Package 'RForcecom'

July 19, 2016

Type Package

Version 1.1

Title Data Integration Feature for Force.com and Salesforce.com

Description Insert, update,
retrieve, delete and bulk operate datasets with a SaaS based CRM
Salesforce.com and a PaaS based application platform Force.com from R.

URL http://rforcecom.plavox.info/

BugReports https://github.com/hiratake55/RForcecom/issues

Depends R (>= 3.1.0)

Imports XML, httr, plyr, methods, RCurl, utils

Suggests testthat

LazyLoad yes

License Apache License 2.0

Date 2016-07-18

Collate 'RForcecom.R' 'rforcecom.abortBulkJob.R' 'rforcecom.api.R'
'rforcecom.closeBulkJob.R' 'rforcecom.getBulkQueryResult.R'
'rforcecom.getBatchDetails.R' 'rforcecom.checkBatchStatus.R'
'rforcecom.submitBulkQuery.R' 'rforcecom.createBulkJob.R'
'rforcecom.bulkQuery.R' 'rforcecom.create.R'
'rforcecom.createBulkBatch.R' 'rforcecom.debug.R'
'rforcecom.delete.R' 'rforcecom.getObjectDescription.R'
'rforcecom.getObjectList.R' 'rforcecom.getServerTimestamp.R'
'rforcecom.insertBulkAttachments.R' 'rforcecom.login.R'
'rforcecom.logout.R' 'rforcecom.query.R'
'rforcecom.queryMore.R' 'rforcecom.retrieve.R'
'rforcecom.search.R' 'rforcecom.update.R' 'rforcecom.upsert.R'
'rforcecom.utils.R' 'rforcecom.write.csv.R'

RoxygenNote 5.0.1

NeedsCompilation no

Author Takekatsu Hiramura [aut, cre],
Steven Mortimer [ctb],
Alexis Iglauer [ctb]

Maintainer Takekatsu Hiramura <`thira@plavox.info`>

Repository CRAN

Date/Publication 2016-07-19 08:15:08

R topics documented:

`RForcecom` *RForcecom*

Description

Data Integration Feature for Force.com and Salesforce.com

Details

Package:	RForcecom
Type:	Package
Version:	1.1
Date:	2016-07-01
License:	Apache License 2.0
LazyLoad:	yes

Author(s)

Takekatsu Hiramura <thira@plavox.info>

References

Force.com REST API Developer's Guide
https://developer.salesforce.com/docs/atlas.en-us.api_rest.meta/api_rest/

Web Services API Developer's Guide
https://developer.salesforce.com/docs/atlas.en-us.api.meta/api/

Bulk API Developer's Guide
https://developer.salesforce.com/docs/atlas.en-us.api_asynch.meta/api_asynch/

