

Package ‘viafoundry’

August 22, 2025

Type Package

Title R Client for 'Via Foundry' API

Version 1.0.1

Language en-US

Description 'Via Foundry' API provides streamlined tools for interacting with and extracting data from structured responses, particularly for use cases involving hierarchical data from Foundry's API. It includes functions to fetch and parse process-level and file-level metadata, allowing users to efficiently query and manipulate nested data structures. Key features include the ability to list all unique process names, retrieve file metadata for specific or all processes, and dynamically load or download files based on their type. With built-in support for handling various file formats (e.g., tabular and non-tabular files) and seamless integration with API through authentication, this package is designed to enhance workflows involving large-scale data management and analysis. Robust error handling and flexible configuration ensure reliable performance across diverse data environments. Please consult the documentation for the API endpoint for your installation.

License Apache License 2.0

Encoding UTF-8

Imports httr, jsonlite, dplyr, askpass, stringr, mime

RoxygenNote 7.3.2

URL <https://github.com/ViaScientific/viafoundry-R-SDK>

NeedsCompilation no

Author Alper Kucukural [aut, cre],
Via Scientific [aut, cph]

Maintainer Alper Kucukural <alper@viascientific.com>

Repository CRAN

Date/Publication 2025-08-22 21:10:02 UTC

Contents

addFilesToDataset	3
authenticate	3

calculate_expiration_date	4
call_endpoint	5
checkProcessUsage	5
createCanvas	6
createCollection	7
createField	7
createMenuGroup	8
createParameter	8
createProcess	9
createProcessConfig	9
deleteCanvas	10
deleteCollection	11
deleteData	11
deleteField	12
deleteParameter	12
deleteProcess	13
discover	13
duplicateProcess	14
fetchReportData	14
filterParameters	15
getAllFileNames	15
getAllReportPaths	16
getCanvas	16
getCanvasFields	17
getCollection	17
getCollectionFields	18
getData	18
getField	19
getFieldsForCollection	19
getFileNames	20
getMenuGroupByName	20
getPipelineParameters	21
getProcess	21
getProcessNames	22
getProcessRevisions	22
getReportDirs	23
get_api_status	23
get_bearer_token	24
get_headers	24
listMenuGroups	25
listParameters	25
listProcesses	25
loadFile	26
load_config	26
login	27
prepareSessionHistory	27
searchCanvas	28
searchCollections	29

Usage

```
authenticate(  
    hostname,  
    username = NULL,  
    password = NULL,  
    identity_type = 1,  
    redirect_uri = "http://localhost",  
    config_path = DEFAULT_CONFIG_PATH,  
    overwrite = FALSE  
)
```

Arguments

hostname	The API url.
username	The login username.
password	The login password (optional; will prompt if not provided).
identity_type	The identity type (default is 1).
redirect_uri	The redirect 'URI'.
config_path	Path to save the configuration file.
overwrite	Logical flag to overwrite the existing configuration file (default is FALSE).

Value

None. Saves the bearer token to the configuration file and sets the global config path.

calculate_expiration_date

Calculate expiration date (30 days from now)

Description

Calculate expiration date (30 days from now)

Usage

```
calculate_expiration_date()
```

Value

The expiration date as a string.

call_endpoint	<i>Call an API Endpoint with Optional File Upload</i>
---------------	---

Description

Sends an HTTP request to a specified API endpoint using the stored bearer token for authentication, optionally supporting file uploads.

Usage

```
call_endpoint(method, endpoint, params = list(), data = NULL, files = NULL)
```

Arguments

method	The HTTP method (e.g., "GET", "POST", "PUT", "DELETE").
endpoint	The API endpoint.
params	A named list of query parameters (optional).
data	A named list or 'JSON' object to include in the request body (optional).
files	A list of files to upload, where each file is a named list with 'name', 'path', and optionally 'type'.

Value

A list containing the API response.

Examples

```
## Not run:
response <- call_endpoint("POST", "/api/projects/upload", files = list(
  list(name = "file1", path = "path/to/file1.txt", type = "text/plain"),
  list(name = "file2", path = "path/to/file2.csv", type = "text/csv")
))
print(response)

## End(Not run)
```

checkProcessUsage	<i>Check Process Usage</i>
-------------------	----------------------------

Description

Checks if a process is used in pipelines or runs.

