
  getFullJobs = FALSE, skipJobs = 0L, to = NULL, from = NULL, ...)
```

Arguments

account	An object of class "account". An account object see account.
username	SauceLabs username
limit	Specifies the number of jobs to return. Default is 100 and max is 500.
getFullJobs	Get full job information, rather than just IDs. Default is FALSE.
skipJobs	Skips the specified number of jobs. Default is 0.
to	Get jobs until the specified time (POSIXct)
from	Get jobs since the specified time (POSIXct)
...	Additonal function arguments - Currently unused.

Value

returns a named list. "data" is the job data minus the tags and custom-data. tagsAndCD are a list of tags and custom-data for each job. If getFullJobs = FALSE then data only contains the job ids and tagsAndCD contains empty lists for each job.

See Also

Other jobMethods: deleteJobAssets, deleteJob, getJobAssetFiles, getJobAssetNames, stopJob, updateJob

Examples

```
## Not run:
myAcc <- account()
myJobs <- getJobs(myAcc)
#> myJobs$data[1,]
#id
#1: 4152e0a185f945bfa43e091eef1e7c30
myJobs <- getJobs(myAcc, getFullJobs = TRUE)
#> myJobs$data[1,.(id, browser)]
#id      browser
#1: 4152e0a185f945bfa43e091eef1e7c30 googlechrome
testId <- myJobs$data[1, id]

#> myJobs$data[1,.(build, passed)]
#build passed
#1:    24  FALSE

# update this job
updateJob(myAcc, jobID = testId, passed = TRUE, build = 20)
myJobs <- getJobs(myAcc, getFullJobs = TRUE)
#> myJobs$data[1,.(build, passed)]
#build passed
#1:    20   TRUE
# deleteJob(myAcc, jobID = testId)
stopJob(myAcc, jobID = testId)

jobAssets <- getJobAssetNames(myAcc, jobID = testId)
```

```
#> jobAssets[["selenium-log"]]
#[1] "selenium-server.log"

jobLog <- getJobAssetFiles(myAcc, jobID = testId)
# deleteJobAssets(myAcc, jobID = testId)

## End(Not run)
```

`getJsUnitTestStatus` *Get JS Unit Test Status*

Description

Get the status of your JS unit tests

Usage

```
getJsUnitTestStatus(account, username = Sys.getenv("SLUSER"), js_tests, ...)
```

Arguments

`account`	An object of class "account". An account object see `account`.
`username`	SauceLabs username
`js_tests`	a vector of job ids.
`...`	Additonal function arguments - Currently unused.

See Also

Other jsUnitTestMethods: `startJsUnitTests`

Examples

```
## Not run:
# use test example from
#https://wiki.saucelabs.com/display/DOCS/JavaScript+Unit+Testing+Methods
platforms <- list(c("Windows 7", "firefox", "27"),
                  c("Linux", "googlechrome", "")
                  )
appUrl <- "https://saucelabs.com/test_helpers/front_tests/index.html"
framework <- "jasmine"
myAcc <- account()
myTest <- startJsUnitTests(myAcc, platforms = platforms, url = appUrl, framework = framework)

#> unlist(myTest, use.names = FALSE)
#[1] "bc8b9ef6e6184ed8a7e5270344115999" "bf43cef30bca429eaa2ed08da09dbdce"
testIds <- unlist(myTest, use.names = FALSE)
testRes <- getJsUnitTestStatus(myAcc,js_tests = testIds)
```

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

`getListOfSiblingAccounts` *Get a list of sibling accounts*

Description

Get a list of sibling accounts associated with provided account

Usage

```
getListOfSiblingAccounts(account, username = Sys.getenv("SLUSER"),
  page = NULL, per_page = 50L, ...)
```

Arguments

account	An object of class "account". An account object see account.
username	SauceLabs username
page	optional defaults to NULL
per_page	results per page (max 50L). Defaults to 50L
...	Additonal function arguments - Currently unused.

See Also

Other accountMethods: changeAccessKey, createUser, getListOfSubAccounts, getSubAccountInformation, getUserConcurrency, getUser

Examples

