Package ‘RonFHIR’

January 10, 2020

Type Package
Title Read and Search Interface to the 'HL7 FHIR' REST API
Version 0.4.0
Description
R on FHIR is an easy to use wrapper around the 'HL7 FHIR' REST API (STU 3 and R4). It provides tools to easily read and search resources on a FHIR server and brings the results into the R environment. R on FHIR is based on the FhirClient of the official 'HL7 FHIR .NET API', also made by Firely.
License BSD 3-clause License + file LICENSE
Imports R6, httr, jsonlite, utils, stringr
Suggests testthat
Encoding UTF-8
LazyData true
Depends R (>= 3.1.0)
NeedsCompilation no
RoxygenNote 6.0.1
Author Sander Laverman [aut, cre], Firely B.V. [aut, cph]
Maintainer Sander Laverman <sander@fire.ly>
Repository CRAN
Date/Publication 2020-01-10 12:30:05 UTC

R topics documented:

  fhirBulkClient .................................................. 2
  fhirClient ....................................................... 4
  searchParams .................................................... 6

Index 9
**Description**

Bulk data client in R for FHIR STU 3.

**Usage**

```r
bulkclient <- fhirBulkClient$new(endpoint, tokenURL = NULL, token = NULL)

bulkclient$patientExport(criteria = NULL)
bulkclient$groupExport(groupId, criteria = NULL)
bulkclient$wholeSystemExport(criteria = NULL)
bulkclient$getBulkStatus()
bulkclient$downloadBulk(requestNumber, returnType = "parsed", deleteFromQueue = TRUE)
bulkclient$deleteBulkRequest(requestNumber)
bulkclient$retrieveToken(jwt, scopes, tokenURL = NULL)
bulkclient$setToken(token)

print(bulkclient)
```

**Arguments**

- `bulkclient` A `fhirBulkClient` object.
- `endpoint` The URL of the server to connect to.
- `tokenURL` Authorization server’s endpoint.
- `token` Access token.
- `criteria` The search parameters to filter the Resources on. Each given string is a combined key/value pair (separated by '=').
- `groupId` Id of the Group resource.
- `requestNumber` Number of the request in the queue.
- `returnType` Specify the return type. This can be "parsed" or "ndjson".
- `deleteFromQueue` If the request needs to be deleted from the queue after it’s been downloaded.
- `jwt` JSON Web Token signed with the app’s private key (RSA SHA-256).
- `scopes` Desired scope(s).

**Details**

- `$new()` Creates a new `fhirBulkClient` using a given endpoint. If the endpoint does not end with a slash (/), it will be added.
- `$patientExport()` Request all data on all patients. Possible to filter the results with the _output-Format, _since and _type parameters. The request will be added to the queue.
$groupExport() Request all data of a patient group. Possible to filter the results with the _output-Format, _since and _type parameters. The request will be added to the queue.

$wholeSystemExport() Request all data. Possible to filter the results with the _outputFormat, _since and _type parameters. The request will be added to the queue.

$getBulkStatus() Update and return the queue to see the progress of your requests.

$downloadBulk() Download a request from the queue.

$deleteBulkRequest() Cancel a request from the queue.

$retrieveToken() Retrieve a token from the authentication server.

$setToken Set a token.

print(p) or p$print() Shows which endpoint is configured.

Examples

```r
## Not run:
# Read your private key
privatekey <- openssl::read_key("PrivateKey.pem")

# Create your claim
claim <- jose::jwt_claim(iss = "ServiceURL",
sub = "ClientID",
aud = "TokenURL",

# expiration date as epoch (5 minutes)
exp = as.integer(as.POSIXct(Sys.time() + 300)),

# 'random' number
jti = charToRaw(as.character(runif(1, 0.5, 100000000000))))

# Sign your claim with your private key
jwt <- jose::jwt_encode_sig(claim, privatekey)

# Define your scope(s)
scopes <- c("system/*.read", "system/CommunicationRequest.write")

# Create a new fhirBulkClient
bulkclient <- fhirBulkClient$new("FHIRBulkServerURL", tokenURL = "TokenURL")

# Retrieve your token
token <- bulkclient$retrieveToken(jwt, scopes)

# Set your token
bulkclient$setToken(token$access_token)

# Request a download for Patient Cohort 3
bulkclient$groupExport(3)

# Request the progress of the requests
bulkclient$getBulkStatus()

# When the downloads a available, download the bulkdata
```
patient_cohort_3 <- bulkclient$downloadBulk()

View(patient_cohort_3)

## End(Not run)

### Description

Read and search only client in R for FHIR STU 3 and R4. Based on the official HL7 FHIR .NET API.

### Usage

```r
client <- fhirClient$new(endpoint, token = NULL)

client$read(location, summaryType = NULL, returnType = "parsed")
client$search(resourceType, criteria = NULL, includes = NULL, pageSize = NULL, summaryType = NULL, returnType = "parsed")
client$searchById(resourceType, id, includes = NULL, summaryType = NULL, returnType = "parsed")
client$wholeSystemSearch(criteria = NULL, includes = NULL, pageSize = NULL, summaryType = NULL, returnType = "parsed")
client$searchParams(params, resourceType = NULL, returnType = "parsed")
client$continue(bundle)

client$setToken(token)

client$endpoint
client$authUrl
client$tokenUrl
client$registerUrl
client$token

print(client)
```

### Arguments

- **client** A fhirClient object.
- **endpoint** The URL of the server to connect to.
- **token** An oauth 2.0 Token (httr Token 2.0)
- **resourceType** The type of resource to search for.
- **id** The id of the Resource to search for.
- **summaryType** Whether to include only return a summary of the Resource(s).
- **location** The url of the Resource to fetch. This can be a Resource id url or a version-specific.
**criteria**  The search parameters to filter the Resources on. Each given string is a combined key/value pair (separated by '=').

