Package 'AzureGraph'

November 2, 2026

Title Simple Interface to 'Microsoft Graph'

Version 1.3.6

Description A simple interface to the 'Microsoft Graph' API <https://learn.microsoft.com/en-us/graph/overview>. 'Graph' is a comprehensive framework for accessing data in various online Microsoft services. This package was originally intended 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 has since been expanded into a more general tool for interacting with Graph. 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, curl, R6

Suggests AzureRMR, vctrs, knitr, rmarkdown, testthat

RoxygenNote 7.3.1

NeedsCompilation no

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

Maintainer Hong Ooi <hongooi73@gmail.com>

Repository CRAN

Date/Publication 2026-11-02 11:30:02 UTC

Contents

az_app 2
az_device 5

az_directory_role 6
az_group 7
az_object 8
az_service_principal 10
az_user 11
call_batch_endpoint 13
call_graph_endpoint 14
create_graph_login 16
extract_list_values 18
find_class_generator 19
format_public_fields 20
graph_request 21
is_app 22
ms_graph 23
ms_graph_pager 26
ms_object 28
named_list 30
register_graph_class 31

Index **33**

---

az_app *Registered app in Azure Active Directory*

Description

Base class representing an AAD 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_owners(type=c("user", "group", "application", "servicePrincipal"), filter=NULL, n=Inf)`: Return a list of all owners of this app. Specify the `type` argument to limit the result to 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

List methods

All `list_*` methods have `filter` and `n` arguments to limit the number of results. The former should be an OData expression as a string to filter the result set on. The latter should be a number setting the maximum number of (filtered) results to return. The default values are `filter=NULL` and `n=Inf`. If `n=NULL`, the `ms_graph_pager` iterator object is returned instead to allow manual iteration over the results.

Support in the underlying Graph API for OData queries is patchy. Not all endpoints that return lists of objects support filtering, and if they do, they may not allow all of the defined operators. If your filtering expression results in an error, you can carry out the operation without filtering and then filter the results on the client side.

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

Class representing a registered 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.

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_directory_role *Directory role*

Description

Class representing a role in Azure Active Directory.

Format

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

Fields

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

Methods

- `new(...)`: Initialize a new object. Do not call this directly; see 'Initialization' below.
- `delete(confirm=TRUE)`: Delete this item. By default, ask for confirmation first.
- `update(...)`: Update the item's properties in Microsoft Graph.
- `do_operation(...)`: Carry out an arbitrary operation on the item.
- `sync_fields()`: Synchronise the R object with the item metadata in Microsoft Graph.
- `list_members(filter=NULL, n=Inf)`: Return a list of all members of this group.

Initialization

Currently support for directory roles is limited. Objects of this class should not be initialized directly.

List methods

All `list_*` methods have `filter` and `n` arguments to limit the number of results. The former should be an OData expression as a string to filter the result set on. The latter should be a number setting the maximum number of (filtered) results to return. The default values are `filter=NULL` and `n=Inf`. If `n=NULL`, the `ms_graph_pager` iterator object is returned instead to allow manual iteration over the results.

Support in the underlying Graph API for OData queries is patchy. Not all endpoints that return lists of objects support filtering, and if they do, they may not allow all of the defined operators. If your filtering expression results in an error, you can carry out the operation without filtering and then filter the results on the client side.

See Also

ms_graph, az_user

Microsoft Graph overview, REST API reference

---

az_group *Group in Azure Active Directory*

Description

Class representing an AAD 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_members(type=c("user", "group", "application", "servicePrincipal"), filter=NULL, n=Inf): Return a list of all members of this group. Specify the type argument to limit the result to specific object type(s).
- list_owners(type=c("user", "group", "application", "servicePrincipal"), filter=NULL, n=Inf): Return a list of all owners of this group. Specify the type argument to limit the result to 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.

List methods

All `list_*` methods have `filter` and `n` arguments to limit the number of results. The former should be an OData expression as a string to filter the result set on. The latter should be a number setting the maximum number of (filtered) results to return. The default values are `filter=NULL` and `n=Inf`. If `n=NULL`, the `ms_graph_pager` iterator object is returned instead to allow manual iteration over the results.

Support in the underlying Graph API for OData queries is patchy. Not all endpoints that return lists of objects support filtering, and if they do, they may not allow all of the defined operators. If your filtering expression results in an error, you can carry out the operation without filtering and then filter the results on the client side.