```
## Not run:
myAcc <- account()
appData <- getUser(myAcc)

createUser(myAcc, newUsername = "superstartester", password = "johndoe"
           , name = "John", email = "superstartester@example.com")
# $errors
# [1] "Subaccount capacity exhausted."
uC <- getUserConcurrency(myAcc)
#> rbindlist(uC$concurrency$self[c("allowed", "current")], fill = TRUE)
#manual mac overall real_device
#1:       5   5       5           0
#2:       0   0       0          NA
users <- getListOfSubAccounts(myAcc)
#> users$users_total
#[1] 1
siblings <- getListOfSiblingAccounts(myAcc)
```

```
#> getListOfSiblingAccounts(myAcc)
#list()
subAcc <- getSubAccountInformation(myAcc)

# change accesskey for a user
# changeAccessKey(myAcc, "rsaucelabs")

## End(Not run)
```

`getListOfSubAccounts` *Get a list of sub accounts*

Description

Get a list of sub accounts associated with a parent account

Usage

```
getListOfSubAccounts(account, username = Sys.getenv("SLUSER"), from = NULL,
  limit = 100L, ...)
```

Arguments

`account`	An object of class "account". An account object see `account`.
`username`	SauceLabs username
`from`	Get user from this user number. Defaults to NULL
`limit`	The limit on users returned. Defaults to 50L (50L is the max).
`...`	Additonal function arguments - Currently unused.

See Also

Other accountMethods: `changeAccessKey`, `createUser`, `getListOfSiblingAccounts`, `getSubAccountInformation`, `getUserConcurrency`, `getUser`

Examples

```
## Not run:
myAcc <- account()
appData <- getUser(myAcc)

createUser(myAcc, newUsername = "superstartester", password = "johndoe"
           , name = "John", email = "superstartester@example.com")
# $errors
# [1] "Subaccount capacity exhausted."
uC <- getUserConcurrency(myAcc)
#> rbindlist(uC$concurrency$self[c("allowed", "current")], fill = TRUE)
#manual mac overall real_device
#1:      5   5       5           0
```

```
#2:      0   0       0          NA
users <- getListOfSubAccounts(myAcc)
#> users$users_total
#[1] 1
siblings <- getListOfSiblingAccounts(myAcc)
#> getListOfSiblingAccounts(myAcc)
#list()
subAcc <- getSubAccountInformation(myAcc)

# change accesskey for a user
# changeAccessKey(myAcc, "rsaucelabs")

## End(Not run)
```

`getRealTimeJobActivity` *Get Real-Time Job Activity*

Description

Get information about concurrency, minutes and jobs used by the user over a specific duration (default 90 days). Concurrency is separated in mean and peak concurrency.

Usage

```
getRealTimeJobActivity(account, username = Sys.getenv("SLUSER"), ...)
```

Arguments

`account`	An object of class "account". An account object see `account`.
`username`	SauceLabs username
`...`	Additonal function arguments - Currently unused.

See Also

Other actUsageMethods: `getUserAccountUsage`, `getUserActivity`

Examples

```
## Not run:
myAcc <- account()
jobActivity <- getRealTimeJobActivity(myAcc)
#> jobActivity$concurrency$self$allowed
#$manual
#[1] 5
#
#$mac
#[1] 5
#
```

```
#$overall
#[1] 5
#
#$real_device
#[1] 0
userActivity <- getUserActivity(myAcc)
#> userActivity$subaccounts$rsaucelabs
#$`in progress`
#[1] 0
#
#$all
#[1] 0
#
#$queued
#[1] 0
userAccUsage <- getUserAccountUsage(myAcc)
#> userAccUsage
#user_name      date no_of_jobs vm_minutes
#1: seleniumPipes 2016-8-12          2        239
#2: seleniumPipes 2016-8-13         65       6399
#3: seleniumPipes 2016-8-15         36       7235
#4: seleniumPipes 2016-8-16          7       1101

## End(Not run)
```

`getSauceLabsStatus` *Get Sauce Labs Status*

Description

Get the current status of Sauce Labs services

Usage

```
getSauceLabsStatus(account, ...)
```

Arguments

`account`	An object of class "account". An account object see `account`.
`...`	Additonal function arguments - Currently unused.

See Also

Other infoMethods: `getAppiumEolDates`, `getSupportedPlatforms`

Examples