**includes** Paths to include in the search.

**pageSize** Asks server to limit the number of entries per page returned.

**query** A searchParams object containing the search parameters.

**bundle** The bundle as received from the last response.

**returnType** Specify the return type. This can be "parsed", "json" or "xml".

**Details**

$:new() Creates a new fhirClient using a given endpoint. If the endpoint does not end with a slash (/), it will be added.

$:read() Fetches a typed Resource from a FHIR resource endpoint.

$:search() Search for Resources of a certain type that match the given criteria.

$:searchById() Search for Resources based on a Resource’s id.

$:wholeSystemSearch() Search for Resources across the whole server that match the given criteria.

$:searchByQuery() Search for Resources based on a searchParams object.

$:continue() Uses the FHIR paging mechanism to go navigate around a series of paged result Bundles.

$:setToken() Saves an Oauth 2.0 token in a variable.

$:endpoint Returns the endpoint.

$:authUrl Returns the authorization server’s OAuth authorization endpoint.

$:tokenUrl Returns the authorization server’s OAuth token endpoint.

$:registerUrl Returns the endpoint where the client can register.

$:token Returns the initialized token.

print(p) or p$print() Shows which endpoint is configured.

**Examples**

```r
## Not run:
# Setting up a fhirClient
client <- fhirClient$new("https://vonk.fire.ly")
# Read
client$read("Patient/example")

# Search
bundle <- client$search("Patient", c("name=Peter", "address-postalcode=3999"))

while(!is.null(bundle)){
  # Do something useful
  bundle <- client$continue(bundle)
}

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

### Not run:

```r
# Using Oauth 2.0
client <- fhirClient$new("https://vonk.fire.ly")

# Retrieving a token

client_id <- "id"
client_secret <- "secret"
app_name <- "TestApp"
scopes <- c("patient/*.read")

app <- http::oauth_app(appname = app_name, client_id, client_secret)
oauth_endpoint <- http::oauth_endpoint(
  authorize = paste(client$authUrl, "?aud=", client$endpoint, sep=""),
  access = client$tokenUrl)

token <- http::oauth2.0_token(endpoint = oauth_endpoint, app = app, scope = scopes)

# Set a token and read a patient resource
client$setToken(token$credentials$access_token)

client$read("Patient/example")

# Token refresh
token <- token$refresh()

client$setToken(token$credentials$access_token)

## End(Not run)
```

---

#### searchParams

**Description**

An alternative way to specify a query is by creating a `searchParams` object and pass this to the `fhirClient`'s `searchByQuery`. The `searchParams` class has a set of fluent calls to allow you to easily construct more complex queries. Based on the official HL7 FHIR .NET API.

#### Usage

```r
query <- searchParams$new()

query$select(elements)
query$where(criteria)
query$include(path)
query$orderBy(paramName, sortOrder = "asc")
query$limitTo(count)
```
searchParams

query$countOnly()
query$summaryOnly()
query$textOnly()
query$dataOnly()

Arguments

query  A searchParams object that contains all specified search criteria.
elements  Elements defined at the root level in the Resource.
criteria  The search parameters to filter the Resources on. Each given string is a combined key/value pair (separated by '=').
path  Paths to include in the search.
paramName  Name of the parameter to order by.
sortOrder  Direction of the order. Can be asc or desc (ascending and descending).
count  The number of returned Resources per page.

Details

$new()  Creates a new searchParams object.
$select()  Specify the elements to be returned as part of a Resource.
$where()  Specify on which parameters to filter.
$include()  Specify the paths to include.
$orderBy()  Specify the order to return the results.
$limitTo()  Specify how many Resources should be returned in a single page of a Bundle.
$countOnly()  Specify to just return a count of the matching Resources, without returning the actual matches.
$summaryOnly()  Specify to return only those elements marked as "summary" in the base definition of the Resource(s).
$textOnly()  Specify to return only the "text" element, the 'id' element, the 'meta' element, and only top-level mandatory elements.
$dataOnly()  Specify to remove the text element.

Examples

```r
## Not run:
# Setting up a fhirClient
client <- fhirClient$new("http://vonk.furore.com")

# Creating a new searchParams object
query <- searchParams$new()
query$select(c("name", "birthDate"))$where("given:exact=Peter")$orderBy("family")

peters <- client$searchByQuery(query, "Patient")
# equivalent:
```
# client$search("Patient", c("_elements=name,birthDate","given:exact=Peter","_sort=family"))

while(!is.null(bundle)){
    # Do something useful
    peters <- client$continue(peters)
}

## End(Not run)
Index

fhirBulkClient, 2
fhirClient, 4
fhirClient's, 6
searchParams, 6