Usage

```
checkProcessUsage(processID)
```

Arguments

processID The ID of the process.

Value

A JSON object indicating usage information.

createCanvas	<i>Create Canvas</i>
--------------	----------------------

Description

Create Canvas

Usage

```
createCanvas(canvasData)
```

Arguments

canvasData A named list with canvas details.

Value

A JSON object with the created canvas.

Examples

```
## Not run:
canvasData <- list(
  name = "new_canvas",
  label = "New canvas"
)
createCanvas(canvasData)

## End(Not run)
```

createCollection	Create Collection
------------------	-------------------

Description

Create Collection

Usage

createCollection(collectionData)

Arguments

collectionData A named list with collection details.

Value

A JSON object with the created collection.

createField	Create Field
-------------	--------------

Description

Create Field

Usage

createField(fieldData)

Arguments

fieldData A named list with field details.

Value

A JSON object with the created field.

createMenuGroup	Create Menu Group
-----------------	-------------------

Description

Creates a new menu group.

Usage

createMenuGroup(name)

Arguments

name	The name of the menu group to create.
------	---------------------------------------

Value

A JSON object containing the created menu group details.

createParameter	Create a Parameter
-----------------	--------------------

Description

Creates a new parameter with the given data.

Usage

createParameter(parameterData)

Arguments

parameterData	A list containing the parameter data.
---------------	---------------------------------------

Value

A JSON object containing the created parameter details.

createProcess	Create a New Process
---------------	----------------------

Description

Creates a new process using the provided data.

Usage

```
createProcess(processData)
```

Arguments

processData A list containing the process data.

Value

A JSON object containing the created process details.

createProcessConfig	Create Process Config
---------------------	-----------------------

Description

Assembles a complete process configuration object.

Usage

```
createProcessConfig(  
  name,  
  menuGroupName,  
  inputParams,  
  outputParams,  
  summary = "",  
  scriptBody = "",  
  scriptLanguage = "bash",  
  scriptHeader = "",  
  scriptFooter = "",  
  permissionSettings = list(viewPermissions = 3, writeGroupIds = list()),  
  revisionComment = "Initial revision"  
)
```

Arguments

name	Name of the process.
menuGroupName	Name of the menu group.
inputParams	A list of input parameter specs.
outputParams	A list of output parameter specs.
summary	Optional. Summary string.
scriptBody	Optional. Script body.
scriptLanguage	Optional. Language (default "bash").
scriptHeader	Optional. Script header.
scriptFooter	Optional. Script footer.
permissionSettings	Optional. A list of permission settings.
revisionComment	Optional. Revision comment.

Value

A list representing the full process config.

deleteCanvas	<i>Delete canvas</i>
--------------	----------------------

Description

Delete canvas

Usage

deleteCanvas(canvasID)

Arguments

canvasID	A character string with canvas ID.
----------	------------------------------------

Value

A confirmation object or message.

Examples

```
## Not run:
canvasData <- list(
  name = "new_canvas",
  label = "New canvas"
)
createCanvas(canvasData)
search_params <- list(
  filter = list(
    name="new_canvas"
  )
)
canvas <- searchCanvas(search_params)
deleteCanvas(canvas$data$`_id`[1])

## End(Not run)
```

deleteCollection	<i>Delete Collection</i>
------------------	--------------------------

Description

Delete Collection

Usage

```
deleteCollection(collectionID)
```

Arguments

collectionID A character string with collection ID.

Value

A confirmation object or message.

deleteData	<i>Delete Data Entry</i>
------------	--------------------------

Description

Deletes a data entry by ID.

Usage

```
deleteData(canvasID, collectionName, dataID)
```

Arguments

- canvasID The canvas ID.
- collectionName The collection name.
- dataID The ID of the data entry.

Value

A JSON confirmation.

deleteField	<i>Delete Field</i>
-------------	---------------------

Description

Delete Field

Usage

deleteField(fieldID)

Arguments

- fieldID A character string with field ID.

Value

A confirmation object or message.

deleteParameter	<i>Delete a Parameter</i>
-----------------	---------------------------

Description

Deletes a specific parameter by its ID.