```
## Not run:
myAcc <- account()
getSauceLabsStatus(myAcc)
#$wait_time
#[1] 1580.536
#
#$service_operational
#[1] TRUE
#
#$status_message
#[1] "Basic service status checks passed."
supportedPlatforms <- getSupportedPlatforms(myAcc)
supportedPlatforms[os == "Linux" & api_name == "chrome" & short_version > 44
                   , .(api_name, long_version)]
#api_name  long_version
#1:   chrome 45.0.2454.85.
#2:   chrome  46.0.2490.71
#3:   chrome  47.0.2526.73
#4:   chrome  48.0.2564.97
getAppiumEolDates(myAcc)
#$`1.4.0`
#[1] "2016-04-09 PDT"
#
#$`1.4.3`
#[1] "2016-04-09 PDT"
#....

## End(Not run)
```

`getStoredFiles` *Get Stored Files*

Description

Check which files are in your temporary storage

Usage

```
getStoredFiles(account, username = Sys.getenv("SLUSER"), ...)
```

Arguments

`account`	An object of class "account". An account object see `account`.
`username`	SauceLabs username
`...`	Additonal function arguments - Currently unused.

See Also

Other tempStorageMethods: uploadFile

Examples

```
## Not run:
myAcc <- account()
# create a temporary file
myTempFile <- file.path(tempdir(), "notsecret.html")
write("SUPER SECRET STUFF", myTempFile)
# check stored files
myStoredFiles <- getStoredFiles(myAcc)

# upload new file
res <- uploadFile(myAcc, file = myTempFile)
#> res
#$username
#[1] "seleniumPipes"
#
#$size
#[1] 19
#
#$md5
#[1] "e459fe3803b78d64cc5c2998804909a9"
#
#$filename
#[1] "notsecret.html"

#> digest::digest(file = myTempFile, algo = "md5")
#[1] "e459fe3803b78d64cc5c2998804909a9"

myStoredFiles <- getStoredFiles(myAcc)

#> rbindlist(myStoredFiles$files)
#size      mtime           name                               md5
#1:   19 1472401537 notsecret.html e459fe3803b78d64cc5c2998804909a9
#2:   14 1472350499      testDoc.R adfc8afc373f0b3fd6f93c3891bdd11b


## End(Not run)
```

---

getSubAccountInformation

Get information about a sub account

Description

Get information about a sub account

Usage

```
getSubAccountInformation(account, username = Sys.getenv("SLUSER"), ...)
```

Arguments

account	An object of class "account". An account object see account.
username	SauceLabs username
...	Additonal function arguments - Currently unused.

See Also

Other accountMethods: changeAccessKey, createUser, getListOfSiblingAccounts, getListOfSubAccounts, getUserConcurrency, getUser

Examples

```
## Not run:
myAcc <- account()
appData <- getUser(myAcc)

createUser(myAcc, newUsername = "superstartester", password = "johndoe"
           , name = "John", email = "superstartester@example.com")
# $errors
# [1] "Subaccount capacity exhausted."
uC <- getUserConcurrency(myAcc)
#> rbindlist(uC$concurrency$self[c("allowed", "current")], fill = TRUE)
#manual mac overall real_device
#1:      5   5       5           0
#2:      0   0       0          NA
users <- getListOfSubAccounts(myAcc)
#> users$users_total
#[1] 1
siblings <- getListOfSiblingAccounts(myAcc)
#> getListOfSiblingAccounts(myAcc)
#list()
subAcc <- getSubAccountInformation(myAcc)

# change accesskey for a user
# changeAccessKey(myAcc, "rsaucelabs")

## End(Not run)
```

---

getSupportedPlatforms *Get Supported Platforms*

Description

Get a list of objects describing all the OS and browser platforms currently supported on Sauce Labs. Choose the automation API you need, bearing in mind that WebDriver and Selenium RC are each compatible with a different set of platforms.

Usage

```
getSupportedPlatforms(account, autoAPI = "webdriver", ...)
```

Arguments

account	An object of class "account". An account object see account.
autoAPI	Accepted Values for autoAPI "all", "appium", "webdriver". Defaults to "webdriver"
...	Additonal function arguments - Currently unused.

See Also

Other infoMethods: getAppiumEolDates, getSauceLabsStatus

Examples