See Also

XML httr plyr

Examples

```
## Not run:
 # Sign in to the Force.com
 username <- "yourname@yourcompany.com"
 password <- "YourPasswordSECURITY_TOKEN"
 session <- rforcecom.login(username, password)

 # Execute a SOQL
 soqlQuery <- "SELECT Id, Name, Industry, AnnualRevenue FROM Account"
 rforcecom.query(session, soqlQuery)

 # Execute a SOSL
 queryString <- "United"
 rforcecom.search(session, queryString)

 # Create a record
 objectName <- "Account"
 fields <- c(Name="R Analytics Service Ltd", Phone="5555-5555-5555")
 rforcecom.create(session, objectName, fields)
```

```
# Retrieve record
objectName <- "Account"
fields <- c("name", "Industry", "AnnualRevenue")
rforcecom.retrieve(session, objectName, fields)

# Update a record
objectName <- "Account"
id <- "999x000000xxxxxZZZ"
fields <- c(Phone="9999-9999-9999")
rforcecom.update(session, objectName, id, fields)

# Upsert a record
objectName <- "Account";
externalIdField <- "AccountMaster__c"
externalId <- "AM-00000151"
fields <- c(Name="ABC Network Company", Phone="3333-3333-3333")
rforcecom.upsert(session, objectName, externalIdField, externalId, fields)

# Delete a record
objectName <- "Account";
id <- "999x000000xxxxxZZZ"
rforcecom.delete(session, objectName, id)

# Retrieve a server timestamp
rforcecom.getServerTimestamp(session)

# Logout
rforcecom.logout(session)

####
# Using the Bulk API
####

# Sign in to the Force.com
username <- "yourname@yourcompany.com"
password <- "YourPasswordSECURITY_TOKEN"
instanceURL <- "https://xxx.salesforce.com/"
apiVersion <- "34.0"
session <- rforcecom.login(username, password, instanceURL, apiVersion)


## BULK INSERT

# create a sample data.frame of 1000 records
n <- 1000
data <- data.frame(Name=paste('New Record:', 1:n),
                   stringsAsFactors=FALSE)

# run an insert job into the Account object
job_info <- rforcecom.createBulkJob(session,
                                    operation='insert',
                                    object='Account')
```

```
# split into batch sizes of 500 (2 batches for our 1000 row sample dataset)
batches_info <- rforcecom.createBulkBatch(session,
                                          jobId=job_info$id,
                                          data,
                                          multiBatch = TRUE,
                                          batchSize=500)

# check on status of each batch
batches_status <- lapply(batches_info,
                         FUN=function(x){
                          rforcecom.checkBatchStatus(session,
                                                     jobId=x$jobId,
                                                     batchId=x$id)
                                                     })
# get details on each batch
batches_detail <- lapply(batches_info,
                         FUN=function(x){
                          rforcecom.getBatchDetails(session,
                                                    jobId=x$jobId,
                                                    batchId=x$id)
                                                    })
# close the job
close_job_info <- rforcecom.closeBulkJob(session, jobId=job_info$id)


## BULK DELETE THE PRIOR INSERT

# format the data
batch_details_together <- plyr::ldply(batches_detail)
delete_ids <- data.frame(id=batch_details_together[,"Id"],
                         stringsAsFactors=FALSE)

job_info <- rforcecom.createBulkJob(session, operation='delete', object='Account')
batches_info <- rforcecom.createBulkBatch(session,
                                          jobId=job_info$id,
                                          data=delete_ids)
# check on status of each batch
batches_status <- lapply(batches_info,
                         FUN=function(x){
                          rforcecom.checkBatchStatus(session,
                                                     jobId=x$jobId,
                                                     batchId=x$id)
                                                     })
# get details on each batch
batches_detail <- lapply(batches_info,
                         FUN=function(x){
                          rforcecom.getBatchDetails(session,
                                                    jobId=x$jobId,
                                                    batchId=x$id)
                                                    })
# close the job
close_job_info <- rforcecom.closeBulkJob(session, jobId=job_info$id)
```

```
  ## BULK QUERY

  query <- "SELECT Id, Name FROM Account LIMIT 10"
  job_info <- rforcecom.createBulkJob(session, operation='query', object='Account')
  batch_query_info <- rforcecom.submitBulkQuery(session,
                                               jobId=job_info$id,
                                               query=query)

  batch_query_status <- rforcecom.checkBatchStatus(session,
                                                  jobId=batch_query_info$jobId,
                                                  batchId=batch_query_info$id)

  batch_query_details <- rforcecom.getBatchDetails(session,
                                                  jobId=batch_query_info$jobId,
                                                  batchId=batch_query_info$id)

  batch_query_recordset <- rforcecom.getBulkQueryResult(session,
                                                       jobId=batch_query_info$jobId,
                                                       batchId=batch_query_info$id,
                                                    resultId=batch_query_details$result)
  close_job_info <- rforcecom.closeBulkJob(session, jobId=job_info$id)


  ## BULK INSERT ATTACHMENTS

  # prepare your .zip file and request.txt manifest before calling these functions
  file <- 'request.zip'
  job_info <- rforcecom.createBulkJob(session, operation='insert', object='Attachment')
  batch_attachment_info <- rforcecom.insertBulkAttachments(session,
                                                          jobId=job_info$id,
                                                          file=file)
  batch_attachment_status <- rforcecom.checkBatchStatus(session,
                                                      jobId=batch_attachment_info$jobId,
                                                       batchId=batch_attachment_info$id)
  batch_attachment_details <- rforcecom.getBatchDetails(session,
                                                      jobId=batch_attachment_info$jobId,
                                                       batchId=batch_attachment_info$id)
  # close the job
  close_job_info <- rforcecom.closeBulkJob(session, jobId=job_info$id)


## End(Not run)
```

