Package ‘fabricQueryR’

September 8, 2025

Title Query Data in 'Microsoft Fabric'

Version 0.1.1

Description Query data hosted in 'Microsoft Fabric'. Provides helpers to open 'DBI' connections to 'SQL' endpoints of 'Lakehouse' and 'Data Warehouse' items; submit 'Data Analysis Expressions' ('DAX') queries to semantic model datasets in 'Microsoft Fabric' and 'Power BI'; and read 'Delta Lake' tables stored in 'OneLake' ('Azure Data Lake Storage Gen2').

License MIT + file LICENSE

Suggests DBI, odbc, AzureStor, jsonlite, readr, fs, arrow, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

Imports AzureAuth, dplyr, httr2, purrr, rlang, tibble, utils, cli, stringr

URL `https://github.com/kennispunttwente/fabricQueryR`,
`https://kennispunttwente.github.io/fabricQueryR/`

BugReports `https://github.com/kennispunttwente/fabricQueryR/issues`

Depends R (>= 4.1.0)

NeedsCompilation no

Author Luka Koning [aut, cre, cph],
Kennispunt Twente [fnd]

Maintainer Luka Koning <`l.koning@kennispunttwente.nl`>

Repository CRAN

Date/Publication 2025-09-08 19:10:08 UTC

Contents

`fabric_onelake_read_delta_table` *Read a Microsoft Fabric/OneLake Delta table (ADLS Gen2)*

Description

Authenticates to OneLake (ADLS Gen2), resolves the table's `_delta_log` to determine the *current* active Parquet parts, downloads only those parts to a local staging directory, and returns the result as a tibble.

Usage

```
fabric_onelake_read_delta_table(
  table_path,
  workspace_name,
  lakehouse_name,
  tenant_id = Sys.getenv("FABRICQUERYR_TENANT_ID"),
  client_id = Sys.getenv("FABRICQUERYR_CLIENT_ID", unset =
    "04b07795-8ddb-461a-bbee-02f9e1bf7b46"),
  dest_dir = NULL,
  verbose = TRUE,
  dfs_base = "https://onelake.dfs.fabric.microsoft.com"
)
```

Arguments

`table_path`	Character. Table name or nested path (e.g. `"Patienten"` or `"Patienten/patienten_hash"`). Only the last path segment is used as the table directory under `Tables/`.
`workspace_name`	Character. Fabric workspace display name or GUID (this is the ADLS filesystem/container name).
`lakehouse_name`	Character. Lakehouse item name, with or without the `.Lakehouse` suffix (e.g. `"Lakehouse"` or `"Lakehouse.Lakehouse"`).
`tenant_id`	Character. Entra ID (Azure AD) tenant GUID. Defaults to `Sys.getenv("FABRICQUERYR_TENANT_ID")` if missing.
`client_id`	Character. App registration (client) ID. Defaults to `Sys.getenv("FABRICQUERYR_CLIENT_ID")`, falling back to the Azure CLI app id `"04b07795-8ddb-461a-bbee-02f9e1bf7b46"` if not set.

`dest_dir`	Character or `NULL`. Local staging directory for Parquet parts. If `NULL` (default), a temp dir is used and cleaned up on exit.
`verbose`	Logical. Print progress messages via `{cli}`. Default `TRUE`.
`dfs_base`	Character. OneLake DFS endpoint. Default `"https://onelake.dfs.fabric.microsoft.com"`.

Details

- In Microsoft Fabric, OneLake exposes each workspace as an ADLS Gen2 filesystem. Within a Lakehouse item, Delta tables are stored under `Tables/<table>` with a `_delta_log/` directory that tracks commit state. This helper replays the JSON commits to avoid double-counting compacted/removed files.
- Ensure the account/principal you authenticate with has access via **Lakehouse -> Manage OneLake data access** (or is a member of the workspace).
- **AzureAuth** is used to acquire the token. Be wary of caching behavior; you may want to call `AzureAuth::clean_token_directory()` to clear cached tokens if you run into issues

Value

A tibble with the table's current rows (0 rows if the table is empty).

Examples

```
# Example is not executed since it requires configured credentials for Fabric
## Not run:
df <- fabric_onelake_read_delta_table(
  table_path     = "Patients/PatientInfo",
  workspace_name = "PatientsWorkspace",
  lakehouse_name = "Lakehouse.Lakehouse",
  tenant_id      = Sys.getenv("FABRICQUERYR_TENANT_ID"),
  client_id      = Sys.getenv("FABRICQUERYR_CLIENT_ID")
)
dplyr::glimpse(df)

## End(Not run)
```

`fabric_pbi_dax_query` *Query a Microsoft Fabric/Power Bi semantic model with DAX*

Description

High-level helper that authenticates against Azure AD, resolves the workspace & dataset from a Power BI (Microsoft Fabric) XMLA/connection string, executes a DAX statement via the Power BI REST API, and returns a tibble with the resulting data.