Usage

deleteParameter(parameterID)

Arguments

- parameterID The ID of the parameter to delete.

Value

A confirmation message.

deleteProcess	<i>Delete a Process</i>
---------------	-------------------------

Description

Deletes a specific process by its ID.

Usage

```
deleteProcess(processID)
```

Arguments

processID	The ID of the process to delete.
-----------	----------------------------------

Value

A confirmation message.

discover	<i>Discover Available Endpoints</i>
----------	-------------------------------------

Description

Fetches and lists all available API endpoints from the Swagger documentation.

Usage

```
discover()
```

Value

A character vector of available endpoints.

Examples

```
## Not run:  
discover()  
  
## End(Not run)
```

duplicateProcess	<i>Duplicate Process</i>
------------------	--------------------------

Description

Creates a duplicate of an existing process.

Usage

```
duplicateProcess(processID)
```

Arguments

processID	The ID of the process to duplicate.
-----------	-------------------------------------

Value

A JSON object containing the duplicated process details.

fetchReportData	<i>Fetch the 'JSON' data for a report</i>
-----------------	---

Description

Fetch the 'JSON' data for a report

Usage

```
fetchReportData(reportID)
```

Arguments

reportID	The ID of the report to fetch data for.
----------	---

Value

The 'JSON' object containing the report data.

filterParameters	<i>Filter Parameters</i>
------------------	--------------------------

Description

Filters parameters by name, qualifier, file type, and/or ID.

Usage

```
filterParameters(name = NULL, qualifier = NULL, fileType = NULL, id = NULL)
```

Arguments

name	Optional. Name to filter.
qualifier	Optional. Qualifier to filter.
fileType	Optional. File type to filter.
id	Optional. ID to filter.

Value

A list of filtered parameters.

getAllFileNames	<i>Recursively extract all files from deeply nested report JSON</i>
-----------------	---

Description

Recursively extract all files from deeply nested report JSON

Usage

```
getAllFileNames(json_data)
```

Arguments

json_data	The parsed report JSON returned from fetch Report Data function
-----------	---

Value

A data frame of all files and their metadata

getAllReportPaths	<i>Get unique report directories and attempt IDs for a specific report</i>
-------------------	--

Description

Get unique report directories and attempt IDs for a specific report

Usage

```
getAllReportPaths(report_id)
```

Arguments

report_id	The ID of the report.
-----------	-----------------------

Value

A character vector of unique report directories.

getCanvas	<i>Get canvas</i>
-----------	-------------------

Description

Get canvas

Usage

```
getCanvas(canvasID)
```

Arguments

canvasID	A character string with canvas ID.
----------	------------------------------------

Value

A JSON object with canvas details.

Examples

```
## Not run:
canvasData <- list(
  name = "new_canvas",
  label = "New canvas"
)
createCanvas(canvasData)
search_params <- list(
  filter = list(
    name="new_canvas"
  )
)
canvas <- searchCanvas(search_params)
getCanvas(canvas$data$`_id`[1])

## End(Not run)
```

getCanvasFields	<i>Get canvas Fields</i>
-----------------	--------------------------

Description

Retrieves all metadata fields for a given canvas by collecting them from each collection.

Usage

```
getCanvasFields(canvasID)
```

Arguments

canvasID The ID of the canvas.

Value

A JSON object with a list of all fields.

getCollection	<i>Get Collection</i>
---------------	-----------------------

Description

Get Collection

Usage

```
getCollection(collectionID)
```

Arguments

collectionID A character string with collection ID.

Value

A JSON object with collection details.

getCollectionFields	<i>Get Collection Fields</i>
---------------------	------------------------------

Description

Retrieves metadata fields by collection ID.

Usage

```
getCollectionFields(collectionID)
```

Arguments

collectionID The ID of the collection.

Value

A JSON object with a list of fields.

getData	<i>Get Data Entry</i>
---------	-----------------------

Description

Retrieves a single data record by ID.

Usage

```
getData(canvasID, collectionName, dataID)
```

Arguments

canvasID The canvas ID.
collectionName The collection name.
dataID The ID of the data record.

Value

A JSON object with data entry.