rforcecom.abortBulkJob

Abort Bulk API Job

Description

This function aborts a Job in the Salesforce Bulk API

Usage

```
rforcecom.abortBulkJob(session, jobId)
```

Arguments

`session`	a named character vector defining parameters of the api connection as returned by rforcecom.login
`jobId`	a character string defining the salesforce id assigned to a submitted job as returned by rforcecom.createBulkJob

Value

A `list` of parameters defining the now aborted job

References

`https://developer.salesforce.com/docs/atlas.en-us.api_asynch.meta/api_asynch/`

Examples

```
## Not run:
job_abort_info <- rforcecom.abortBulkJob(session, jobId=job_info$id)

## End(Not run)
```

`rforcecom.api` *Functions to connect API.*

Description

For internal use.

Usage

```
rforcecom.api.getSoapEndpoint(apiVersion)

rforcecom.api.getRestEndpoint(apiVersion)

rforcecom.api.getSoqlEndpoint(apiVersion)

rforcecom.api.getSoslEndpoint(apiVersion)

rforcecom.api.getObjectListEndpoint(apiVersion)

rforcecom.api.getObjectDescriptionEndpoint(apiVersion, objectName)
rforcecom.api.getObjectEndpoint(apiVersion, objectName)

rforcecom.api.getRecordEndpoint(apiVersion, objectName, id)

rforcecom.api.getFieldEndpoint(apiVersion, objectName, field)

rforcecom.api.getExternalIdFieldEndpoint(apiVersion, objectName, field, id)
```

Arguments

`apiVersion`	An REST API or SOAP API version to connect. (ex: "34.0". As of Oct. 2015)
`objectName`	An object name. (ex: "Account", "Contact", "CustomObject__c")
`field`	A field name. (ex: "Id", "Name", "Industry", "AnnualRevenue")
`id`	A record ID. (ex: "999x000000xxxxxZZZ")

Value

Return values depends on the function used.

Author(s)

Takekatsu Hiramura <thira@plavox.info>

`rforcecom.bulkQuery` *Run Bulk Query*

Description

This function is a convenience wrapper for submitting and retrieving bulk query API jobs

Usage

```
rforcecom.bulkQuery(session,
                           soqlQuery,
                           object,
                           interval_seconds=5,
                           max_attempts=100,
                           verbose=FALSE)
```

Arguments

`session`	a named character vector defining parameters of the api connection as returned by rforcecom.login
`soqlQuery`	a character string defining a SOQL query. (ex: "SELECT Id, Name FROM Account")

`object`	a character string defining the target salesforce object that the operation will be performed on. This must match the target object in the query
`interval_seconds`	an integer defining the seconds between attempts to check for job completion
`max_attempts`	an integer defining then max number attempts to check for job completion before stopping
`verbose`	a boolean on whether to print the API attempt numbers

Value

A `data.frame` of the recordset returned by query

References

https://developer.salesforce.com/docs/atlas.en-us.api_asynch.meta/api_asynch/

Examples

```
## Not run:
# select all Ids from Account object
ids <- rforcecom.bulkQuery(session, soqlQuery='Select Id from Account', object='Account')

## End(Not run)
```

`rforcecom.checkBatchStatus`

Checking the Status of a Batch in a Bulk API Job

Description

This function checks on and returns status information on an existing batch which has already been submitted to Bulk API Job

Usage

```
rforcecom.checkBatchStatus(session, jobId, batchId)
```

Arguments

`session`	a named character vector defining parameters of the api connection as returned by rforcecom.login
`jobId`	a character string defining the salesforce id assigned to a submitted job as returned by rforcecom.createBulkJob
`batchId`	a character string defining the salesforce id assigned to a submitted batch as returned by rforcecom.createBulkBatch

