Package: qiverse.azure (via r-universe)

April 2, 2025

Title PowerBI, Sharepoint and Snowflake Connectivity

Version 0.0.1.0

Description Defines functions needed to generate Azure access tokens,
connect to PowerBI dataflows/datasets, Sharepoint and
Snowflake.

License GPL (>= 3)

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

URL `https://github.com/AUS-DOH-Safety-and-Quality/qiverse/qiverse.azure`

BugReports `https://github.com/AUS-DOH-Safety-and-Quality/qiverse/issues`

Imports httr, jsonlite, AzureKeyVault, AzureAuth, utils

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/pak/sysreqs libssl-dev

Repository https://aus-doh-safety-and-quality.r-universe.dev

RemoteUrl https://github.com/AUS-DOH-Safety-and-Quality/qiverse

RemoteRef main

RemoteSha 85fe2406a27c567f5bc0911477e1e2a3f9847502

RemoteSubdir qiverse.azure

Contents

`.init_key_vault` *Initialise an AzureKeyVault object for interacting with stored secrets. If an AzureAuth access token object is not provided then the authentication process is also initiated*

Description

Initialise an AzureKeyVault object for interacting with stored secrets. If an AzureAuth access token object is not provided then the authentication process is also initiated

Usage

```
.init_key_vault(vault_name, token_object)
```

Arguments

`vault_name`	The name of the target Azure Key Vault
`token_object`	A previously-generated AzureAuth token object created for the Key Vault resource

Value

An AzureKeyVault::key_vault() object

See Also

Other Key Vault methods: `.secrets_operation()`, `kv_delete_secret()`, `kv_get_secret()`, `kv_list_secrets()`, `kv_write_secret()`

Examples

```
-
```

`.secrets_operation` *A generic implementation function for interacting with Azure Key Vault secrets. Handles the connection and authentication (if necessary) to the Key Vault service*

Description

A generic implementation function for interacting with Azure Key Vault secrets. Handles the connection and authentication (if necessary) to the Key Vault service

Usage

```
.secrets_operation(vault_name, operation, token_object, ...)
```

Arguments

`vault_name`	The name of the target Azure Key Vault
`operation`	The name of the operation to be applied, see the AzureKeyVault::secrets documentation for a full list of possible operations
`token_object`	(optional) A previously-generated AzureAuth token object created for the Key Vault resource
`...`	Any arguments to be passed to the Key Vault operation, see the AzureKeyVault::secrets documentation for the possible arguments for each operation

Value

An AzureKeyVault::key_vault() object

See Also

Other Key Vault methods: `.init_key_vault()`, `kv_delete_secret()`, `kv_get_secret()`, `kv_list_secrets()`, `kv_write_secret()`

Examples

```
-
```

`az_authenticated_api_query` *Make a call to a web API which requires an Azure access token for authentication*

Description

Make a call to a web API which requires an Azure access token for authentication

Usage

```
az_authenticated_api_query(method, url, access_token, ...)
```

Arguments

`method`	The HTTP method to call (e.g., "POST", "GET", etc.)
`url`	The web URL of the API to be called
`access_token`	The azure access token string which will be used for authentication
`...`	Any additional arguments to be passed to the method call. See the httr documentation for the intended method for valid arguments

Value

The result of the API call

See Also

Other Azure methods: `db_secret_scopes_api()`, `db_secrets_api()`, `store_databricks_access_token()`

Examples

```
-
```

`db_secrets_api` *Interact with the Databricks API for managing individual secrets. The Databricks API allows for listing, creating, and deleting secret scopes.*

Description

Interact with the Databricks API for managing individual secrets. The Databricks API allows for listing, creating, and deleting secret scopes.