`getField`*Get Field*

Description

Get Field

Usage

```
getField(fieldID)
```

Arguments

fieldID A character string with field ID.

Value

A JSON object with field details.

`getFieldsForCollection`*Get Fields for a Specific Collection in a canvas*

Description

Given a canvas ID and collection name, returns the metadata fields defined for that collection.

Usage

```
getFieldsForCollection(canvasID, collectionName)
```

Arguments

canvasID The ID of the canvas.
collectionName The name of the collection (e.g., "surveys").

Value

A list of field definitions for the collection.

getFileNames	<i>Get file names for a specific process</i>
--------------	--

Description

Get file names for a specific process

Usage

```
getFileNames(json_data, processName)
```

Arguments

json_data	The 'JSON' object containing the report data.
processName	The name of the process to filter by.

Value

A data frame containing 'id', 'name', 'extension', 'file size', and 'route path'.

getMenuGroupByName	<i>Get Menu Group by Name</i>
--------------------	-------------------------------

Description

Searches for a menu group by name and returns its ID.

Usage

```
getMenuGroupByName(groupName)
```

Arguments

groupName	The name of the menu group.
-----------	-----------------------------

Value

The ID of the menu group if found, otherwise NULL.

getPipelineParameters	<i>Get Pipeline Parameters</i>
-----------------------	--------------------------------

Description

Retrieves the list of parameters for a specific pipeline.

Usage

```
getPipelineParameters(pipelineID)
```

Arguments

pipelineID	The ID of the pipeline.
------------	-------------------------

Value

A JSON object containing the list of parameters.

getProcess	<i>Get Process Information</i>
------------	--------------------------------

Description

Fetches detailed information about a specific process.

Usage

```
getProcess(processID)
```

Arguments

processID	The ID of the process to retrieve.
-----------	------------------------------------

Value

A JSON object containing the process details.

getProcessNames	<i>Get unique process names</i>
-----------------	---------------------------------

Description

Get unique process names

Usage

```
getProcessNames(json_data)
```

Arguments

json_data	The 'JSON' object containing the report data.
-----------	---

Value

A character vector of unique process names.

getProcessRevisions	<i>Get Process Revisions</i>
---------------------	------------------------------

Description

Fetches all revisions for a given process.

Usage

```
getProcessRevisions(processID)
```

Arguments

processID	The ID of the process.
-----------	------------------------

Value

A JSON object containing the revisions.

getReportDirs	<i>Get directories following pub web in the route path</i>
---------------	--

Description

Get directories following pub web in the route path

Usage

```
getReportDirs(report_id)
```

Arguments

report_id	The ID of the report.
-----------	-----------------------

Value

A character vector of unique directories found after pub web.

get_api_status	<i>Get API Status</i>
----------------	-----------------------

Description

Sends a simple GET request to check the status of the API.

Usage

```
get_api_status()
```

Value

A character vector with the API status.

Examples

```
## Not run:  
status <- get_api_status()  
print(status)  
  
## End(Not run)
```

get_bearer_token	<i>Get bearer token using the cookie token</i>
------------------	--

Description

Get bearer token using the cookie token

Usage

```
get_bearer_token(hostname, cookie_token, name = "token")
```

Arguments

hostname	The API url
cookie_token	The cookie token.
name	The name of the token (default is "token").

Value

The bearer token.

get_headers	<i>Get headers for API requests</i>
-------------	-------------------------------------

Description

Get headers for API requests

Usage

```
get_headers()
```

Value

A list of headers with the bearer token.

listMenuGroups	<i>List Menu Groups</i>
----------------	-------------------------

Description

Fetches all menu groups from the API.

Usage

```
listMenuGroups()
```

Value

A JSON object containing the list of menu groups.

listParameters	<i>List All Parameters</i>
----------------	----------------------------

Description

Fetches all parameters from the API.

Usage

```
listParameters()
```

Value

A JSON object containing the list of parameters.

listProcesses	<i>List All Processes</i>
---------------	---------------------------

Description

Fetches all existing processes from the API.