Value

A `list` of parameters defining the batch identified by the batchId

References

https://developer.salesforce.com/docs/atlas.en-us.api_asynch.meta/api_asynch/

Examples

```
## Not run:
batch_status <- rforcecom.checkBatchStatus(session, jobId=job_info$id, batchId=batches_info[[1]]$id)

## End(Not run)
```

`rforcecom.closeBulkJob` *Close Bulk API Job*

Description

This function closes a Job in the Salesforce Bulk API

Usage

```
rforcecom.closeBulkJob(session, jobId)
```

Arguments

`session`	a named character vector defining parameters of the api connection as returned by rforcecom.login
`jobId`	a character string defining the salesforce id assigned to a submitted job as returned by rforcecom.createBulkJob

Value

A `list` of parameters defining the now closed job

References

https://developer.salesforce.com/docs/atlas.en-us.api_asynch.meta/api_asynch/

Examples

```
## Not run:
job_close_info <- rforcecom.closeBulkJob(session, jobId=job_info$id)

## End(Not run)
```

`rforcecom.create` *Create a record*

Description

Create a record

Usage

```
rforcecom.create(session, objectName, fields)
```

Arguments

`session`	Session data. It can be retrieve from `rforcecom.login`.
`objectName`	An object name. (ex: "Account", "Contact", "CustomObject__c")
`fields`	Field names and values. (ex: Name="CompanyName", Phone="000-000-000")

Value

No data.

Author(s)

Takekatsu Hiramura <thira@plavox.info>

Examples

```
## Not run:
 objectName <- "Account"
 fields <- c(Name="R Analytics Service Ltd", Phone="5555-5555-5555")
 rforcecom.create(session, objectName, fields)

## End(Not run)
```

`rforcecom.createBulkBatch` *Create and Add Batches to a Bulk API Job*

Description

This function takes a data frame and submits it in batches to a an already existing Bulk API Job by chunking into temp files

Usage

```
rforcecom.createBulkBatch(session, jobId, data, multiBatch=TRUE, batchSize=10000)
```

Arguments

`session`	a named character vector defining parameters of the api connection as returned by rforcecom.login
`jobId`	a character string defining the salesforce id assigned to a submitted job as returned by rforcecom.createBulkJob
`data`	a matrix or data.frame that can be coerced into .csv file for submitting as batch request
`multiBatch`	a boolean value defining whether or not submit data in batches to the api
`batchSize`	an integer value defining the number of records to submit if multiBatch is true. The max value is 10000 in accordance with salesforce limits.

Value

A `list` of `list`s, one for each batch submitted, containing 10 parameters of the batch

References

`https://developer.salesforce.com/docs/atlas.en-us.api_asynch.meta/api_asynch/`

Examples

```
## Not run:

# inserting 2 records
my_data <- data.frame(Name=c('New Record 1', 'New Record 2'),
                      My_External_Id__c=c('11111','22222'),
                      stringsAsFactors=FALSE)
batches_info <- rforcecom.createBulkBatch(session,
                                          jobId=job_info$id,
                                          data=my_data,
                                          multiBatch=TRUE,
                                          batchSize=50)
#upserting 3 records
my_data <- data.frame(My_External_Id__c=c('11111','22222', '99999'),
                      Name=c('Updated_Name1', 'Updated_Name2', 'Upserted_Record'),
                      stringsAsFactors=FALSE)
batches_info <- rforcecom.createBulkBatch(session,
                                          jobId=job_info$id,
                                          data=my_data,
                                          multiBatch=TRUE,
                                          batchSize=50)

## End(Not run)
```

rforcecom.createBulkJob

Create Bulk API Job

Description

This function initializes a Job in the Salesforce Bulk API

Usage

```
rforcecom.createBulkJob(session,
                                   operation=c('insert', 'delete',
                                               'query', 'upsert',
                                               'update', 'hardDelete'),
                                   object='Account',
                                   externalIdFieldName=NULL,
                                   concurrencyMode='Parallel')
```

Arguments

`session`	a named character vector defining parameters of the api connection as returned by rforcecom.login
`operation`	a character string defining the type of operation being performed
`object`	a character string defining the target salesforce object that the operation will be performed on
`externalIdFieldName`	a character string identifying a custom field that has the "External ID" attribute on the target object. This field is only used when performing upserts. It will be ignored for all other operations.
`concurrencyMode`	a character string either "Parallel" or "Serial" that specifies whether batches should be completed sequentially or in parallel. Use "Serial" only if Lock contentions persist with in "Parallel" mode.