Usage

```
fabric_pbi_dax_query(
  connstr,
  dax,
  tenant_id = Sys.getenv("FABRICQUERYR_TENANT_ID"),
  client_id = Sys.getenv("FABRICQUERYR_CLIENT_ID", unset =
    "04b07795-8ddb-461a-bbee-02f9e1bf7b46"),
  include_nulls = TRUE,
  api_base = "https://api.powerbi.com/v1.0/myorg"
)
```

Arguments

`connstr`	Character. Power BI connection string, e.g. `"Data Source=powerbi://api.powerbi.com/v1.0/myorg/` `Catalog=Dataset;"`. The function accepts either `Data Source=` and `Initial Catalog=` parts, or a bare `powerbi://...` for the data source plus a `Dataset=/Catalog=/Initial Catalog=` key (see details).
`dax`	Character scalar with a valid DAX query (see example).
`tenant_id`	Microsoft Azure tenant ID. Defaults to `Sys.getenv("FABRICQUERYR_TENANT_ID")` if missing.
`client_id`	Microsoft Azure application (client) ID used to authenticate. Defaults to `Sys.getenv("FABRICQUERYR_CL` You may be able to use the Azure CLI app id `"04b07795-8ddb-461a-bbee-02f9e1bf7b46"`, but may want to make your own app registration in your tenant for better control.
`include_nulls`	Logical; pass-through to the REST serializer setting. Defaults to TRUE. If TRUE, null values are included in the response; if FALSE, they are omitted.
`api_base`	API base URL. Defaults to "https://api.powerbi.com/v1.0/myorg". 'myorg' is appropriate for most use cases and does not necessarily need to be changed.

Details

- In Microsoft Fabric/Power BI, you can find and copy the connection string by going to a 'Semantic model' item, then go to 'File' -> 'Settings' -> 'Server settings'. Ensure that the account you use to authenticate has access to the workspace, or has been granted 'Build' permissions on the dataset (via sharing).
- **AzureAuth** is used to acquire the token. Be wary of caching behavior; you may want to call `AzureAuth::clean_token_directory()` to clear cached tokens if you run into issues

Value

A tibble with the query result (0 rows if the DAX query returned no rows).

Examples

```
# Example is not executed since it requires configured credentials for Fabric
## Not run:
conn <- "Data Source=powerbi://api.powerbi.com/v1.0/myorg/My Workspace;Initial Catalog=SalesModel;"
df <- fabric_pbi_dax_query(
  connstr = conn,
  dax = "EVALUATE TOPN(1000, 'Customers')",
  tenant_id = Sys.getenv("FABRICQUERYR_TENANT_ID"),
  client_id = Sys.getenv("FABRICQUERYR_CLIENT_ID")
)
dplyr::glimpse(df)

## End(Not run)
```

---

fabric_sql_connect *Connect to a Microsoft Fabric SQL endpoint*

Description

Opens a DBI/ODBC connection to a Microsoft Fabric **Data Warehouse** or **Lakehouse SQL endpoint**, authenticating with Azure AD (MSAL v2) and passing an access token to the ODBC driver.

Usage

```
fabric_sql_connect(
  server,
  database = "Lakehouse",
  tenant_id = Sys.getenv("FABRICQUERYR_TENANT_ID"),
  client_id = Sys.getenv("FABRICQUERYR_CLIENT_ID", unset =
    "04b07795-8ddb-461a-bbee-02f9e1bf7b46"),
  access_token = NULL,
 odbc_driver = getOption("fabricqueryr.sql.driver", "ODBC Driver 18 for SQL Server"),
  port = 1433L,
  encrypt = "yes",
  trust_server_certificate = "no",
  timeout = 30L,
  verbose = TRUE,
  ...
)
```

Arguments

server	Character. Microsoft Fabric SQL connection string or `Server=...` string (see details).
database	Character. Database name. Defaults to `"Lakehouse"`.
tenant_id	Character. Entra ID (AAD) tenant GUID. Defaults to `Sys.getenv("FABRICQUERYR_TENANT_ID")`.
client_id	Character. App registration (client) ID. Defaults to `Sys.getenv("FABRICQUERYR_CLIENT_ID")`, falling back to the Azure CLI app id `"04b07795-8ddb-461a-bbee-02f9e1bf7b46"` if unset.
access_token	Optional character. If supplied, use this bearer token instead of acquiring a new one via `{AzureAuth}`.

