Package 'powerbiR'

October 14, 2022

Type Package

Title An Interface to the 'Power BI REST APIs'

Version 0.1.0

Description Makes it easy to push data to 'Power BI' using R and the 'Power BI REST APIs' (see <https://docs.microsoft.com/en-us/rest/api/power-bi/>). A set of functions for turning data frames into 'Power BI' datasets and refreshing these datasets are provided. Administrative tasks such as monitoring refresh statuses and pulling metadata about workspaces and users are also supported.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

Imports data.table, jsonlite, httr, utils, AzureAuth

Suggests spelling

Depends R (>= 4.2.0)

Language en-US

NeedsCompilation no

Author Christian Vermehren [aut, cre]

Maintainer Christian Vermehren <cv@cantab.net>

Repository CRAN

Date/Publication 2022-08-23 12:50:02 UTC

R topics documented:

`dim_hour` *Demo data: Dim Hour*

Description

A look-up dimension related to Fact Visitors through the hour_key column.

Usage

```
dim_hour
```

Format

A data frame with 24 rows and 2 columns:

hour_key Primary key (unique identifier of hour).

hour Hour as a name (character type).

Source

Anonymized data from google analytics.

`fact_visitors` *Demo data: Fact Visitors*

Description

A fact table showing individual visitors and their transactions on an e-commerce website.

Usage

```
fact_visitors
```

Format

A data frame with 10,033 rows and 5 columns:

visitor_id Unique identifier of the visitor.

transaction_id Unique identifier of the transactions.

revenue The value of the transaction in USD.

timestamp The time of visit in minutes since 1970-01-01.

hour_key Foreign key referring to dim_hour.

Source

Anonymized data from google analytics.

---

pbi_auth *Authenticate to Power BI*

Description

This function authenticates your Power BI session using a service principal that represents an application registered in Azure Active Directory.

Usage

```
pbi_auth(
  tenant = Sys.getenv("PBI_TENANT"),
  app = Sys.getenv("PBI_APP"),
  password = Sys.getenv("PBI_PW")
)
```

Arguments

`tenant`	Your Microsoft tenant ID.
`app`	Your Microsoft app ID.
`password`	Your Microsoft app password (client secret).

Details

The function returns an authentication token invisibly and makes it available to other functions in this package. The token is automatically refreshed upon expiration.

To auto-authenticate, you can specify credentials in environment variables via an `.Renviron` file or using `Sys.setenv` (see example below).

pbi_auth() is a wrapper for AzureAuth::get_azure_token(). Currently, only non-interactive authentication is supported. You therefore need to register an Azure Active Directory service-principal application and obtain tenant ID, app ID and app password (client secret).

For reasons of CRAN policy, the first time AzureAuth is loaded, it will prompt you for permission to create a user-specific directory in order to cache the token. The prompt only appears in an interactive session, not in a batch script. For more details, see AzureAuth.

Value

Returns a token invisibly.

Examples

```
## Not run:

# Basic authentications
pbi_auth(
tenant = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx", # The tenant ID
app = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",    # The app ID
password = "****"                                # The client secret
)

# Using environment variables
Sys.setenv(
  PBI_TENANT = "my_tenant_id",
  PBI_APP = "my_app_id",
  PBI_PW = "my_app_client_secret"
  )

pbi_auth()

## End(Not run)
```

`pbi_dataset_refresh` *Refresh dataset*

Description

Triggers a refresh for the specified dataset from the specified workspace.

Usage

```
pbi_dataset_refresh(group_id, dataset_id)
```

Arguments

`group_id`	The ID of the workspace.
`dataset_id`	The ID of the dataset.

Value

If successful, the refresh request ID is returned.

See Also

pbi_dataset_refresh_hist

Examples

```
## Not run:

group_id <- "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
dataset_id <- "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"

pbi_dataset_refresh(group_id, dataset_id)
#> A refresh of dataset xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx was triggered.
#>
#> To check status, use pbi_dataset_refresh_hist() and the request ID returned
#> by this function.
#> [1] "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"

## End(Not run)
```

`pbi_dataset_refresh_hist`

Refresh history of a dataset

Description

Returns the refresh history for the specified dataset from the specified workspace.