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$list_group_memberships()
grp <- gr$get_group(grps[1])

grp$list_members()
grp$list_owners()

# capping the number of results
grp$list_members(n=10)

# get the pager object for a listing method
pager <- grp$list_members(n=NULL)
pager$value


## End(Not run)
```

---

az_object *Azure Active Directory object*

Description

Base class representing an Azure Active Directory object in Microsoft Graph.

Format

An R6 object of class `az_object`, inheriting from `ms_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(security_only=FALSE, filter=NULL, n=Inf)`: Return the IDs of all groups this object is a member of. If `security_only` is TRUE, only security group IDs are returned.
- `list_object_memberships(security_only=FALSE, filter=NULL, n=Inf)`: 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.

List methods

All `list_*` methods have `filter` and `n` arguments to limit the number of results. The former should be an OData expression as a string to filter the result set on. The latter should be a number setting the maximum number of (filtered) results to return. The default values are `filter=NULL` and `n=Inf`. If `n=NULL`, the `ms_graph_pager` iterator object is returned instead to allow manual iteration over the results.

Support in the underlying Graph API for OData queries is patchy. Not all endpoints that return lists of objects support filtering, and if they do, they may not allow all of the defined operators. If your filtering expression results in an error, you can carry out the operation without filtering and then filter the results on the client side.

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

Class representing an AAD 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.

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

Class representing an AAD user account.

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_direct_memberships(filter=NULL, n=Inf)`: List the groups and directory roles this user is a direct member of.
- `list_owned_objects(type=c("user", "group", "application", "servicePrincipal"), filter=NULL, n=Inf)`: List directory objects (groups/apps/service principals) owned by this user. Specify the `type` argument to limit the result to specific object type(s).
- `list_created_objects(type=c("user", "group", "application", "servicePrincipal"), filter=NULL, n=Inf)`: List directory objects (groups/apps/service principals) created by this user. Specify the `type` argument to limit the result to specific object type(s).
- `list_owned_devices(filter=NULL, n=Inf)`: List the devices owned by this user.
- `list_registered_devices(filter=NULL, n=Inf)`: 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.

List methods

All `list_*` methods have `filter` and `n` arguments to limit the number of results. The former should be an OData expression as a string to filter the result set on. The latter should be a number setting the maximum number of (filtered) results to return. The default values are `filter=NULL` and `n=Inf`. If `n=NULL`, the `ms_graph_pager` iterator object is returned instead to allow manual iteration over the results.

Support in the underlying Graph API for OData queries is patchy. Not all endpoints that return lists of objects support filtering, and if they do, they may not allow all of the defined operators. If your filtering expression results in an error, you can carry out the operation without filtering and then filter the results on the client side.

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"))

# first 5 objects
usr$list_owned_objects(n=5)

# get the pager object
pager <- usr$list_owned_objects(n=NULL)
pager$value


## End(Not run)
```

`call_batch_endpoint` *Call the Graph API batch endpoint*

Description

Call the Graph API batch endpoint

Usage

```
call_batch_endpoint(token, requests = list(), depends_on = NULL,
  api_version = getOption("azure_graph_api_version"))
```

Arguments

`token`	An Azure OAuth token, of class AzureAuth::AzureToken.
`requests`	A list of graph_request objects, representing individual requests to the Graph API.
`depends_on`	An optional named vector, or TRUE. See below.
`api_version`	The API version to use, which will form part of the URL sent to the host.

Details

Use this function to combine multiple requests into a single HTTPS call. This can save significant network latency.

The `depends_on` argument specifies the dependencies that may exist between requests. The default is to treat the requests as independent, which allows them to be executed in parallel. If `depends_on` is TRUE, each request is specified as depending on the immediately preceding request. Otherwise, this should be a named vector or list that gives the dependency or dependencies for each request.

There are 2 restrictions on `depends_on`:

- If one request has a dependency, then all requests must have dependencies specified
- A request can only depend on previous requests in the list, not on later ones.

A request list that has dependencies will be executed serially.

Value

A list containing the responses to each request. Each item has components `id` and `status` at a minimum. It may also contain `headers` and `body`, depending on the specifics of the request.

