Package 'shroomDK'

July 23, 2025

Type Package

Title Accessing the Flipside Crypto ShroomDK API

Version 0.3.0

Author Carlos Mercado

Maintainer Carlos Mercado <carlos.mercado@flipsidecrypto.com>

Description Programmatic access to Flipside Crypto data via the Compass RPC API: <https://api-docs.flipsidecrypto.xyz/>. As simple as auto_paginate_query() but with core functions as needed for troubleshooting. Note, 0.1.1 support deprecated 2023-05-31.

Imports jsonlite, httr

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.1

NeedsCompilation no

Repository CRAN

Date/Publication 2024-01-10 15:50:02 UTC

Contents

`auto_paginate_query` *Auto Paginate Queries*

Description

Intelligently grab up to 1 Gigabyte of data from a SQL query including automatic pagination and cleaning.

Usage

```
auto_paginate_query(
  query,
  api_key,
  page_size = 25000,
  page_count = NULL,
  data_source = "snowflake-default",
  data_provider = "flipside",
  api_url = "https://api-v2.flipsidecrypto.xyz/json-rpc"
)
```

Arguments

`query`	The SQL query to pass to ShroomDK
`api_key`	ShroomDK API key.
`page_size`	Default 25,000. May return error if ‘page_size‘ is too large (if page exceeds 30MB or entire query >1GB). Ignored if results fit on 1 page of < 15 Mb of data.
`page_count`	How many pages, of page_size rows each, to read. Default NULL calculates the ceiling (# rows in results / page_size). Ignored if results fit on 1 page of < 15 Mb of data.
`data_source`	Where data is sourced, including specific computation warehouse. Default ‘"snowflake-default"‘. Non default data sources may require registration of api_key to allowlist.
`data_provider`	Who provides data, Default ‘"flipside"‘. Non default data providers may require registration of api_key to allowlist.
`api_url`	default to https://api-v2.flipsidecrypto.xyz/json-rpc but upgradeable for user.

Value

data frame of up to ‘page_size * page_count‘ rows, see ?clean_query for more details on column classes.

Examples

```
## Not run:
pull_data <- auto_paginate_query("
SELECT * FROM ETHEREUM.CORE.FACT_TRANSACTIONS LIMIT 10001",
api_key = readLines("api_key.txt"),
page_size = 9000, # ends up ignored because results fit on 1 page.
page_count = NULL)

## End(Not run)
```

`cancel_query` *Cancel Query*

Description

Uses Flipside ShroomDK to CANCEL a query run id from 'create_query_token()', as the new API uses warehouse-seconds to charge users above the free tier, the ability to cancel is critical for cost management.

Usage

```
cancel_query(
  query_run_id,
  api_key,
  api_url = "https://api-v2.flipsidecrypto.xyz/json-rpc"
)
```

Arguments

`query_run_id`	queryRunId from 'create_query_token()', for token stored as 'x', use 'x$result$queryRequest$queryRunI
`api_key`	Flipside Crypto ShroomDK API Key
`api_url`	default to https://api-v2.flipsidecrypto.xyz/json-rpc but upgradeable for user.

Value

returns a list of the status_canceled (TRUE or FALSE) and the cancel object (which includes related details).

Examples

```
## Not run:
query <- create_query_token("SELECT * FROM ETHEREUM.CORE.FACT_TRANSACTIONS LIMIT 1000000", api_key)
query_status <- get_query_status(query$result$queryRequest$queryRunId, api_key)
canceled <- cancel_query(query$result$queryRequest$queryRunId, api_key)

## End(Not run)
```

clean_query *Clean Query*

Description

Converts query response to data frame while attempting to coerce classes intelligently.

Usage

```
clean_query(request, try_simplify = TRUE)
```

Arguments

request	The request output from get_query_from_token()
try_simplify	because requests can return JSON and may not have the same length across values, they may not be data frame compliant (all columns having the same number of rows). A key example would be TX_JSON in EVM FACT_TRANSACTION tables which include 50+ extra details from transaction logs. But other examples like NULLs in TO_ADDRESS can have similar issues. Default TRUE.

Value

A data frame. If 'try_simplify' is FALSE OR if 'try_simplify' TRUE fails: the data frame is comprised of lists, where each column must be coerced to a desired class (e.g., with 'as.numeric()').