Value

A `list` parameters defining the created job, including id

References

https://developer.salesforce.com/docs/atlas.en-us.api_asynch.meta/api_asynch/

Examples

```
## Not run:
# insert into Account
job_info <- rforcecom.createBulkJob(session, operation='update', object='Account')

# delete from Account
job_info <- rforcecom.createBulkJob(session, operation='delete', object='Account')

# update into Account
job_info <- rforcecom.createBulkJob(session, operation='update', object='Account')

# upsert into Account
job_info <- rforcecom.createBulkJob(session, operation='upsert',
                                    externalIdFieldName='My_External_Id__c',
                                    object='Account')

# insert attachments
job_info <- rforcecom.createBulkJob(session, operation='insert', object='Attachment')

## End(Not run)
```

rforcecom.debug *For debug use.*

Description

For internal use.

Usage

```
rforcecom.debug
```

Value

If it is TRUE, print debug information. The default setting is FALSE.

Author(s)

Takekatsu Hiramura <thira@plavox.info>

`rforcecom.delete` *Delete a record*

Description

Delete a record

Usage

```
rforcecom.delete(session, objectName, id)
```

Arguments

`session`	Session data. It can be retrieve from `rforcecom.login`.
`objectName`	An object name. (ex: "Account", "Contact", "CustomObject__c")
`id`	Record ID to retrieve. (ex: "999x000000xxxxxZZZ")

Value

No data.

Author(s)

Takekatsu Hiramura <thira@plavox.info>

Examples

```
## Not run:
 # Deleting a record
 objectName <- "Account";
 id <- "999x000000xxxxxZZZ" # Record's Id
 rforcecom.delete(session, objectName, id)

## End(Not run)
```

`rforcecom.getBatchDetails` *Returning the Details of a Batch in a Bulk API Job*

Description

This function returns detailed (row-by-row) information on an existing batch which has already been submitted to Bulk API Job

Usage

```
rforcecom.getBatchDetails(session, jobId, batchId)
```

Arguments

`session`	a named character vector defining parameters of the api connection as returned by rforcecom.login
`jobId`	a character string defining the salesforce id assigned to a submitted job as returned by rforcecom.createBulkJob
`batchId`	a character string defining the salesforce id assigned to a submitted batch as returned by rforcecom.createBulkBatch

Value

A `data.frame`, formatted by salesforce, with information containing the success or failure or certain rows in a submitted batch, unless the operation was query, then it is a data.frame containing the resultId for retrieving the recordset.

References

https://developer.salesforce.com/docs/atlas.en-us.api_asynch.meta/api_asynch/

Examples

```
## Not run:
batch_details <- rforcecom.getBatchDetails(session, jobId=job_info$id, batchId=batches_info[[1]]$id)

## End(Not run)
```

---

rforcecom.getBulkQueryResult

Retrieving the Results of a Bulk Query Batch in a Bulk API Job

---

Description

This function returns the resultset of a Bulk Query batch which has already been submitted to Bulk API Job and has Completed state

Usage

```
rforcecom.getBulkQueryResult(session, jobId, batchId, resultId)
```

Arguments

`session`	a named character vector defining parameters of the api connection as returned by rforcecom.login
`jobId`	a character string defining the salesforce id assigned to a submitted job as returned by rforcecom.createBulkJob
`batchId`	a character string defining the salesforce id assigned to a submitted batch as returned by rforcecom.createBulkBatch
`resultId`	a character string returned from rforcecom.getBatchDetails when a query has completed and specifies how to get the recordset

Value

A `data.frame`, formatted by salesforce, containing query results

References

https://developer.salesforce.com/docs/atlas.en-us.api_asynch.meta/api_asynch/

Examples

```
## Not run:
result <- rforcecom.getBatchDetails(session,
                                    jobId=batch_query_info$jobId,
                                    batchId=batch_query_info$id)
recordset <- rforcecom.getBulkQueryResult(session,
                                          jobId=batch_query_info$jobId,
                                          batchId=batch_query_info$id,
                                          resultId=result$result)

## End(Not run)
```