See Also

graph_request, call_graph_endpoint

Microsoft Graph overview, Batch endpoint documentation

OData documentation on batch requests

Examples

```
## Not run:

req1 <- graph_request$new("me")

# a new email message in Outlook
req_create <- graph_request$new("me/messages",
    body=list(
        body=list(
            content="Hello from R",
            content_type="text"
        ),
        subject="Hello",
        toRecipients="bob@example.com"
    ),
    http_verb="POST"
)

# messages in drafts
req_get <- graph_request$new("me/mailFolders/drafts/messages")

# requests are dependent: 2nd list of drafts will include just-created message
call_batch_endpoint(token, list(req_get, req_create, req_get), depends_on=TRUE)

# alternate way: enumerate all requests
call_batch_endpoint(token, list(req_get, req_create, req_get), depends_on=c("2"=1, "3"=2))


## 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"),
  simplify = FALSE, auto_refresh = TRUE)
```

Arguments

token	An Azure OAuth token, of class AzureAuth::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.
simplify	Whether to turn arrays of objects in the JSON response into data frames. Set this to TRUE if you are expecting the endpoint to return tabular data and you want a tabular result, as opposed to a list of objects.
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.

If simplify is TRUE, call_graph_url will exploit the ability of jsonlite::fromJSON to convert arrays of objects into R data frames. This can be useful for REST calls that return tabular data. However, it can also cause problems for *paged* lists, where each page will be turned into a separate data frame; as the individual objects may not have the same fields, the resulting data frames will also have differing columns. This will cause base R's rbind to fail when binding the pages together. When processing paged lists, AzureGraph will use vctrs::vec_rbind instead of rbind when the vctrs package is available; vec_rbind does not have this problem. For safety, you should only set simplify=TRUE when vctrs is installed.

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 = NULL, password = NULL,
  username = NULL, certificate = NULL, auth_type = NULL, version = 2,
  host = "https://graph.microsoft.com/",
  aad_host = "https://login.microsoftonline.com/", scopes = ".default",
  config_file = NULL, token = NULL, ...)

get_graph_login(tenant = "common", selection = NULL, app = NULL,
  scopes = NULL, auth_type = 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.
`version`	The Azure Active Directory version to use for authenticating.
`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.

scopes	The Microsoft Graph scopes (permissions) to obtain for this Graph login. For create_graph_login, this is used only for version=2. For get_graph_login, set this to NA to require an AAD v1.0 token.
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 AzureAuth::get_azure_token, which automatically caches and reuses tokens for subsequent sessions.

For interactive use, you would normally *not* supply the username and password arguments. Omitting them will prompt create_graph_login to authenticate you with AAD using your browser, which is the recommended method. If you don't have a browser available to your R session, for example if you're using RStudio Server or Azure Databricks, you can specify auth_type="device_code".

For non-interactive use, for example if you're calling AzureGraph in a deployment pipeline, the recommended authentication method is via client credentials. For this method, you supply *only* the password argument, which should contain the client secret for your app registration. You must also specify your own app registration ID, in the app argument.

The AzureAuth package has a vignette that goes into more detail on these authentication scenarios.

get_graph_login returns a previously created login client. If there are multiple existing clients, you can specify which client to return via the selection, app, scopes and auth_type arguments. If you don't specify which one to return, it will pop up a menu and ask 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.

See Also

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

AzureAuth vignette on authentication scenarios

Microsoft Graph overview, REST API reference

Examples

```
## Not run:

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

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

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

# you can also login using credentials in a json file
az <- create_graph_login(config_file="~/creds.json")

# creating and obtaining a login with specific scopes
create_graph_login("mytenant", scopes=c("User.Read", "Files.ReadWrite.All"))
get_graph_login("mytenant", scopes=c("User.Read", "Files.ReadWrite.All"))

# to use your personal account, set the tenant to one of the following
create_graph_login("9188040d-6c67-4c5b-b112-36a304b66dad")
create_graph_login("consumers")  # requires AzureAuth 1.3.0


## End(Not run)
```

---

extract_list_values *Get the list of values from a Graph pager object*

Description

Get the list of values from a Graph pager object

Usage

```
extract_list_values(pager, n = Inf)
```

Arguments

pager	An object of class ms_graph_pager, which is an iterator for a list of paged query results.
n	The number of items from the list to return. Note this is *not* the number of *pages* (each page will usually contain multiple items). The default value of Inf extracts all the values from the list, leaving the pager empty. If this is NULL, the pager itself is returned.

