Package 'AzureGraph'

October 12, 2020

Title Simple Interface to 'Microsoft Graph'

Version 1.1.2

Description A simple interface to the 'Microsoft Graph' API <https://docs.microsoft.com/en-us/graph/overview>. 'Graph' is a comprehensive framework for accessing data in various online Microsoft services. Currently, this package aims to provide an R interface only to the 'Azure Active Directory' part, with a view to supporting interoperability of R and 'Azure': users, groups, registered apps and service principals. However it can be easily extended to cover other services. Part of the 'AzureR' family of packages.

URL https://github.com/Azure/AzureGraph
https://github.com/Azure/AzureR

BugReports https://github.com/Azure/AzureGraph/issues

License MIT + file LICENSE

VignetteBuilder knitr

Depends R (>= 3.3)

Imports AzureAuth (>= 1.0.1), utils, httr (>= 1.3), jsonlite, openssl, R6

Suggests AzureRMR, knitr, rmarkdown, testthat

RoxygenNote 6.1.1

NeedsCompilation no

Author Hong Ooi [aut, cre], Microsoft [cph]

Maintainer Hong Ooi <hongooi73@gmail.com>

Repository CRAN

Date/Publication 2020-10-12 11:30:06 UTC

R topics documented:

---

az_app *Registered app in Azure Active Directory*

Description

Base class representing an AAD app.

Usage

```
az_app
```

Format

An R6 object of class `az_app`, inheriting from `az_object`.

Fields

- `token`: The token used to authenticate with the Graph host.
- `tenant`: The Azure Active Directory tenant for this app.
- `type`: always "application" for an app object.
- `properties`: The app properties.
- `password`: The app password. Note that the Graph API does not return previously-generated passwords. This field will only be populated for an app object created with `ms_graph$create_app()`, or after a call to the `add_password()` method below.

Methods

- `new(...)`: Initialize a new app object. Do not call this directly; see 'Initialization' below.
- `delete(confirm=TRUE)`: Delete an app. By default, ask for confirmation first.
- `update(...)`: Update the app data in Azure Active Directory. For what properties can be updated, consult the REST API documentation link below.
- `do_operation(...)`: Carry out an arbitrary operation on the app.
- `sync_fields()`: Synchronise the R object with the app data in Azure Active Directory.
- `list_group_memberships()`: Return the IDs of all groups this app is a member of.
- `list_object_memberships()`: Return the IDs of all groups, administrative units and directory roles this app is a member of.

- `list_owners(type=c("user","group","application","servicePrincipal"))`: Return a list of all owners of this app. Specify the `type` argument to filter the result for specific object type(s).
- `create_service_principal(...)`: Create a service principal for this app, by default in the current tenant.
- `get_service_principal()`: Get the service principal for this app.
- `delete_service_principal(confirm=TRUE)`: Delete the service principal for this app. By default, ask for confirmation first.
- `add_password(password_name=NULL,password_duration=NULL)`: Adds a strong password. `password_duration` is the length of time in years that the password remains valid, with default duration 2 years. Returns the ID of the generated password.
- `remove_password(password_id,confirm=TRUE)`: Removes the password with the given ID. By default, ask for confirmation first.
- `add_certificate(certificate)`: Adds a certificate for authentication. This can be specified as the name of a .pfx or .pem file, an `openssl::cert` object, an `AzureKeyVault::stored_cert` object, or a raw or character vector.
- `remove_certificate(certificate_id,confirm=TRUE)`: Removes the certificate with the given ID. By default, ask for confirmation first.

Initialization

Creating new objects of this class should be done via the `create_app` and `get_app` methods of the ms_graph class. Calling the `new()` method for this class only constructs the R object; it does not call the Microsoft Graph API to create the actual app.