Usage

```
db_secrets_api(
  operation,
  workspace_url,
  access_token,
  scope_name,
  secret_name = NULL,
  secret_value = NULL,
  bytestring = FALSE
)
```

Arguments

`operation`	The scopes operation to apply ("list", "put", or "delete")
`workspace_url`	The url of the target databricks workspace
`access_token`	The azure access token string which will be used for authentication
`scope_name`	The name of the scope containing the target secret
`secret_name`	The name of the target secret
`secret_value`	The value to assign to the target secret
`bytestring`	Whether the target secret is storing a string of bytes or characters

Value

The result of the API call

See Also

Other Azure methods: `az_authenticated_api_query()`, `db_secret_scopes_api()`, `store_databricks_access_token`

Examples

```
-
```

`db_secret_scopes_api` *Interact with the Databricks API for managing secret scopes (collections of secrets). The Databricks API allows for listing, creating, and deleting secret scopes.*

Description

Interact with the Databricks API for managing secret scopes (collections of secrets). The Databricks API allows for listing, creating, and deleting secret scopes.

Usage

```
db_secret_scopes_api(operation, workspace_url, access_token, scope_name = NULL)
```

Arguments

`operation`	The scopes operation to apply ("list", "create", or "delete")
`workspace_url`	The url of the target databricks workspace
`access_token`	The azure access token string which will be used for authentication
`scope_name`	If creating or deleting a secret scope, the name of the target scope

Value

The result of the API call

See Also

Other Azure methods: `az_authenticated_api_query()`, `db_secrets_api()`, `store_databricks_access_token()`

Examples

```
-
```

`get_az_tk` *Generate an Azure authentication token*

Description

Generate an Azure authentication token

Usage

```
get_az_tk(
  token_type,
  tenant_id = Sys.getenv("az_tenant_id"),
  app_id_pbi_df = Sys.getenv("az_app_id_pbi_dataflow"),
  app_id_pbi_ds = Sys.getenv("az_app_id_pbi_dataset"),
  graph_api_shp = Sys.getenv("az_graph_api_sharepoint"),
  app_id_shp = Sys.getenv("az_app_id_sharepoint"),
  cli_sec_shp = Sys.getenv("az_cli_secret_id_sharepoint"),
  system_type = "local",
  db_scope = "",
  ...
)
```

Arguments

`token_type`	The type of token to be generated. This can be either: "pbi_df" for PowerBI Dataflows, "pbi_ds" for PowerBI datasets, "sp" for SharePoint, "sf" for Snowflake, "databricks" for the Databricks API, "key_vault" for Azure Key Vault
`tenant_id`	Your organisation's tenant identifier
`app_id_pbi_df`	The application identifier with access to PowerBI dataflows
`app_id_pbi_ds`	The application identifier with access to PowerBI datasets
`graph_api_shp`	The Graph API created for access to SharePoint
`app_id_shp`	The application identifier with access to SharePoint
`cli_sec_shp`	The client secret with access to SharePoint
`system_type`	The system where the package is being called from. Can be either "local" if running on your local machine or virtual machine, or "databricks" if running on a databricks instance. The "databricks" option will pull the token from your secret created using qiverse.azure::store_databricks_access_token
`db_scope`	Optional parameter for the databricks scope to pull the Azure access token from
`...`	Additional arguments to be passed to `AzureAuth::get_azure_token()`, #nolint such as 'use_cache' or 'auth_type'

Value

An Azure token to authenticate connections.

Examples

```
## Not run:
tk <- get_az_tk('pbi_df')
tk <- get_az_tk('pbi_ds')
tk <- get_az_tk('sp')
tk <- get_az_tk('pbi_df', auth_type = 'device_code')

## End(Not run)
```

`kv_delete_secret` *Delete a specified secret from a given Key Vault*

Description

Delete a specified secret from a given Key Vault

Usage

```
kv_delete_secret(vault_name, secret_name, confirm = TRUE, token_object = NULL)
```