```
## Not run:
myAcc <- account()
getSauceLabsStatus(myAcc)
#$wait_time
#[1] 1580.536
#
#$service_operational
#[1] TRUE
#
#$status_message
#[1] "Basic service status checks passed."
supportedPlatforms <- getSupportedPlatforms(myAcc)
supportedPlatforms[os == "Linux" & api_name == "chrome" & short_version > 44
                   , .(api_name, long_version)]
#api_name  long_version
#1:   chrome 45.0.2454.85.
#2:   chrome  46.0.2490.71
#3:   chrome  47.0.2526.73
#4:   chrome  48.0.2564.97
getAppiumEolDates(myAcc)
#$`1.4.0`
#[1] "2016-04-09 PDT"
#
#$`1.4.3`
#[1] "2016-04-09 PDT"
#....

## End(Not run)
```

getTunnel *getTunnel*

Description

Get information for a tunnel given its ID

Usage

```
getTunnel(account, username = Sys.getenv("SLUSER"), tunnelID, ...)
```

Arguments

account	An object of class "account". An account object see account.
username	SauceLabs username
tunnelID	id of the tunnel to get more information on
...	Additonal function arguments - Currently unused.

See Also

Other tunnelMethods: deleteTunnel, getTunnels

Examples

```
## Not run:
myAcc <- account()
myTunnels <- getTunnels(myAcc)
#> myTunnels
#list()
# A tunnel needs to be started with sauceConnect
# ./sc-4.3.16-linux/bin/sc -u seleniumPipes -k ##########-####-####-####-#######
# we start one
myTunnels <- getTunnels(myAcc)
#> myTunnels
#[[1]]
#[1] "cbfb1981c9dd45d1a1ecb9dc47de5ba4"
tunnelInfo <- getTunnel(myAcc, tunnelID = myTunnels[[1]])
#> tunnelInfo[c("status", "host", "owner", "id")]
#$status
#[1] "running"
#
#$host
#[1] "maki81013.miso.saucelabs.com"
#
#$owner
#[1] "seleniumPipes"
#
#$id
```

```
#[1] "cbfb1981c9dd45d1a1ecb9dc47de5ba4"
res <- deleteTunnel(myAcc, tunnelID = myTunnels[[1]])
#> res
#$jobs_running
#[1] 0
#
#$result
#[1] TRUE
#
#$id
#[1] "cbfb1981c9dd45d1a1ecb9dc47de5ba4"

#> getTunnels(myAcc)
#list()


## End(Not run)
```

`getTunnels` *Get Tunnels*

Description

Retrieves all running tunnels for a specific user

Usage

```
getTunnels(account, username = Sys.getenv("SLUSER"), ...)
```

Arguments

`account`	An object of class "account". An account object see `account`.
`username`	SauceLabs username
`...`	Additonal function arguments - Currently unused.

See Also

Other tunnelMethods: `deleteTunnel`, `getTunnel`

Examples

```
## Not run:
myAcc <- account()
myTunnels <- getTunnels(myAcc)
#> myTunnels
#list()
# A tunnel needs to be started with sauceConnect
# ./sc-4.3.16-linux/bin/sc -u seleniumPipes -k ##########-####-####-####-#######
# we start one
```

```
myTunnels <- getTunnels(myAcc)
#> myTunnels
#[[1]]
#[1] "cbfb1981c9dd45d1a1ecb9dc47de5ba4"
tunnelInfo <- getTunnel(myAcc, tunnelID = myTunnels[[1]])
#> tunnelInfo[c("status", "host", "owner", "id")]
#$status
#[1] "running"
#
#$host
#[1] "maki81013.miso.saucelabs.com"
#
#$owner
#[1] "seleniumPipes"
#
#$id
#[1] "cbfb1981c9dd45d1a1ecb9dc47de5ba4"
res <- deleteTunnel(myAcc, tunnelID = myTunnels[[1]])
#> res
#$jobs_running
#[1] 0
#
#$result
#[1] TRUE
#
#$id
#[1] "cbfb1981c9dd45d1a1ecb9dc47de5ba4"

#> getTunnels(myAcc)
#list()