Microsoft Graph overview, REST API reference

See Also

ms_graph, az_service_principal, az_user, az_group, az_object

Examples

```
## Not run:

gr <- get_graph_login()
app <- gr$create_app("MyNewApp")

# password resetting: remove the old password, add a new one
pwd_id <- app$properties$passwordCredentials[[1]]$keyId
app$add_password()
app$remove_password(pwd_id)

# set a redirect URI
app$update(publicClient=list(redirectUris=I("http://localhost:1410")))

# add API permission (access Azure Storage as user)
app$update(requiredResourceAccess=list(
    list(
```

```
        resourceAppId="e406a681-f3d4-42a8-90b6-c2b029497af1",
        resourceAccess=list(
            list(
                id="03e0da56-190b-40ad-a80c-ea378c433f7f",
                type="Scope"
            )
        )
    )
))

# add a certificate from a .pem file
app$add_certificate("cert.pem")

# can also read the file into an openssl object, and then add the cert
cert <- openssl::read_cert("cert.pem")
app$add_certificate(cert)

# add a certificate stored in Azure Key Vault
vault <- AzureKeyVault::key_vault("mytenant")
cert2 <- vault$certificates$get("certname")
app$add_certificate(cert2)

# change the app name
app$update(displayName="MyRenamedApp")


## End(Not run)
```

`az_device` *Device in Azure Active Directory*

Description

Base class representing a registered device.

Usage

```
az_device
```

Format

An R6 object of class `az_device`, inheriting from `az_object`.

Fields

- `token`: The token used to authenticate with the Graph host.
- `tenant`: The Azure Active Directory tenant for this group.
- `type`: always "device" for a device object.
- `properties`: The device properties.

Methods

- `new(...)`: Initialize a new device object. Do not call this directly; see 'Initialization' below.
- `delete(confirm=TRUE)`: Delete a device. By default, ask for confirmation first.
- `update(...)`: Update the device information in Azure Active Directory.
- `do_operation(...)`: Carry out an arbitrary operation on the device.
- `sync_fields()`: Synchronise the R object with the app data in Azure Active Directory.
- `list_group_memberships()`: Return the IDs of all groups this device is a member of.
- `list_object_memberships()`: Return the IDs of all groups, administrative units and directory roles this device is a member of.

Initialization

Create objects of this class via the `list_registered_devices()` and `list_owned_devices()` methods of the `az_user` class.

See Also

ms_graph, az_user, az_object

Microsoft Graph overview, REST API reference

`az_group` *Group in Azure Active Directory*

Description

Base class representing an AAD group.

Usage

```
az_group
```

Format

An R6 object of class `az_group`, inheriting from `az_object`.

Fields

- `token`: The token used to authenticate with the Graph host.
- `tenant`: The Azure Active Directory tenant for this group.
- `type`: always "group" for a group object.
- `properties`: The group properties.

Methods

- `new(...)`: Initialize a new group object. Do not call this directly; see 'Initialization' below.
- `delete(confirm=TRUE)`: Delete a group. By default, ask for confirmation first.
- `update(...)`: Update the group information in Azure Active Directory.
- `do_operation(...)`: Carry out an arbitrary operation on the group.
- `sync_fields()`: Synchronise the R object with the app data in Azure Active Directory.
- `list_group_memberships()`: Return the IDs of all groups this group is a member of.
- `list_object_memberships()`: Return the IDs of all groups, administrative units and directory roles this group is a member of.
- `list_members(type=c("user","group","application","servicePrincipal"))`: Return a list of all members of this group. Specify the `type` argument to filter the result for specific object type(s).
- `list_owners(type=c("user","group","application","servicePrincipal"))`: Return a list of all owners of this group. Specify the `type` argument to filter the result for specific object type(s).

Initialization

Creating new objects of this class should be done via the `create_group` and `get_group` methods of the ms_graph and az_app classes. Calling the `new()` method for this class only constructs the R object; it does not call the Microsoft Graph API to create the actual group.

See Also

ms_graph, az_app, az_user, az_object

Microsoft Graph overview, REST API reference

Examples

```
## Not run:

gr <- get_graph_login()
usr <- gr$get_user("myname@aadtenant.com")

grps <- usr$list_direct_memberships()
grp <- grp[[1]]

grp$list_members()
grp$list_owners()


## End(Not run)
```

az_object *Azure Active Directory object*

Description

Base class representing a directory object in Microsoft Graph.