Details

This is a convenience function to perform the common task of extracting all or some of the items from a paged response.

Value

If n is Inf or a number, the items from the paged query results. The format of the returned value depends on the pager settings. This will either be a nested list containing the properties for each of the items; a list of R6 objects; or a data frame. If the pager is empty, an error is thrown.

If n is NULL, the pager itself is returned.

See Also

ms_graph_pager, ms_object, call_graph_endpoint

Examples

```
## Not run:

firstpage <- call_graph_endpoint(token, "me/memberOf")
pager <- ms_graph_pager$new(token, firstpage)
extract_list_values(pager)

# trying to extract values a 2nd time will fail
try(extract_list_values(pager))


## End(Not run)
```

---

find_class_generator *Find the R6 class for a Graph object*

Description

Find the R6 class for a Graph object

Usage

```
find_class_generator(props, type_filter = NULL,
  default_generator = ms_object)
```

Arguments

`props`	A list of object properties, generally the result of a Graph API call.
`type_filter`	An optional vector of types by which to filter the result.
`default_generator`	The default class generator to use, if a match couldn't be found.

Details

This function maps Graph objects to AzureGraph classes.

Value

An R6 class generator for the appropriate AzureGraph class. If no matching R6 class could be found, the default generator is returned. If `type_filter` is provided, but the matching R6 class isn't in the filter, NULL is returned.

`format_public_fields` *Format a Microsoft Graph or Azure object*

Description

Miscellaneous functions for printing Microsoft Graph and Azure R6 objects

Usage

```
format_public_fields(env, exclude = character(0))

format_public_methods(env)
```

Arguments

`env`	An R6 object's environment for printing.
`exclude`	Objects in `env` to exclude from the printout.

Details

These are utilities to aid in printing R6 objects created by this package or its descendants. They are not meant to be called by the user.

graph_request *Microsoft Graph request*

Description

Class representing a request to the Microsoft Graph API. Currently this is used only in building a batch call.

Format

An R6 object of class `graph_request`.

Methods

- `new(...)`: Initialize a new request object with the given parameters. See 'Details' below.
- `batchify()`: Generate a list object suitable for incorporating into a call to the batch endpoint.

Details

The `initialize()` method takes the following arguments, representing the components of a HTTPS request:

- `op`: The path of the HTTPS URL, eg `/me/drives`.
- `body`: The body of the HTTPS request, if it is a PUT, POST or PATCH.
- `options`: A list containing the query parameters for the URL.
- `headers`: Any optional HTTP headers for the request.
- `encode`: If a request body is present, how it should be encoded when sending it to the endpoint. The default is `json`, meaning it will be sent as JSON text; an alternative is `raw`, for binary data.
- `http_verb`: One of "GET" (the default), "DELETE", "PUT", "POST", "HEAD", or "PATCH".

This class is currently used only for building batch calls. Future versions of AzureGraph may be refactored to use it in general API calls as well.

See Also

call_batch_endpoint

Microsoft Graph overview, Batch endpoint documentation

Examples

```
graph_request$new("me")

# a new email message in Outlook
graph_request$new("me/messages",
    body=list(
        body=list(
            content="Hello from R",
```

```
            content_type="text"
        ),
        subject="Hello",
        toRecipients="bob@example.com"
    ),
    http_verb="POST"
)
```

---

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_role(object)

is_aad_object(object)