Usage

```
pbi_dataset_refresh_hist(group_id, dataset_id, top = NULL, request_id = NULL)
```

Arguments

`group_id`	The workspace ID
`dataset_id`	The dataset ID
`top`	The number of most recent entries in the refresh history. The default is all available entries.
`request_id`	The request ID returned by pbi_dataset_refresh(). If provided the refresh status of the request ID is returned.

Details

By default the function will return all historical refreshes. You can reduce the list to the most recent refreshes using the `top` argument.

If `request_id` is provided the function will return a single refresh status, but will still query the Power BI API for all historical entries. If you query the top 5 most recent refreshes using the `top` argument, the function will only return a status if the provided request_id is in this list.

The status value return can be either 'Completed', 'Failed' or 'Unknown', which means that the refresh is still in progress.

Value

A data frame with status, start and end times of historical refreshes or a single refresh status message if `request_id` is used.

See Also

pbi_dataset_refresh

Examples

```
## Not run:

group_id <- "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
dataset_id <- "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"

pbi_dataset_refresh_hist(group_id, dataset_id)

## End(Not run)
```

`pbi_delete_dataset` *Delete dataset*

Description

Deletes the specified dataset from the specified workspace. Applicable to push datasets as well as imported datasets.

Usage

```
pbi_delete_dataset(group_id, dataset_id)
```

Arguments

`group_id`	The dataset ID.
`dataset_id`	The workspace ID.

Value

Deletes the entire dataset.

Examples

```
## Not run:

group_id <- "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
dataset_id <- "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"

pbi_delete_dataset(group_id, dataset_id)

## End(Not run)
```

`pbi_delete_rows` *Delete rows*

Description

Deletes all rows from the specified table within the specified dataset from the specified workspace (group ID). Only applicable to push datasets.

Usage

```
pbi_delete_rows(group_id, dataset_id, table_name)
```

Arguments

`group_id`	The Power BI workspace ID.
`dataset_id`	The Power BI dataset ID.
`table_name`	The Power BI table name.

Value

All rows will be deleted from the specified table.

Examples

```
## Not run:

group_id <- "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
dataset_id <- "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
table_name <- "My Table"

pbi_delete_rows(group_id, dataset_id, table_name)

## End(Not run)
```

pbi_list_datasets *Get a list of datasets in a workspace*

Description

Returns the IDs and meta data of all available datasets in the specified Power BI workspace (group ID).

Usage

```
pbi_list_datasets(group_id)
```

Arguments

group_id	The Power BI workspace ID.

Value

A data.table / data frame with dataset information.

Examples

```
## Not run:

group_id <- "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"

pbi_list_datasets(group_id)

## End(Not run)
```

pbi_list_groups *Get a list of workspaces*

Description

Returns the ids and meta data of all Power BI workspaces to which the service principal app has been granted access.

Usage

```
pbi_list_groups()
```

Value

A data frame with workspaces.

Examples

```
## Not run:

pbi_list_groups()

## End(Not run)
```

---

pbi_push_dataset_schema

Push a dataset schema to Power BI

Description

Pushes a dataset schema to the specified Power BI workspace. To add rows to the dataset, use pbi_push_rows().

Usage

```
pbi_push_dataset_schema(schema, group_id, retention = c("none", "basicFIFO"))
```

Arguments

schema	A push-dataset schema created by pbi_schema_create().
group_id	The ID of the destination Power BI workspace.
retention	The retention policy of the dataset. Default is "none".

Value

A dataset with tables will be created in the specified Power BI workspace.

Examples

```
## Not run:

group_id <- "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"

schema <- pbi_schema_create(
  dt_list = list(iris),
  dataset_name = "The iris dataset",
  table_name_list = list(iris)
)

pbi_push_dataset_schema(schema, group_id)

## End(Not run)
```

pbi_push_rows *Push rows to a dataset table*

Description

Adds new data rows to the specified table within the specified dataset from the specified Power BI workspace. Only applicable to push datasets.

Usage

```
pbi_push_rows(dt, group_id, dataset_id, table_name, overwrite = FALSE)
```

Arguments

dt	A data frame with rows to be added to the specified Power BI table (table_name). The columns and data types must match the specified table.
group_id	The ID of the destination Power BI workspace.
dataset_id	The ID of the destination Power BI dataset.
table_name	The name of the destination Power BI table.
overwrite	If TRUE, existing rows will be deleted prior to adding new rows. If FALSE, the new rows will be appended to the existing rows.