Examples

```
## Not run:
query <- create_query_token("SELECT * FROM ETHEREUM.CORE.FACT_TRANSACTIONS LIMIT 1000", api_key)
request <- get_query_from_token(query$result$queryRequest$queryRunId, api_key)
df1 <- clean_query(request, try_simplify = TRUE)

## End(Not run)
```

create_query_token *Create Query Token*

Description

Uses Flipside ShroomDK to create a Query Token to access Flipside Crypto data. The query token is kept 'ttl' hours and available for no-additional cost reads up to 'mam' minutes (i.e., cached to the same exact result). allowing for pagination and multiple requests before expending more daily request uses.

Usage

```
create_query_token(
  query,
  api_key,
  ttl = 1,
  mam = 10,
  data_source = "snowflake-default",
  data_provider = "flipside",
  api_url = "https://api-v2.flipsidecrypto.xyz/json-rpc"
)
```

Arguments

query	Flipside Crypto Snowflake SQL compatible query as a string.
api_key	Flipside Crypto ShroomDK API Key
ttl	time-to-live (in hours) to keep query results available. Default 1 hour.
mam	max-age-minutes, lifespan of cache. set to 0 to always re-execute. Default 10 minutes.
data_source	Where data is sourced, including specific computation warehouse. Default "snowflake-default". Non default data sources may require registration of api_key to allowlist.
data_provider	Who provides data, Default "flipside". Non default data providers may require registration of api_key to allowlist.
api_url	default to https://api-v2.flipsidecrypto.xyz/json-rpc but upgradeable for user.

Value

list of ‘token‘ and ‘cached‘ use ‘token‘ in ‘get_query_from_token()‘

Examples

```
## Not run:
create_query_token(
query = "SELECT * FROM ethereum.core.fact_transactions LIMIT 33",
api_key = readLines("api_key.txt"),
ttl = 1,
mam = 5
)

## End(Not run)
```

`get_query_from_token` *Get Query From Token*

Description

Uses Flipside ShroomDK to access a Query Token (Run ID). This function is for pagination and multiple requests. It is best suited for debugging and testing new queries. Consider ‘auto_paginate_query()‘ for queries already known to work as expected. Note: To reduce payload it returns a list of outputs (separating column names from rows). See ‘clean_query()‘ for converting result to a data frame.

Usage

```
get_query_from_token(
  query_run_id,
  api_key,
  page_number = 1,
  page_size = 1000,
  result_format = "csv",
  api_url = "https://api-v2.flipsidecrypto.xyz/json-rpc"
)
```

Arguments

`query_run_id`	queryRunId from ‘create_query_token()‘, for token stored as ‘x‘, use ‘x$result$queryRequest$queryRunI
`api_key`	Flipside Crypto ShroomDK API Key
`page_number`	Results are cached, max 30MB of data per page.
`page_size`	Default 1000. Paginate via page_number. May return error if page_size causes data to exceed 30MB.
`result_format`	Default to csv. Options: csv and json.
`api_url`	default to https://api-v2.flipsidecrypto.xyz/json-rpc but upgradeable for user.

Value

returns a list of jsonrpc, id, and result. Within result are: columnNames, columnTypes, rows, page, sql, format, originalQueryRun, redirectedToQueryRun use ‘clean_query()‘ to transform this into a data frame.

Examples

```
## Not run:
query <- create_query_token("SELECT * FROM ETHEREUM.CORE.FACT_TRANSACTIONS LIMIT 1000", api_key)
fact_transactions <- get_query_from_token(query$result$queryRequest$queryRunId, api_key, 1, 1000)

## End(Not run)
```

get_query_status *Get Query ID Status*

Description

Uses Flipside ShroomDK to access the status of a query run id from ‘create_query_token()‘

Usage

```
get_query_status(
  query_run_id,
  api_key,
  api_url = "https://api-v2.flipsidecrypto.xyz/json-rpc"
)
```

Arguments

query_run_id	queryRunId from ‘create_query_token()‘, for token stored as ‘x‘, use ‘x$result$queryRequest$queryRunI
api_key	Flipside Crypto ShroomDK API Key
api_url	default to https://api-v2.flipsidecrypto.xyz/json-rpc but upgradeable for user.

Value

returns request content; for content ‘x‘, use ‘x$result$queryRun$state‘ and ‘x$result$queryRun$errorMessage‘.
Expect one of QUERY_STATE_READY, QUERY_STATE_RUNNING, QUERY_STATE_STREAMING_RESULTS, QUERY_STATE_SUCCESS, QUERY_STATE_FAILED, QUERY_STATE_CANCELED

Examples

```
## Not run:
query = create_query_token("SELECT * FROM ETHEREUM.CORE.FACT_TRANSACTIONS LIMIT 10000", api_key)
get_query_status(query$result$queryRequest$queryRunId, api_key)

## End(Not run)
```

Index