is_msgraph_object(object)
```

Arguments

object	An R object.

Value

A boolean.

ms_graph *Microsoft Graph*

Description

Base class for interacting with Microsoft Graph API.

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, create_service_principal=TRUE)`: Creates a new app registration in Azure Active Directory. See 'App creation' below.
- `get_app(app_id, object_id)`: Retrieves an existing app registration, via either its app ID or object ID.
- `list_apps(filter=NULL, n=Inf)`: Lists the app registrations in the current tenant.
- `delete_app(app_id, object_id, confirm=TRUE)`: Deletes an existing app registration. Any associated service principal will also be deleted.
- `create_service_principal(app_id, ...)`: Creates a service principal for a app registration.
- `get_service_principal()`: Retrieves an existing service principal.
- `list_service_principals(filter=NULL, n=Inf)`: Lists the service principals in the current tenant.
- `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, email, name)`: Retrieves an existing user account. You can supply either the user ID, email address, or display name. The default is to return the logged-in user.
- `list_users(filter=NULL, n=Inf)`: Lists the users in the current tenant.
- `delete_user(user_id, email, name, 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, name)`: Retrieves an existing group.
- `list_groups(filter=NULL, n=Inf)`: Lists the groups in the current tenant.
- `delete_group(group_id, name, 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.
- `call_batch_endpoint(requests=list(), ...)`: Calls the batch endpoint with a list of individual requests. See call_batch_endpoint.
- `get_aad_object(id)`: Retrieves an arbitrary Azure Active Directory object by ID.

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 AzureAuth::get_azure_token for how the default method is chosen, along with some caveats.
- `version`: The Azure Active Directory (AAD) version to use for authenticating.
- `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.
- `scopes`: The Microsoft Graph scopes (permissions) to obtain for this Graph login. Only for `version=2`.
- `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 app registration. 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`

List methods

All `list_*` methods have `filter` and `n` arguments to limit the number of results. The former should be an OData expression as a string to filter the result set on. The latter should be a number setting the maximum number of (filtered) results to return. The default values are `filter=NULL` and `n=Inf`. If `n=NULL`, the `ms_graph_pager` iterator object is returned instead to allow manual iteration over the results.

Support in the underlying Graph API for OData queries is patchy. Not all endpoints that return lists of objects support filtering, and if they do, they may not allow all of the defined operators. If your filtering expression results in an error, you can carry out the operation without filtering and then filter the results on the client side.

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")

# 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 an app registration
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)

# retrieving your own user details (assuming interactive authentication)
gr$get_user()

# retrieving another user's details
gr$get_user("username@myaadtenant.onmicrosoft.com")
gr$get_user(email="firstname.lastname@mycompany.com")
```

```
gr$get_user(name="Hong Ooi")

# get an AAD object (a group)
id <- gr$get_user()$list_group_memberships()[1]
gr$get_aad_object(id)

# list the users in the tenant
gr$list_users()

# list (guest) users with a 'gmail.com' email address
gr$list_users(filter="endsWith(mail,'gmail.com')")

# list Microsoft 365 groups
gr$list_groups(filter="groupTypes/any(c:c eq 'Unified')")


## End(Not run)
```

`ms_graph_pager` *Pager object for Graph list results*

Description

Class representing an *iterator* for a set of paged query results.

Format

An R6 object of class `ms_graph_pager`.

Fields

- `token`: The token used to authenticate with the Graph host.
- `output`: What the pager should yield on each iteration, either "data.frame","list" or "object". See 'Value' below.

Methods

- `new(...)`: Initialize a new user object. See 'Initialization' below.
- `has_data()`: Returns TRUE if there are pages remaining in the iterator, or FALSE otherwise.

Active bindings

- `value`: The returned value on each iteration of the pager.

Initialization

The recommended way to create objects of this class is via the `ms_object$get_list_pager()` method, but it can also be initialized directly. The arguments to the `new()` method are:

- `token`: The token used to authenticate with the Graph host.
- `first_page`: A list containing the first page of results, generally from a call to `call_graph_endpoint()` or the `do_operation()` method of an AzureGraph R6 object.
- `next_link_name,value_name`: The names of the components of `first_page` containing the link to the next page, and the set of values for the page respectively. The default values are `@odata.nextLink` and `value`.
- `generate_objects`: Whether the iterator should return a list containing the parsed JSON for the page values, or convert it into a list of R6 objects. See 'Value' below.
- `type_filter`: Any extra arguments required to initialise the returned objects. Only used if `generate_objects` is TRUE.
- `default_generator`: The default generator object to use when converting a list of properties into an R6 object, if the class can't be detected. Defaults to `ms_object`. Only used if `generate_objects` is TRUE.
- `...`: Any extra arguments required to initialise the returned objects. Only used if `generate_objects` is TRUE.

Value

The `value` active binding returns the page values for each iteration of the pager. This can take one of 3 forms, based on the initial format of the first page and the `generate_objects` argument.