`rforcecom.getObjectDescription`
Retrieve object descriptions

Description

Retrieve object descriptions

Usage

```
rforcecom.getObjectDescription(session, objectName)
```

Arguments

`session`	Session data. It can be retrieve from `rforcecom.login`.
`objectName`	An object name. (ex: "Account", "Contact", "CustomObject__c")

Value

Object descriptions

Author(s)

Takekatsu Hiramura <thira@plavox.info>

Examples

```
## Not run:
 # Retrieve object descriptions
  objectName <- "Account"
  rforcecom.getObjectDescription(session, objectName)

## End(Not run)
```

---

rforcecom.getObjectList

Retrieve a list of objects

Description

Retrieve a list of objects

Usage

```
rforcecom.getObjectList(session)
```

Arguments

session	Session data. It can be retrieve from `rforcecom.login`.

Value

A list of objects

Author(s)

Takekatsu Hiramura <thira@plavox.info>

Examples

```
## Not run:
 # Retrieve a list of objects
 rforcecom.getObjectList(session)

## End(Not run)
```

`rforcecom.getServerTimestamp`
Retrieve a server timestamp

Description

Retrieve a server timestamp

Usage

```
rforcecom.getServerTimestamp(session)
```

Arguments

`session`	Session data. It can be retrieve from `rforcecom.login`.

Value

Server timestamp (POSIXlt format).

Author(s)

Takekatsu Hiramura <thira@plavox.info>

Examples

```
## Not run:
 # Retrieve a server timestamp
 rforcecom.getServerTimestamp(session)
 # [1] "2012-04-01 11:30:05 GMT"

## End(Not run)
```

`rforcecom.insertBulkAttachments`
Insert Attachments via Bulk API Job

Description

This function takes a file path to a structured .zip file of attachments and submits it to an already existing Bulk API Job

Usage

```
rforcecom.insertBulkAttachments(session, jobId, file)
```

Arguments

`session`	a named character vector defining parameters of the api connection as returned by rforcecom.login
`jobId`	a character string defining the salesforce id assigned to a submitted job as returned by rforcecom.createBulkJob
`file`	a file path to a .zip file containing request.txt manifest formatted as CSV and any binary attachments. It should have request.txt in the top-level (aka base) directory. Attachment files can be in subdirectories if desired. See Salesforce documentation for details on how to format the .zip file.

Value

A `list` of parameters defining the created batch

References

https://developer.salesforce.com/docs/atlas.en-us.api_asynch.meta/api_asynch/

Examples

```
## Not run:

# sample .zip directory structure
request.zip
  request.txt
  attachment1.gif
  subdir/
    attachment2.doc

# sample format of request.txt
Name,ParentId,Body
attachment1.gif,TheTargetRecordIdHere,#attachment1.gif
subdir/attachment2.doc,TheTargetRecordIdHere,#subdir/attachment2.doc

f <- 'request.zip'
batch_attachment_info <- rforcecom.insertBulkAttachments(session, jobId=job_info$id, file=f)

## End(Not run)
```

`rforcecom.login` *Sign in to the Force.com (Salesforce.com)*

Description

This function retrives a session ID from Salesforce.com.

Usage

```
rforcecom.login(username, password, loginURL, apiVersion)
```

Arguments

`username`	Your username for login to the Salesforce.com. In many cases, username is your E-mail address.
`password`	Your password for login to the Salesforce.com. Note: DO NOT FORGET your Security Token. (Ex.) If your password is "Pass1234" and your security token is "XYZXYZXYZXYZ", you should set "Pass1234XYZXYZXYZXYZ".
`loginURL`	(optional) Login URL. If your environment is sandbox specify (ex:) "https://test.salesforce.com/".
`apiVersion`	(optional) Version of the REST API and SOAP API that you want to use. (ex:) "35.0" Supported versions from v20.0 and up.

Value

`sessionID`	Session ID.
`instanceURL`	Instance URL.
`apiVersion`	API Version.