Usage

```
az_object
```

Format

An R6 object of class `az_object`.

Fields

- `token`: The token used to authenticate with the Graph host.
- `tenant`: The Azure Active Directory tenant for this object.
- `type`: The type of object: user, group, application or service principal.
- `properties`: The object properties.

Methods

- `new(...)`: Initialize a new directory object. Do not call this directly; see 'Initialization' below.
- `delete(confirm=TRUE)`: Delete an object. By default, ask for confirmation first.
- `update(...)`: Update the object information in Azure Active Directory.
- `do_operation(...)`: Carry out an arbitrary operation on the object.
- `sync_fields()`: Synchronise the R object with the data in Azure Active Directory.
- `list_group_memberships()`: Return the IDs of all groups this object is a member of.
- `list_object_memberships()`: Return the IDs of all groups, administrative units and directory roles this object is a member of.

Initialization

Objects of this class should not be created directly. Instead, create an object of the appropriate subclass: az_app, az_service_principal, az_user, az_group.

See Also

ms_graph, az_app, az_service_principal, az_user, az_group

Microsoft Graph overview, REST API reference

`az_service_principal` *Service principal in Azure Active Directory*

Description

Base class representing an AAD service principal.

Usage

```
az_service_principal
```

Format

An R6 object of class `az_service_principal`, inheriting from `az_object`.

Fields

- `token`: The token used to authenticate with the Graph host.
- `tenant`: The Azure Active Directory tenant for this service principal.
- `type`: always "service principal" for a service principal object.
- `properties`: The service principal properties.

Methods

- `new(...)`: Initialize a new service principal object. Do not call this directly; see 'Initialization' below.
- `delete(confirm=TRUE)`: Delete a service principal. By default, ask for confirmation first.
- `update(...)`: Update the service principal information in Azure Active Directory.
- `do_operation(...)`: Carry out an arbitrary operation on the service principal.
- `sync_fields()`: Synchronise the R object with the service principal data in Azure Active Directory.
- `list_group_memberships()`: Return the IDs of all groups this service principal is a member of.
- `list_object_memberships()`: Return the IDs of all groups, administrative units and directory roles this service principal is a member of.

Initialization

Creating new objects of this class should be done via the `create_service_principal` and `get_service_principal` methods of the ms_graph and az_app classes. Calling the `new()` method for this class only constructs the R object; it does not call the Microsoft Graph API to create the actual service principal.

See Also

ms_graph, az_app, az_object

Azure Microsoft Graph overview, REST API reference

az_user *User in Azure Active Directory*

Description

Base class representing an AAD user account.

Usage

```
az_user
```

Format

An R6 object of class `az_user`, inheriting from `az_object`.

Fields

- `token`: The token used to authenticate with the Graph host.
- `tenant`: The Azure Active Directory tenant for this user.
- `type`: always "user" for a user object.
- `properties`: The user properties.

Methods

- `new(...)`: Initialize a new user object. Do not call this directly; see 'Initialization' below.
- `delete(confirm=TRUE)`: Delete a user account. By default, ask for confirmation first.
- `update(...)`: Update the user information in Azure Active Directory.
- `do_operation(...)`: Carry out an arbitrary operation on the user account.
- `sync_fields()`: Synchronise the R object with the app data in Azure Active Directory.
- `list_group_memberships()`: Return the IDs of all groups this user is a member of.
- `list_object_memberships()`: Return the IDs of all groups, administrative units and directory roles this user is a member of.
- `list_direct_memberships(id_only=TRUE)`: List the groups this user is a direct member of. Set `id_only=TRUE` to return only a vector of group IDs (the default), or `id_only=FALSE` to return a list of group objects.
- `list_owned_objects(type=c("user","group","application","servicePrincipal"))`: List directory objects (groups/apps/service principals) owned by this user. Specify the `type` argument to filter the result for specific object type(s).
- `list_created_objects(type=c("user","group","application","servicePrincipal"))`: List directory objects (groups/apps/service principals) created by this user. Specify the `type` argument to filter the result for specific object type(s).
- `list_owned_devices()`: List the devices owned by this user.
- `list_registered_devices()`: List the devices registered by this user.
- `reset_password(password=NULL,force_password_change=TRUE)`: Resets a user password. By default the new password will be randomly generated, and must be changed at next login.