Arguments

`vault_name`	The name of the target Azure Key Vault
`secret_name`	The name of the target secret to delete
`confirm`	Whether to display an interactive prompt to confirm deletion of stored secret
`token_object`	(optional) A previously-generated AzureAuth token object created for the Key Vault resource

See Also

Other Key Vault methods: `.init_key_vault()`, `.secrets_operation()`, `kv_get_secret()`, `kv_list_secrets()`, `kv_write_secret()`

Examples

```
-
```

`kv_get_secret` *Retrieve the value of a specified secret from a given Key Vault*

Description

Retrieve the value of a specified secret from a given Key Vault

Usage

```
kv_get_secret(vault_name, secret_name, token_object = NULL)
```

Arguments

`vault_name`	The name of the target Azure Key Vault
`secret_name`	The name of the target secret to retrieve
`token_object`	(optional) A previously-generated AzureAuth token object created for the Key Vault resource

Value

The value of a secret stored in the given Key Vault

See Also

Other Key Vault methods: `.init_key_vault()`, `.secrets_operation()`, `kv_delete_secret()`, `kv_list_secrets()`, `kv_write_secret()`

Examples

```
-
```

`kv_list_secrets` *List the names of secrets stored in a given Azure Key Vault*

Description

List the names of secrets stored in a given Azure Key Vault

Usage

```
kv_list_secrets(vault_name, token_object = NULL)
```

Arguments

`vault_name`	The name of the target Azure Key Vault
`token_object`	(optional) A previously-generated AzureAuth token object created for the Key Vault resource

Value

A list of the (names of) secrets stored in the given Key Vault

See Also

Other Key Vault methods: `.init_key_vault()`, `.secrets_operation()`, `kv_delete_secret()`, `kv_get_secret()`, `kv_write_secret()`

Examples

```
-
```

`kv_write_secret` *Create or overwrite a secret in an Azure Key Vault*

Description

Create or overwrite a secret in an Azure Key Vault

Usage

```
kv_write_secret(
  vault_name,
  secret_name,
  secret_value,
  bytestring = FALSE,
  token_object = NULL
)
```

Arguments

`vault_name`	The name of the target Azure Key Vault
`secret_name`	The name of the target secret to update
`secret_value`	The value to set the target secret to
`bytestring`	Whether the secret value is a string of bytes, for storing serialised objects
`token_object`	(optional) A previously-generated AzureAuth token object created for the Key Vault resource

See Also

Other Key Vault methods: `.init_key_vault()`, `.secrets_operation()`, `kv_delete_secret()`, `kv_get_secret()`, `kv_list_secrets()`

Examples

```
-
```

`store_databricks_access_token`
Create an Azure authentication token and store it as a Databricks secret when run on a databricks cluster

---

Description

Create an Azure authentication token and store it as a Databricks secret when run on a databricks cluster

Usage

```
store_databricks_access_token(token, url, username)
```

Arguments

`token`	The Azure access token to access the scopes API, and to be stored as a secret
`url`	The workspace URL of the databricks instance
`username`	The current user's username, for the secret scope

See Also

Other Azure methods: `az_authenticated_api_query()`, `db_secret_scopes_api()`, `db_secrets_api()`

Examples

```
## Not run:
# Set with your tenant_id and app_id. Ensure that this has it's own command
# chunk, so the command will complete after authentication
token <- qiverse.azure::get_az_tk(
  "pbi_df",
  tenant_id = tenant_id,
  app_id_pbi_df = app_id,
  auth_type = "device_code"
)

# Store token as databricks secret
update_secret <- qiverse.azure::store_databricks_access_token(
  token = token,
  url = paste0("https://",
    SparkR::sparkR.conf("spark.databricks.workspaceUrl")),
  user_name =
    SparkR::first(SparkR::sql("SELECT current_user() AS username"))$username
)

# Check whether the HTTP request returned a success code
if(update_secret$status_code == 200) {
  "Token successfully updated"
} else {
  "Error occurred"
}

## End(Not run)
```

Index