If the first page of results is a data frame (each item has been converted into a row), then the pager will return results as data frames. In this case, the `output` field is automatically set to "data.frame" and the `generate_objects` initialization argument is ignored. Usually this will be the case when the results are meant to represent external data, eg items in a SharePoint list.

If the first page of results is a list, the `generate_objects` argument sets whether to convert the items in each page into R6 objects defined by the AzureGraph class framework. If `generate_objects` is TRUE, the `output` field is set to "object", and if `generate_objects` is FALSE, the `output` field is set to "list".

See Also

ms_object, extract_list_values

Microsoft Graph overview, Paging documentation

Examples

```
## Not run:

# list direct memberships
firstpage <- call_graph_endpoint(token, "me/memberOf")

pager <- ms_graph_pager$new(token, firstpage)
pager$has_data()
```

```
pager$value

# once all the pages have been returned
isFALSE(pager$has_data())
is.null(pager$value)

# returning items, 1 per page, as raw lists of properties
firstpage <- call_graph_endpoint(token, "me/memberOf", options=list(`$top`=1))
pager <- ms_graph_pager$new(token, firstpage, generate_objects=FALSE)
lst <- NULL
while(pager$has_data())
    lst <- c(lst, pager$value)

# returning items as a data frame
firstdf <- call_graph_endpoint(token, "me/memberOf", options=list(`$top`=1),
                               simplify=TRUE)
pager <- ms_graph_pager$new(token, firstdf)
df <- NULL
while(pager$has_data())
    df <- vctrs::vec_rbin(df, pager$value)


## End(Not run)
```

`ms_object` *Microsoft Graph object*

Description

Base class representing a object in Microsoft Graph. All other Graph object classes ultimately inherit from this class.

Format

An R6 object of class `ms_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, in a human-readable format.
- `properties`: The object properties, as obtained from the Graph host.

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.
- `get_list_pager(...)`: Returns a pager object, which is an *iterator* for a set of paged query results. See 'Paged results' below.

Initialization

Objects of this class should not be created directly. Instead, create an object of the appropriate subclass.

List methods

All `list_*` methods have `filter` and `n` arguments to limit the number of results. The former should be an OData expression as a string to filter the result set on. The latter should be a number setting the maximum number of (filtered) results to return. The default values are `filter=NULL` and `n=Inf`. If `n=NULL`, the `ms_graph_pager` iterator object is returned instead to allow manual iteration over the results.

Support in the underlying Graph API for OData queries is patchy. Not all endpoints that return lists of objects support filtering, and if they do, they may not allow all of the defined operators. If your filtering expression results in an error, you can carry out the operation without filtering and then filter the results on the client side.

Paged results

Microsoft Graph returns lists in pages, with each page containing a subset of objects and a link to the next page. AzureGraph provides an iterator-based API that lets you access each page individually, or collect them all into a single object.

To create a new pager object, call the `get_list_pager()` method with the following arguments:

- `lst`: A list containing the first page of results, generally from a call to the `do_operation()` method.
- `next_link_name,value_name`: The names of the components of `first_page` containing the link to the next page, and the set of values for the page respectively. The default values are `@odata.nextLink` and `value`.
- `generate_objects`: Whether the iterator should return a list containing the parsed JSON for the page values, or convert it into a list of R6 objects.
- `type_filter`: Any extra arguments required to initialise the returned objects. Only used if `generate_objects` is TRUE.
- `default_generator`: The default generator object to use when converting a list of properties into an R6 object, if the class can't be detected. Defaults to `ms_object`. Only used if `generate_objects` is TRUE.
- `...`: Any extra arguments required to initialise the returned objects. Only used if `generate_objects` is TRUE.

This returns an object of class ms_graph_pager, which is an *iterator* for the set of paged results. Each call to the object's `value` active binding yields the next page. When all pages have been returned, `value` contains NULL.

The format of the returned values can take one of 3 forms, based on the initial format of the first page and the `generate_objects` argument.

If the first page of results is a data frame (each item has been converted into a row), then the pager will return results as data frames. In this case, the `output` field is automatically set to "data.frame" and the `generate_objects` initialization argument is ignored. Usually this will be the case when the results are meant to represent external data, eg items in a SharePoint list.

If the first page of results is a list, the `generate_objects` argument sets whether to convert the items in each page into R6 objects defined by the AzureGraph class framework. If `generate_objects` is TRUE, the `output` field is set to "object", and if `generate_objects` is FALSE, the `output` field is set to "list".