Initialization

Creating new objects of this class should be done via the `create_user` and `get_user` methods of the ms_graph and az_app classes. Calling the `new()` method for this class only constructs the R object; it does not call the Microsoft Graph API to create the actual user account.

See Also

ms_graph, az_app, az_group, az_device, az_object

Microsoft Graph overview, REST API reference

Examples

```
## Not run:

gr <- get_graph_login()

# my user account
gr$get_user()

# another user account
usr <- gr$get_user("myname@aadtenant.com")

grps <- usr$list_direct_memberships()
head(grps)

# owned objects
usr$list_owned_objects()

# owned apps and service principals
usr$list_owned_objects(type=c("application", "servicePrincipal"))


## End(Not run)
```

---

call_graph_endpoint *Call the Microsoft Graph REST API*

Description

Call the Microsoft Graph REST API

Usage

```
call_graph_endpoint(token, operation, ..., options = list(),
  api_version = getOption("azure_graph_api_version"))

call_graph_url(token, url, ..., body = NULL, encode = "json",
  http_verb = c("GET", "DELETE", "PUT", "POST", "HEAD", "PATCH"),
  http_status_handler = c("stop", "warn", "message", "pass"),
  auto_refresh = TRUE)
```

Arguments

token	An Azure OAuth token, of class AzureToken.
operation	The operation to perform, which will form part of the URL path.
...	Other arguments passed to lower-level code, ultimately to the appropriate functions in httr.
options	A named list giving the URL query parameters.
api_version	The API version to use, which will form part of the URL sent to the host.
url	A complete URL to send to the host.
body	The body of the request, for PUT/POST/PATCH.
encode	The encoding (really content-type) for the request body. The default value "json" means to serialize a list body into a JSON object. If you pass an already-serialized JSON object as the body, set encode to "raw".
http_verb	The HTTP verb as a string, one of GET, PUT, POST, DELETE, HEAD or PATCH.
http_status_handler	How to handle in R the HTTP status code of a response. "stop", "warn" or "message" will call the appropriate handlers in httr, while "pass" ignores the status code.
auto_refresh	Whether to refresh/renew the OAuth token if it is no longer valid.

Details

These functions form the low-level interface between R and Microsoft Graph. call_graph_endpoint forms a URL from its arguments and passes it to call_graph_url.

Value

If http_status_handler is one of "stop", "warn" or "message", the status code of the response is checked. If an error is not thrown, the parsed content of the response is returned with the status code attached as the "status" attribute.

If http_status_handler is "pass", the entire response is returned without modification.

See Also

httr::GET, httr::PUT, httr::POST, httr::DELETE, httr::stop_for_status, httr::content

---

create_graph_login *Login to Azure Active Directory Graph*

Description

Login to Azure Active Directory Graph

Usage

```
create_graph_login(tenant = "common", app = .az_cli_app_id,
  password = NULL, username = NULL, certificate = NULL,
  auth_type = NULL, host = "https://graph.microsoft.com/",
  aad_host = "https://login.microsoftonline.com/", config_file = NULL,
  token = NULL, ...)

get_graph_login(tenant = "common", selection = NULL, refresh = TRUE)

delete_graph_login(tenant = "common", confirm = TRUE)

list_graph_logins()
```