## End(Not run)
```

`getUser` *Access basic account information*

Description

Access basic account information

Usage

```
getUser(account, username = Sys.getenv("SLUSER"), ...)
```

Arguments

`account`	An object of class "account". An account object see `account`.
`username`	SauceLabs username
`...`	Additonal function arguments - Currently unused.

See Also

Other accountMethods: changeAccessKey, createUser, getListOfSiblingAccounts, getListOfSubAccounts, getSubAccountInformation, getUserConcurrency

Examples

```
## Not run:
myAcc <- account()
appData <- getUser(myAcc)

createUser(myAcc, newUsername = "superstartester", password = "johndoe"
           , name = "John", email = "superstartester@example.com")
# $errors
# [1] "Subaccount capacity exhausted."
uC <- getUserConcurrency(myAcc)
#> rbindlist(uC$concurrency$self[c("allowed", "current")], fill = TRUE)
#manual mac overall real_device
#1:      5   5       5           0
#2:      0   0       0          NA
users <- getListOfSubAccounts(myAcc)
#> users$users_total
#[1] 1
siblings <- getListOfSiblingAccounts(myAcc)
#> getListOfSiblingAccounts(myAcc)
#list()
subAcc <- getSubAccountInformation(myAcc)

# change accesskey for a user
# changeAccessKey(myAcc, "rsaucelabs")

## End(Not run)
```

---

getUserAccountUsage *Get User Account Usage*

Description

Access historical account usage data

Usage

```
getUserAccountUsage(account, username = Sys.getenv("SLUSER"), ...)
```

Arguments

account	An object of class "account". An account object see account.
username	SauceLabs username
...	Additonal function arguments - Currently unused.

Value

The result is a breakdown summarizing the total number of jobs and VM time used, in seconds, by day.

See Also

Other actUsageMethods: getRealTimeJobActivity, getUserActivity

Examples

```
## Not run:
myAcc <- account()
jobActivity <- getRealTimeJobActivity(myAcc)
#> jobActivity$concurrency$self$allowed
#$manual
#[1] 5
#
#$mac
#[1] 5
#
#$overall
#[1] 5
#
#$real_device
#[1] 0
userActivity <- getUserActivity(myAcc)
#> userActivity$subaccounts$rsaucelabs
#$`in progress`
#[1] 0
#
#$all
#[1] 0
#
#$queued
#[1] 0
userAccUsage <- getUserAccountUsage(myAcc)
#> userAccUsage
#user_name      date no_of_jobs vm_minutes
#1: seleniumPipes 2016-8-12          2        239
#2: seleniumPipes 2016-8-13         65       6399
#3: seleniumPipes 2016-8-15         36       7235
#4: seleniumPipes 2016-8-16          7       1101

## End(Not run)
```

`getUserActivity` *Get User Activity*

Description

Get currently running job counts broken down by account and job status.

Usage

```
getUserActivity(account, username = Sys.getenv("SLUSER"), ...)
```

Arguments

account	An object of class "account". An account object see account.
username	SauceLabs username
...	Additonal function arguments - Currently unused.

See Also

Other actUsageMethods: getRealTimeJobActivity, getUserAccountUsage

Examples

```
## Not run:
myAcc <- account()
jobActivity <- getRealTimeJobActivity(myAcc)
#> jobActivity$concurrency$self$allowed
#$manual
#[1] 5
#
#$mac
#[1] 5
#
#$overall
#[1] 5
#
#$real_device
#[1] 0
userActivity <- getUserActivity(myAcc)
#> userActivity$subaccounts$rsaucelabs
#$`in progress`
#[1] 0
#
#$all
#[1] 0
#
#$queued
#[1] 0
userAccUsage <- getUserAccountUsage(myAcc)
#> userAccUsage
#user_name      date no_of_jobs vm_minutes
#1: seleniumPipes 2016-8-12          2        239
#2: seleniumPipes 2016-8-13         65       6399
#3: seleniumPipes 2016-8-15         36       7235
#4: seleniumPipes 2016-8-16          7       1101
```

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

---

getUserConcurrency *Check account concurrency limits*

Description

Check account concurrency limits

Usage

```
getUserConcurrency(account, username = Sys.getenv("SLUSER"), ...)
```

Arguments

account	An object of class "account". An account object see account.
username	SauceLabs username
...	Additonal function arguments - Currently unused.

See Also

Other accountMethods: changeAccessKey, createUser, getListOfSiblingAccounts, getListOfSubAccounts, getSubAccountInformation, getUser

Examples