See Also

`rforcecom.query` `rforcecom.search` `rforcecom.create` `rforcecom.delete` `rforcecom.retrieve` `rforcecom.update` `rforcecom.upsert` `rforcecom.getServerTimestamp` `rforcecom.getObjectDescription` `rforcecom.getObjectList` `rforcecom.queryMore`

Examples

```
## Not run:
# Sign in to the Force.com
username <- "yourname@yourcompany.com"
password <- "YourPasswordSECURITY_TOKEN"
session <- rforcecom.login(username, password)

## End(Not run)
```

`rforcecom.logout` *Sign out of the Force.com (Salesforce.com)*

Description

Sign out of the Force.com (Salesforce.com)

Usage

```
rforcecom.logout(session)
```

Arguments

`session`	Session data. It can be retrieve from `rforcecom.login`.

Value

`success`	boolean

See Also

`rforcecom.login`

Examples

```
## Not run:
 # Sign out of the Force.com
 rforcecom.logout(session)

## End(Not run)
```

`rforcecom.query` *Execute a SOQL*

Description

Execute a SOQL

Usage

```
rforcecom.query(session, soqlQuery, queryAll=FALSE)
```

Arguments

`session`	Session data. It can be retrieve from `rforcecom.login`.
`soqlQuery`	A SOQL query. (ex: "SELECT Id, Name FROM Account")
`queryAll`	A boolean. Indicate if query should include deleted and archived records (available only on Task and Event records)

Value

Result dataset.

Author(s)

Takekatsu Hiramura <thira@plavox.info>

Examples

```
## Not run:
 # Execute a SOQL
 soqlQuery <- "SELECT Id, Name, Industry, AnnualRevenue FROM Account"
 rforcecom.query(session, soqlQuery)

 # Include records where IsDeleted=true
 soqlQuery <- "SELECT Id, IsDeleted, Subject, Description FROM Task"
 rforcecom.query(session, soqlQuery, queryAll=TRUE)

## End(Not run)
```

`rforcecom.queryMore` *Retrieve remain records when executing a SOQL*

Description

Retrieve remain records when executing a SOQL

Usage

```
rforcecom.queryMore(session, nextRecordsUrl)
```

Arguments

`session`	Session data. It can be retrieve from `rforcecom.login`.
`nextRecordsUrl`	An URL for next records. (ex: "/services/data/v26.0/query/999999999999999ZZZ-2000")

Value

Result dataset.

Author(s)

Takekatsu Hiramura <thira@plavox.info>

Examples

```
## Not run:
 # Retrieve remain records when executing a SOQL
 nextRecordsUrl <- "/services/data/v26.0/query/999999999999999ZZZ-2000"
 rforcecom.queryMore(session, soqlQuery)

## End(Not run)
```

rforcecom.retrieve *Retrieve a record*

Description

Retrieve a record

Usage

```
rforcecom.retrieve(session, objectName, fields, limit=NULL, id=NULL,
                     offset=NULL, order=NULL, inverse=NULL, nullsLast=NULL)
```

Arguments

`session`	Session data. It can be retrieved from `rforcecom.login`.
`objectName`	An object name. (ex: "Account", "Contact", "CustomObject__c")
`fields`	A List of field names. (ex: c("Id", "Name", "Industry", "AnnualRevenue)"))
`limit`	Number of the records to retrieve. (ex: 5)
`id`	Record ID to retrieve. (ex: "999x000000xxxxxZZZ")
`offset`	Specifies the starting row offset. (ex: "100")
`order`	A list for controling the order of query results. (ex: "c("Industry","Name")")
`inverse`	If it is TRUE, the results are ordered in descending order. This parameter works when order parameter has been set. (Default: FALSE)
`nullsLast`	If it is TRUE, null records list in last. If not null records list in first. This parameter works when order parameter has been set. (Default: FALSE)

Value

Result dataset.