`odbc_driver`	Character. ODBC driver name. Defaults to `getOption("fabricqueryr.sql.driver", "ODBC Driver 18 for SQL Server")`.
`port`	Integer. TCP port (default 1433).
`encrypt, trust_server_certificate`	Character flags passed to ODBC. Defaults `"yes"` and `"no"`, respectively.
`timeout`	Integer. Login/connect timeout in seconds. Default 30.
`verbose`	Logical. Emit progress via `{cli}`. Default `TRUE`.
`...`	Additional arguments forwarded to `DBI::dbConnect()`.

Details

- `server` is the Microsoft Fabric SQL connection string, e.g. `"xxxx.datawarehouse.fabric.microsoft.com"`. You can find this by going to your **Lakehouse** or **Data Warehouse** item, then **Settings** -> **SQL analytics endpoint** -> **SQL connection string**. You may also pass a DSN-less `Server=...` string; it will be normalized.
- By default we request a token for `https://database.windows.net/.default`.
- **AzureAuth** is used to acquire the token. Be wary of caching behavior; you may want to call `AzureAuth::clean_token_directory()` to clear cached tokens if you run into issues

Value

A live `DBIConnection` object.

Examples

```
# Example is not executed since it requires configured credentials for Fabric
## Not run:
con <- fabric_sql_connect(
  server    = "2gxz...qiy.datawarehouse.fabric.microsoft.com",
  database  = "Lakehouse",
  tenant_id = Sys.getenv("FABRICQUERYR_TENANT_ID"),
  client_id = Sys.getenv("FABRICQUERYR_CLIENT_ID")
)

# List databases
DBI::dbGetQuery(con, "SELECT name FROM sys.databases")

# List tables
DBI::dbGetQuery(con, "
 SELECT TABLE_SCHEMA, TABLE_NAME
 FROM INFORMATION_SCHEMA.TABLES
 WHERE TABLE_TYPE = 'BASE TABLE'
")

# Get a table
df <- DBI::dbReadTable(con, "Customers")
dplyr::glimpse(df)

DBI::dbDisconnect(con)
```

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

`fabric_sql_query` *Run a SQL query against a Microsoft Fabric SQL endpoint (opening & closing connection)*

Description

Convenience wrapper that opens a connection with `fabric_sql_connect()`, executes `sql`, and returns a tibble. The connection is closed on exit.

Usage

```
fabric_sql_query(
  server,
  sql,
  database = "Lakehouse",
  tenant_id = Sys.getenv("FABRICQUERYR_TENANT_ID"),
  client_id = Sys.getenv("FABRICQUERYR_CLIENT_ID", unset =
    "04b07795-8ddb-461a-bbee-02f9e1bf7b46"),
  access_token = NULL,
 odbc_driver = getOption("fabricqueryr.sql.driver", "ODBC Driver 18 for SQL Server"),
  port = 1433L,
  encrypt = "yes",
  trust_server_certificate = "no",
  timeout = 30L,
  verbose = TRUE,
  ...
)
```

Arguments

`server`	Character. Microsoft Fabric SQL connection string or `Server=...` string (see details).
`sql`	Character scalar. The SQL to run.
`database`	Character. Database name. Defaults to `"Lakehouse"`.
`tenant_id`	Character. Entra ID (AAD) tenant GUID. Defaults to `Sys.getenv("FABRICQUERYR_TENANT_ID")`.
`client_id`	Character. App registration (client) ID. Defaults to `Sys.getenv("FABRICQUERYR_CLIENT_ID")`, falling back to the Azure CLI app id `"04b07795-8ddb-461a-bbee-02f9e1bf7b46"` if unset.
`access_token`	Optional character. If supplied, use this bearer token instead of acquiring a new one via `{AzureAuth}`.
`odbc_driver`	Character. ODBC driver name. Defaults to `getOption("fabricqueryr.sql.driver", "ODBC Driver 18 for SQL Server")`.

`port`	Integer. TCP port (default 1433).
`encrypt`, `trust_server_certificate`	Character flags passed to ODBC. Defaults `"yes"` and `"no"`, respectively.
`timeout`	Integer. Login/connect timeout in seconds. Default 30.
`verbose`	Logical. Emit progress via `{cli}`. Default `TRUE`.
`...`	Additional arguments forwarded to `DBI::dbConnect()`.

Value

A tibble with the query results (0 rows if none).

Examples

```
# Example is not executed since it requires configured credentials for Fabric
## Not run:
df <- fabric_sql_query(
  server    = "2gxz...qiy.datawarehouse.fabric.microsoft.com",
  database  = "Lakehouse",
  sql       = "SELECT TOP 100 * FROM sys.objects",
  tenant_id = Sys.getenv("FABRICQUERYR_TENANT_ID"),
  client_id = Sys.getenv("FABRICQUERYR_CLIENT_ID")
)
dplyr::glimpse(df)

## End(Not run)
```

Index