```
## Not run:
myAcc <- account()
appData <- getUser(myAcc)

createUser(myAcc, newUsername = "superstartester", password = "johndoe"
           , name = "John", email = "superstartester@example.com")
# $errors
# [1] "Subaccount capacity exhausted."
uC <- getUserConcurrency(myAcc)
#> rbindlist(uC$concurrency$self[c("allowed", "current")], fill = TRUE)
#manual mac overall real_device
#1:      5   5       5           0
#2:      0   0       0          NA
users <- getListOfSubAccounts(myAcc)
#> users$users_total
#[1] 1
siblings <- getListOfSiblingAccounts(myAcc)
#> getListOfSiblingAccounts(myAcc)
#list()
subAcc <- getSubAccountInformation(myAcc)

# change accesskey for a user
```

```
# changeAccessKey(myAcc, "rsaucelabs")

## End(Not run)
```

---

queryAPI *Send a query to SauceLabs.*

Description

queryAPI A function to send a query to SauceLabs. Intended for seleniumPipes internal use mainly.

Usage

```
queryAPI(verb = GET, account, url, source, ...)
```

Arguments

verb	The http method to use. See VERB
account	An object of class "account". An account object see account.
url	The url of the remote server endpoint.
source	The name of the RSauceLabs function that called queryDriver.
...	Additonal function arguments - Currently unused.

Value

The contents of the response from the remote server. See content for details.

Examples

```
## Not run:
# function intended for internal use

## End(Not run)
```

---

startJsUnitTests *Start JS Unit Tests*

Description

Start your JavaScript unit tests on as many browsers as you like with a single request

Usage

```
startJsUnitTests(account, username = Sys.getenv("SLUSER"), platforms, url,
  framework, ...)
```

Arguments

account	An object of class "account". An account object see account.
username	SauceLabs username
platforms	A list of platforms (see example)
url	should point to the page that hosts your tests
framework	can be "qunit", "jasmine", "YUI Test", "mocha", or "custom"
...	Additonal function arguments - Currently unused.

See Also

Other jsUnitTestMethods: getJsUnitTestStatus

Examples

```
## Not run:
# use test example from
#https://wiki.saucelabs.com/display/DOCS/JavaScript+Unit+Testing+Methods
platforms <- list(c("Windows 7", "firefox", "27"),
                  c("Linux", "googlechrome", "")
                  )
appUrl <- "https://saucelabs.com/test_helpers/front_tests/index.html"
framework <- "jasmine"
myAcc <- account()
myTest <- startJsUnitTests(myAcc, platforms = platforms, url = appUrl, framework = framework)

#> unlist(myTest, use.names = FALSE)
#[1] "bc8b9ef6e6184ed8a7e5270344115999" "bf43cef30bca429eaa2ed08da09dbdce"
testIds <- unlist(myTest, use.names = FALSE)
testRes <- getJsUnitTestStatus(myAcc,js_tests = testIds)


## End(Not run)
```

---

stopJob *Stop Job*

Description

Terminates a running job

Usage

```
stopJob(account, username = Sys.getenv("SLUSER"), jobID, ...)
```

Arguments

account	An object of class "account". An account object see account.
username	SauceLabs username
jobID	Id of the job to stop
...	Additonal function arguments - Currently unused.

See Also

Other jobMethods: deleteJobAssets, deleteJob, getJobAssetFiles, getJobAssetNames, getJobs, updateJob

Examples

```
## Not run:
myAcc <- account()
myJobs <- getJobs(myAcc)
#> myJobs$data[1,]
#id
#1: 4152e0a185f945bfa43e091eef1e7c30
myJobs <- getJobs(myAcc, getFullJobs = TRUE)
#> myJobs$data[1,.(id, browser)]
#id      browser
#1: 4152e0a185f945bfa43e091eef1e7c30 googlechrome
testId <- myJobs$data[1, id]