Arguments

tenant	The Azure Active Directory tenant for which to obtain a login client. Can be a name ("myaadtenant"), a fully qualified domain name ("myaadtenant.onmicrosoft.com" or "mycompanyname.com"), or a GUID. The default is to login via the "common" tenant, which will infer your actual tenant from your credentials.
app	The client/app ID to use to authenticate with Azure Active Directory. The default is to login interactively using the Azure CLI cross-platform app, but you can supply your own app credentials as well.
password	If `auth_type == "client_credentials"`, the app secret; if `auth_type == "resource_owner"`, your account password.
username	If `auth_type == "resource_owner"`, your username.
certificate	If 'auth_type == "client_credentials", a certificate to authenticate with. This is a more secure alternative to using an app secret.
auth_type	The OAuth authentication method to use, one of "client_credentials", "authorization_code", "device_code" or "resource_owner". If `NULL`, this is chosen based on the presence of the `username` and `password` arguments.
host	Your Microsoft Graph host. Defaults to `https://graph.microsoft.com/`. Change this if you are using a government or private cloud.
aad_host	Azure Active Directory host for authentication. Defaults to `https://login.microsoftonline.com/`. Change this if you are using a government or private cloud.
config_file	Optionally, a JSON file containing any of the arguments listed above. Arguments supplied in this file take priority over those supplied on the command line. You can also use the output from the Azure CLI `az ad sp create-for-rbac` command.
token	Optionally, an OAuth 2.0 token, of class AzureAuth::AzureToken. This allows you to reuse the authentication details for an existing session. If supplied, all other arguments to `create_graph_login` will be ignored.
...	Other arguments passed to `ms_graph$new()`.
selection	For `get_graph_login`, if you have multiple logins for a given tenant, which one to use. This can be a number, or the input MD5 hash of the token used for the login. If not supplied, `get_graph_login` will print a menu and ask you to choose a login.

refresh	For get_graph_login, whether to refresh the authentication token on loading the client.
confirm	For delete_graph_login, whether to ask for confirmation before deleting.

Details

create_graph_login creates a login client to authenticate with Microsoft Graph, using the supplied arguments. The authentication token is obtained using get_azure_token, which automatically caches and reuses tokens for subsequent sessions. Note that credentials are only cached if you allowed AzureGraph to create a data directory at package startup.

get_graph_login returns a login client by retrieving previously saved credentials. It searches for saved credentials according to the supplied tenant; if multiple logins are found, it will prompt for you to choose one.

One difference between create_graph_login and get_graph_login is the former will delete any previously saved credentials that match the arguments it was given. You can use this to force AzureGraph to remove obsolete tokens that may be lying around.

Value

For get_graph_login and create_graph_login, an object of class ms_graph, representing the login client. For list_graph_logins, a (possibly nested) list of such objects.

If the AzureR data directory for saving credentials does not exist, get_graph_login will throw an error.

Linux DSVM note

If you are using a Linux Data Science Virtual Machine in Azure, you may have problems running create_graph_login() (ie, without any arguments). In this case, try create_graph_login(auth_type="device_code").

See Also

ms_graph, AzureAuth::get_azure_token for more details on authentication methods

Microsoft Graph overview, REST API reference

Examples

```
## Not run:

# without any arguments, this will create a client using your AAD credentials
az <- create_graph_login()

# retrieve the login in subsequent sessions
az <- get_graph_login()

# this will create an Microsoft Graph client for the tenant 'microsoft.onmicrosoft.com',
# using the client_credentials method
az <- create_graph_login("microsoft", app="{app_id}", password="{password}")

# you can also login using credentials in a json file
```

```
az <- create_graph_login(config_file="~/creds.json")

## End(Not run)
```

`is_app` *Informational functions*

Description

These functions return whether the object is of the corresponding class.

Usage

```
is_app(object)

is_service_principal(object)

is_user(object)

is_group(object)

is_directory_object(object)
```

Arguments

`object`	An R object.

Value

A boolean.

`ms_graph` *Azure Active Directory Graph*

Description

Base class for interacting with Microsoft Graph API.

Usage

```
ms_graph
```

Format

An R6 object of class `ms_graph`.