Author(s)

Takekatsu Hiramura <thira@plavox.info>

See Also

`rforcecom.login`

Examples

```
## Not run:
 # Retrieving a record
 objectName <- "Account"
 fields <- c("name", "Industry", "AnnualRevenue")
 rforcecom.retrieve(session, objectName, fields)
 rforcecom.retrieve(session, objectName, fields, limit = 5)
 rforcecom.retrieve(session, objectName, fields, id = "999x000000xxxxxZZZ")
 rforcecom.retrieve(session, objectName, fields)
 rforcecom.retrieve(session, objectName, fields, order = c("Industry","Name"))
 rforcecom.retrieve(session, objectName, fields, order = c("Industry","Name"),
                    inverse=TRUE, nullsLast=TRUE)

## End(Not run)
```

`rforcecom.search` *Execute a SOSL*

Description

Execute a SOSL

Usage

```
rforcecom.search(session, queryString)
```

Arguments

`session`	Session data. It can be retrieve from `rforcecom.login`.
`queryString`	Query strings to search. (ex: "United", "Electoronics")

Value

Result dataset.

Author(s)

Takekatsu Hiramura <thira@plavox.info>

Examples

```
## Not run:
 # Execute a SOSL
 queryString <- "United"
 rforcecom.search(session, queryString)

## End(Not run)
```

rforcecom.submitBulkQuery
Submit Bulk Query Batch to a Bulk API Job

Description

This function takes a SOQL text string and submits the query to an already existing Bulk API Job of operation "query"

Usage

```
rforcecom.submitBulkQuery(session, jobId, query)
```

Arguments

`session`	a named character vector defining parameters of the api connection as returned by rforcecom.login
`jobId`	a character string defining the salesforce id assigned to a submitted job as returned by rforcecom.createBulkJob
`query`	a character string defining a valid SOQL query on the Salesforce object associated with the job

Value

A `list` parameters of the batch

Note

Bulk API query doesn't support the following SOQL:

- COUNT
- ROLLUP
- SUM
- GROUP BY CUBE
- OFFSET
- Nested SOQL queries
- Relationship fields

Additionally, Bulk API can't access or query compound address or compound geolocation fields.

References

`https://developer.salesforce.com/docs/atlas.en-us.api_asynch.meta/api_asynch/`

Examples

```
## Not run:
my_query <- "SELECT Id, Name FROM Account LIMIT 10"
query_info <- rforcecom.submitBulkQuery(session, jobId=job_info$id, query=my_query)

## End(Not run)
```

rforcecom.update *Update a record*

Description

Update a record

Usage

```
rforcecom.update(session, objectName, id, fields)
```

Arguments

`session`	Session data. It can be retrieve from `rforcecom.login`.
`objectName`	An object name. (ex: "Account", "Contact", "CustomObject__c")
`id`	Record ID to update. (ex: "999x000000xxxxxZZZ")
`fields`	Sets of field name and its value. (ex: Name="CompanyName", Phone="000-000-000")

Value

No data.

Author(s)

Takekatsu Hiramura <thira@plavox.info>

Examples

```
## Not run:
 # Updating a record
 objectName <- "Account"
 id <- "999x000000xxxxxZZZ"
 fields <- c(Phone="9999-9999-9999")
 rforcecom.update(session, objectName, id, fields)

## End(Not run)
```

rforcecom.upsert *Upsert a record*

Description

Upsert a record

Usage

```
rforcecom.upsert(session, objectName, externalIdField, externalId, fields)
```

Arguments

`session`	Session data. It can be retrieve from `rforcecom.login`.
`objectName`	An object name. (ex: "Account", "Contact", "CustomObject__c")
`externalIdField`	An external Key's field name. (ex: "AccountMaster__c")
`externalId`	An external Key's ID. (ex: "999x000000xxxxxZZZ")
`fields`	Sets of field name and its value. (ex: Name="CompanyName", Phone="000-000-000")

Value

No data.

Author(s)

Takekatsu Hiramura <thira@plavox.info>

Examples

```
## Not run:
 # Upsert a record
 objectName <- "Account";
 externalIdField <- "AccountMaster__c"
 externalId <- "AM-00000151"
 fields <- c(Name="ABC Network Company", Phone="3333-3333-3333")
 rforcecom.upsert(session, objectName, externalIdField, externalId, fields)

## End(Not run)
```

Index