Usage

```
listProcesses()
```

Value

A JSON object containing the list of processes.

loadFile	<i>Load or download a file from a process and file name</i>
----------	---

Description

Load or download a file from a process and file name

Usage

```
loadFile(json_data, processName, fileName, sep = "\t", download_dir = getwd())
```

Arguments

json_data	The 'JSON' object containing the report data.
processName	The name of the process.
fileName	The name of the file to load or download.
sep	The separator for tabular files. Default is tab-separated.
download_dir	The directory where non-tabular files will be downloaded.

Value

A data frame with the file contents if the file is tabular; otherwise, NULL after downloading the file.

load_config	<i>Load Configuration</i>
-------------	---------------------------

Description

Loads the configuration file or triggers authentication if the file is missing.

Usage

```
load_config()
```

Value

A list containing the host name and bearer token.

login	<i>Login and retrieve the cookie token</i>
-------	--

Description

Login and retrieve the cookie token

Usage

```
login(  
  hostname,  
  username,  
  password,  
  identity_type = 1,  
  redirect_uri = "http://localhost"  
)
```

Arguments

hostname	The API url
username	The login username.
password	The login password.
identity_type	The identity type.
redirect_uri	The redirect URI.

Value

The cookie token.

prepareSessionHistory	<i>Prepare Session History</i>
-----------------------	--------------------------------

Description

Saves the current R session history to a timestamped 'R history' file.

Usage

```
prepareSessionHistory()
```

Value

The file path of the saved session history file.

Examples

```
## Not run:
  history_file <- prepareSessionHistory()
  print(history_file)

## End(Not run)
```

searchCanvas	<i>Search canvas</i>
--------------	----------------------

Description

Search canvas

Usage

```
searchCanvas(searchParams = list(filter = list()))
```

Arguments

searchParams A named list of search filters.

Value

A JSON object with canvas search results.

Examples

```
## Not run:
search_params <- list(
  filter = list(
    name="new_canvas"
  )
)
canvas <- searchCanvas(search_params)
print(canvas)

## End(Not run)
```

searchCollections	<i>Search Collections</i>
-------------------	---------------------------

Description

Search Collections

Usage

```
searchCollections(searchParams = list(filter = list()))
```

Arguments

searchParams A named list of search filters.

Value

A JSON object with collection search results.

searchData	<i>Search Collection Data</i>
------------	-------------------------------

Description

Searches entries in a metadata collection.

Usage

```
searchData(canvasID, collectionName, filterData = list(filter = list()))
```

Arguments

canvasID The canvas ID.
collectionName The name of the collection.
filterData Optional filters.

Value

A JSON object of matching data entries.

searchDatasetFiles	<i>Search Dataset Files</i>
--------------------	-----------------------------

Description

Lists files associated with a dataset.

Usage

```
searchDatasetFiles(datasetID, filterData = list(filter = list()))
```

Arguments

- datasetID The ID of the dataset.
- filterData A named list of filter parameters.

Value

A JSON object of matching files.

searchFields	<i>Search Fields</i>
--------------	----------------------

Description

Search Fields

Usage

```
searchFields(searchParams = list(filter = list()))
```

Arguments

- searchParams A named list of search filters.

Value

A JSON object with field search results.

updateCanvas	<i>Update canvas</i>
--------------	----------------------

Description

Update canvas

Usage

updateCanvas(canvasID, updateData)

Arguments

- | | |
|------------|------------------------------------|
| canvasID | A character string with canvas ID. |
| updateData | A named list of canvas updates. |

Value

A JSON object with updated canvas.

updateCollection	<i>Update Collection</i>
------------------	--------------------------

Description

Update Collection

Usage

updateCollection(collectionID, updateData)

Arguments

- | | |
|--------------|--|
| collectionID | A character string with collection ID. |
| updateData | A named list of collection updates. |

Value

A JSON object with updated collection.

updateData	<i>Update Data Entry</i>
------------	--------------------------

Description

Updates an existing data record.

Usage

```
updateData(canvasID, collectionName, dataID, updateData)
```

Arguments

- canvasID The canvas ID.
- collectionName The collection name.
- dataID The ID of the data to update.
- updateData A named list of updates.

Value

A JSON object with updated entry.

updateField	<i>Update Field</i>
-------------	---------------------

Description

Update Field

Usage

```
updateField(fieldID, updateData)
```

Arguments

- fieldID A character string with field ID.
- updateData A named list of field updates.