You can also call the `extract_list_values()` function to get all or some of the values from a pager, without having to manually combine the pages together.

Deprecated methods

The following methods are private and **deprecated**, and form the older AzureGraph API for accessing paged results. They will eventually be removed.

- `get_paged_list(lst, next_link_name, value_name, simplify, n)`: This method reconstructs the list, given the first page.
- `init_list_objects(lst, type_filter, default_generator, ...)`: `get_paged_list` returns a raw list, the result of parsing the JSON response from the Graph host. This method converts the list into actual R6 objects.

See Also

ms_graph, az_object, ms_graph_pager, extract_list_values

Microsoft Graph overview, REST API reference

---

named_list *Miscellaneous utility functions*

Description

Miscellaneous utility functions

Usage

```
named_list(lst = NULL, name_fields = "name")

is_empty(x)
```

Arguments

`lst`	A named list of objects.
`name_fields`	The components of the objects in `lst`, to be used as names.
`x`	For `is_empty`, An R object.

Details

named_list extracts from each object in lst, the components named by name_fields. It then constructs names for lst from these components, separated by a "/".

Value

For named_list, the list that was passed in but with names. An empty input results in a *named list* output: a list of length 0, with a names attribute.

For is_empty, whether the length of the object is zero (this includes the special case of NULL).

---

register_graph_class *Extensible registry of Microsoft Graph classes that AzureGraph supports*

Description

Extensible registry of Microsoft Graph classes that AzureGraph supports

Usage

```
register_graph_class(name, R6_generator, check_function)
```

Arguments

name	The name of the Graph class, eg "user", "servicePrincipal", etc.
R6_generator	An R6 class generator corresponding to this Graph class.
check_function	A boolean function that checks if a list of properties is for an object of this class.

Details

As written, AzureGraph knows about a subset of all the object classes contained in Microsoft Graph. These are mostly the classes originating from Azure Active Directory: users, groups, app registrations, service principals and registered devices.

You can extend AzureGraph by writing your own R6 class that inherits from ms_object. If so, you should also *register* your class by calling register_graph_class and providing the generator object, along with a check function. The latter should accept a list of object properties (as obtained from the Graph REST API), and return TRUE/FALSE based on whether the object is of your class.

Value

An invisible vector of registered class names.

Examples

```
## Not run:

# built-in 'az_user' class, for an AAD user object
register_graph_class("user", az_user,
   function(props) !is.null(props$userPrincipalName))


## End(Not run)
```

Index

az_app, 2, *7–12*
az_device, 5, *12*
az_directory_role, 6
az_group, *4*, 7, *9*, *12*
az_object, *4, 5*, *8*, 8, *10*, *12*, *30*
az_service_principal, *4*, *9*, 10
az_user, *4, 5*, *7–9*, 11
AzureAuth::AzureToken, *13*, *15*, *17*, *24*
AzureAuth::get_azure_token, *17, 18*, *24*

call_batch_endpoint, 13, *21*, *24*
call_graph_endpoint, *13*, 14, *19*, *24*
call_graph_url *(*call_graph_endpoint*)*, 14
create_graph_login, 16, *24, 25*

delete_graph_login
        *(*create_graph_login*)*, 16

extract_list_values, 18, *27*, *30*

find_class_generator, 19
format_public_fields, 20
format_public_methods
        *(*format_public_fields*)*, 20

get_graph_login, *25*
get_graph_login *(*create_graph_login*)*, 16
graph_request, *13*, 21

httr::content, *15*
httr::DELETE, *15*
httr::GET, *15*
httr::POST, *15*
httr::PUT, *15*
httr::stop_for_status, *15*

is_aad_object *(*is_app*)*, 22
is_app, 22
is_directory_role *(*is_app*)*, 22
is_empty *(*named_list*)*, 30
is_group *(*is_app*)*, 22
is_msgraph_object *(*is_app*)*, 22
is_service_principal *(*is_app*)*, 22
is_user *(*is_app*)*, 22

list_graph_logins *(*create_graph_login*)*,
        16

ms_graph, *3–5*, *7–12*, *18*, 23, *30*
ms_graph_pager, *19*, 26, *29, 30*
ms_object, *19*, *27*, 28

named_list, 30

register_graph_class, 31