Details

The Power BI REST has a limit of 10K rows per POST rows request. This limit is handled by splitting the data frame into chunks of 10K rows each and pushing these chunks one at a time. However, you should manually observe the other limitations of the API. See https://docs.microsoft.com/en-au/rest/api/power-bi/ for more details.

Value

A dataset with tables and optionally defined relationships will be created in the specified Power BI workspace.

Examples

```
## Not run:

group_id <- "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
dataset_id <- "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"

pbi_push_rows(group_id, dataset_id, "My table")

## End(Not run)
```

pbi_schema_create *Create a Power BI dataset schema*

Description

Creates a Power BI dataset schema from a set of data frames. Columns and data types will be inferred from the data frames. Only applicable to push datasets.

Usage

```
pbi_schema_create(
  dt_list,
  dataset_name = "My Power BI Dataset",
  table_name_list,
  relations_list = NULL,
  date_format = "yyyy-mm-dd",
  integer_format = "#,###0",
  double_format = "#,###.00",
  sort_by_col = NULL,
  hidden_col = NULL,
  default_mode = c("Push", "Streaming", "PushStreaming", "AsOnPrem", "AsAzure")
)
```

Arguments

dt_list	A list of data frames which the schema should be inferred from.
dataset_name	A custom name of the Power BI dataset.
table_name_list	A list of custom names corresponding to the list of data frames.
relations_list	A list of relation definitions returned by pbi_schema_relation_create()
date_format	The format of date columns (if any). Default is 'yyyy-mm-dd'.
integer_format	The format of integer columns (if any). Default is '#,###0'.
double_format	The format of double columns (if any). Default is '#,###.00'.
sort_by_col	A list of lists of column-sorting definitions. The inner lists must include elements named 'table', 'sort' and 'sort_by'. See example for more details.
hidden_col	A list of lists columns to be hidden. The inner lists must include elements named 'table' and 'hidden'. See examples for more details.
default_mode	The dataset mode or type. Defaults to 'Push'.

Value

A list with schema properties.

Examples

```
# Load package
library(powerbiR)

# Use data from the powerbiR package
data(dim_hour)
data(fact_visitors)

# Define dataset and its tables
table_list <- list(fact_visitors, dim_hour)
table_names  <- c("visitors", "hour")
dataset_name <- c("Online Visitors")

# Define relations between tables
relation <- pbi_schema_relation_create(
  from_table = "visitors",
  from_column = "hour_key",
  to_table = "hour"
)

# Define sorting behavior of columns in the hour table
sortlist = list(
  table = c("hour"),
  sort = c("hour"),
  sort_by = c("hour_key")
)

# Hide hour_key in the hour and visitors tables
hidden <- list(
  list(
    table = c("hour"),
    hidden = c("hour_key")
  ),
  list(
    table = c("visitors"),
    hidden = c("hour_key", "visitor_id")
  )
)

# Create schema
schema <- pbi_schema_create(
  dt_list = table_list,
  dataset_name = dataset_name,
  table_name_list = table_names,
  relations_list = list(relation),
  sort_by_col = list(sortlist),
  hidden_col = hidden
)
```

`pbi_schema_relation_create` *Define table relationship*

Description

Defines a relationship between tables in a Power BI push dataset. To add this definition to a Power BI dataset schema, use pbi_schema_add_relations().

Usage

```
pbi_schema_relation_create(
  from_table = NULL,
  from_column = NULL,
  to_table = NULL,
  to_column = from_column,
  direction = c("OneDirection", "BothDirections", "Automatic"),
  name = paste0(from_table, to_table, from_column)
)
```

Arguments

`from_table`	The name of the foreign key table
`from_column`	The name of the foreign key column
`to_table`	The name of the primary key table
`to_column`	The name of the primary key column. Defaults to from_column
`direction`	The filter direction of the relationship. Defaults to 'OneDirection'
`name`	The relationship name and identifier. Defaults to a concatenation of from_table, to_table and from_column

Value

A data.table

Examples

```
# An example
```

Index