#> myJobs$data[1,.(build, passed)]
#build passed
#1:    24  FALSE

# update this job
updateJob(myAcc, jobID = testId, passed = TRUE, build = 20)
myJobs <- getJobs(myAcc, getFullJobs = TRUE)
#> myJobs$data[1,.(build, passed)]
#build passed
#1:    20   TRUE
# deleteJob(myAcc, jobID = testId)
stopJob(myAcc, jobID = testId)

jobAssets <- getJobAssetNames(myAcc, jobID = testId)
#> jobAssets[["selenium-log"]]
#[1] "selenium-server.log"

jobLog <- getJobAssetFiles(myAcc, jobID = testId)
# deleteJobAssets(myAcc, jobID = testId)

## End(Not run)
```

updateJob *Update Job*

Description

Edit an existing job

Usage

```
updateJob(account, username = Sys.getenv("SLUSER"), jobID, name = NULL,
  tags = NULL, public = NULL, passed = NULL, build = NULL,
  custom_data = NULL, ...)
```

Arguments

account	An object of class "account". An account object see account.
username	SauceLabs username
jobID	Id of the job to edit
name	Change the job name
tags	Change the job tags
public	Set job visibility to "public", "public restricted", "share" (true), "team" (false) or "private"
passed	Set whether the job passed or not on the user end
build	The build number tested by this test
custom_data	A set of key-value pairs with any extra info that a user would like to add to the job. Note that the max data allowed is 64KB
...	Additonal function arguments - Currently unused.

See Also

Other jobMethods: deleteJobAssets, deleteJob, getJobAssetFiles, getJobAssetNames, getJobs, stopJob

Examples

```
## Not run:
myAcc <- account()
myJobs <- getJobs(myAcc)
#> myJobs$data[1,]
#id
#1: 4152e0a185f945bfa43e091eef1e7c30
myJobs <- getJobs(myAcc, getFullJobs = TRUE)
#> myJobs$data[1,.(id, browser)]
#id      browser
#1: 4152e0a185f945bfa43e091eef1e7c30 googlechrome
```

```
testId <- myJobs$data[1, id]

#> myJobs$data[1,.(build, passed)]
#build passed
#1:    24  FALSE

# update this job
updateJob(myAcc, jobID = testId, passed = TRUE, build = 20)
myJobs <- getJobs(myAcc, getFullJobs = TRUE)
#> myJobs$data[1,.(build, passed)]
#build passed
#1:    20   TRUE
# deleteJob(myAcc, jobID = testId)
stopJob(myAcc, jobID = testId)

jobAssets <- getJobAssetNames(myAcc, jobID = testId)
#> jobAssets[["selenium-log"]]
#[1] "selenium-server.log"

jobLog <- getJobAssetFiles(myAcc, jobID = testId)
# deleteJobAssets(myAcc, jobID = testId)

## End(Not run)
```

`uploadFile` *Upload File*

Description

Uploads a file to the temporary sauce storage. The storage will only retain the files for seven days.

Usage

```
uploadFile(account, username = Sys.getenv("SLUSER"), file,
  SLfileName = basename(file), ...)
```

Arguments

`account`	An object of class "account". An account object see `account`.
`username`	SauceLabs username
`file`	file to upload
`SLfileName`	name to give the file on SauceLabs. (Defaults to the current name of the file)
`...`	Additonal function arguments - Currently unused.

See Also

Other tempStorageMethods: `getStoredFiles`

Examples

```
## Not run:
myAcc <- account()
# create a temporary file
myTempFile <- file.path(tempdir(), "notsecret.html")
write("SUPER SECRET STUFF", myTempFile)
# check stored files
myStoredFiles <- getStoredFiles(myAcc)

# upload new file
res <- uploadFile(myAcc, file = myTempFile)
#> res
#$username
#[1] "seleniumPipes"
#
#$size
#[1] 19
#
#$md5
#[1] "e459fe3803b78d64cc5c2998804909a9"
#
#$filename
#[1] "notsecret.html"

#> digest::digest(file = myTempFile, algo = "md5")
#[1] "e459fe3803b78d64cc5c2998804909a9"

myStoredFiles <- getStoredFiles(myAcc)

#> rbindlist(myStoredFiles$files)
#size      mtime           name                                md5
#1:   19 1472401537 notsecret.html e459fe3803b78d64cc5c2998804909a9
#2:   14 1472350499      testDoc.R adfc8afc373f0b3fd6f93c3891bdd11b


## End(Not run)
```

Index