Methods

- `new(tenant,app,...)`: Initialize a new Microsoft Graph connection with the given credentials. See 'Authentication' for more details.
- `create_app(name,...,add_password=TRUE,password_name=NULL,password_duration=2,certificate=NULL,c` Creates a new registered app in Azure Active Directory. See 'App creation' below.
- `get_app(app_id,object_id)`: Retrieves an existing registered app, via either its app ID or object ID.
- `delete_app(app_id,object_id,confirm=TRUE)`: Deletes an existing registered app. Any associated service principal will also be deleted.
- `create_service_principal(app_id,...)`: Creates a service principal for a registered app.
- `get_service_principal()`: Retrieves an existing service principal.
- `delete_service_principal()`: Deletes an existing service principal.
- `create_user(name,email,enabled=TRUE,...,password=NULL,force_password_change=TRUE)`: Creates a new user account. By default this will be a work account (not social or local) in the current tenant, and will have a randomly generated password that must be changed at next login.
- `get_user(user_id)`: Retrieves an existing user account.
- `delete_user(user_id,confirm=TRUE)`: Deletes a user account.
- `create_group(name,email,...)`: Creates a new group. Note that only security groups can be created via the Microsoft Graph API.
- `get_group(group_id)`: Retrieves an existing group.
- `delete_group(group_id,confirm=TRUE)`: Deletes a group.
- `call_graph_endpoint(op="",...)`: Calls the Microsoft Graph API using this object's token and tenant as authentication arguments. See call_graph_endpoint.

Authentication

The recommended way to authenticate with Microsoft Graph is via the create_graph_login function, which creates a new instance of this class.

To authenticate with the `ms_graph` class directly, provide the following arguments to the `new` method:

- `tenant`: Your tenant ID. This can be a name ("myaadtenant"), a fully qualified domain name ("myaadtenant.onmicrosoft.com" or "mycompanyname.com"), or a GUID.
- `app`: The client/app ID to use to authenticate with Azure Active Directory. The default is to login interactively using the Azure CLI cross-platform app, but it's recommended to supply your own app credentials if possible.
- `password`: if `auth_type == "client_credentials"`, the app secret; if `auth_type == "resource_owner"`, your account password.
- `username`: if `auth_type == "resource_owner"`, your username.
- `certificate`: If 'auth_type == "client_credentials", a certificate to authenticate with. This is a more secure alternative to using an app secret.

- auth_type: The OAuth authentication method to use, one of "client_credentials", "authorization_code", "device_code" or "resource_owner". See get_azure_token for how the default method is chosen, along with some caveats.
- host: your Microsoft Graph host. Defaults to https://graph.microsoft.com/.
- aad_host: Azure Active Directory host for authentication. Defaults to https://login.microsoftonline.com/. Change this if you are using a government or private cloud.
- token: Optionally, an OAuth 2.0 token, of class AzureAuth::AzureToken. This allows you to reuse the authentication details for an existing session. If supplied, all other arguments will be ignored.

App creation

The create_app method creates a new registered app. By default, a new app will have a randomly generated strong password with duration of 2 years. To skip assigning a password, set the add_password argument to FALSE.

The certificate argument allows authenticating via a certificate instead of a password. This should be a character string containing the certificate public key (aka the CER file). Alternatively it can be an list, or an object of class AzureKeyVault::stored_cert representing a certificate stored in an Azure Key Vault. See the examples below.

A new app will also have a service principal created for it by default. To disable this, set create_service_principal=FALSE

See Also

create_graph_login, get_graph_login

Microsoft Graph overview, REST API reference

Examples

```
## Not run:

# start a new Graph session
gr <- ms_graph$new(tenant="myaadtenant.onmicrosoft.com", app="app_id", password="password")

# authenticate with credentials in a file
gr <- ms_graph$new(config_file="creds.json")

# authenticate with device code
gr <- ms_graph$new(tenant="myaadtenant.onmicrosoft.com", app="app_id", auth_type="device_code")

# retrieve a registered app
gr$get_app(app_id="myappid")

# create a new app and associated service principal, set password duration to 10 years
app <- gr$create_app("mynewapp", password_duration=10)

# delete the app
gr$delete_app(app_id=app$properties$appId)
# ... but better to call the object's delete method directly
app$delete()
```

```
# create an app with authentication via a certificate
cert <- readLines("mycert.cer")
gr$create_app("mycertapp", password=FALSE, certificate=cert)


## End(Not run)
```

Index