Value

A JSON object with updated field.

updateMenuGroup	<i>Update Menu Group</i>
-----------------	--------------------------

Description

Updates the name of an existing menu group.

Usage

updateMenuGroup(menuGroupID, name)

Arguments

- | | |
|-------------|-------------------------------------|
| menuGroupID | The ID of the menu group to update. |
| name | The new name of the menu group. |

Value

A JSON object containing the updated menu group details.

updateParameter	<i>Update a Parameter</i>
-----------------	---------------------------

Description

Updates an existing parameter with the provided data.

Usage

updateParameter(parameterID, parameterData)

Arguments

- | | |
|---------------|---|
| parameterID | The ID of the parameter to update. |
| parameterData | A list containing the updated parameter data. |

Value

A JSON object containing the updated parameter details.

updateProcess	<i>Update a Process</i>
---------------	-------------------------

Description

Updates an existing process with the given data.

Usage

```
updateProcess(processID, processData)
```

Arguments

processID	The ID of the process to update.
processData	A list containing the updated process data.

Value

A JSON object containing the updated process details.

uploadReportFile	<i>Upload a file to a specific report</i>
------------------	---

Description

Upload a file to a specific report

Usage

```
uploadReportFile(report_id, file_path, dir = NULL)
```

Arguments

report_id	The ID of the report for the API.
file_path	The local path to the file being uploaded.
dir	An optional directory name for organizing files.

Value

A list containing the server response.

uploadSessionHistory	<i>Upload Session History</i>
----------------------	-------------------------------

Description

Uploads the current session history to a specified report.

Usage

```
uploadSessionHistory(report_id, dir = NULL)
```

Arguments

report_id	The ID of the report.
dir	The directory name for organizing session history files on the server. Defaults to 'NULL'.

Value

The parsed server response.

Examples

```
## Not run:
response <- uploadSessionHistory(
  report_id = "12345",
  dir = "session_logs",
)
print(response)

## End(Not run)
```

Index

[addFilesToDataset, 3](#)
[authenticate, 3](#)

[calculate_expiration_date, 4](#)
[call_endpoint, 5](#)
[checkProcessUsage, 5](#)
[createCanvas, 6](#)
[createCollection, 7](#)
[createField, 7](#)
[createMenuGroup, 8](#)
[createParameter, 8](#)
[createProcess, 9](#)
[createProcessConfig, 9](#)

[deleteCanvas, 10](#)
[deleteCollection, 11](#)
[deleteData, 11](#)
[deleteField, 12](#)
[deleteParameter, 12](#)
[deleteProcess, 13](#)
[discover, 13](#)
[duplicateProcess, 14](#)

[fetchReportData, 14](#)
[filterParameters, 15](#)

[get_api_status, 23](#)
[get_bearer_token, 24](#)
[get_headers, 24](#)
[getAllFileNames, 15](#)
[getAllReportPaths, 16](#)
[getCanvas, 16](#)
[getCanvasFields, 17](#)
[getCollection, 17](#)
[getCollectionFields, 18](#)
[getData, 18](#)
[getField, 19](#)
[getFieldsForCollection, 19](#)
[getFileNames, 20](#)
[getMenuGroupByName, 20](#)

[getPipelineParameters, 21](#)
[getProcess, 21](#)
[getProcessNames, 22](#)
[getProcessRevisions, 22](#)
[getReportDirs, 23](#)

[listMenuGroups, 25](#)
[listParameters, 25](#)
[listProcesses, 25](#)
[load_config, 26](#)
[loadFile, 26](#)
[login, 27](#)

[prepareSessionHistory, 27](#)

[searchCanvas, 28](#)
[searchCollections, 29](#)
[searchData, 29](#)
[searchDatasetFiles, 30](#)
[searchFields, 30](#)

[updateCanvas, 31](#)
[updateCollection, 31](#)
[updateData, 32](#)
[updateField, 32](#)
[updateMenuGroup, 33](#)
[updateParameter, 33](#)
[updateProcess, 34](#)
[uploadReportFile, 34](#)
[uploadSessionHistory, 35](#)