

# Package ‘azr’

February 17, 2026

**Title** Credential Chain for Seamless 'OAuth 2.0' Authentication to 'Azure Services'

**Version** 0.3.0

**Description** Implements a credential chain for 'Azure OAuth 2.0' authentication based on the package 'htr2's 'OAuth' framework. Sequentially attempts authentication methods until one succeeds. During development allows interactive browser-based flows ('Device Code' and 'Auth Code' flows) and non-interactive flow ('Client Secret') in batch mode.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**URL** <https://pedrobtz.github.io/azr/>, <https://github.com/pedrobtz/azr>

**BugReports** <https://github.com/pedrobtz/azr/issues>

**Depends** R (>= 4.1)

**Imports** utils, R6, cli, htr2 (>= 1.2.2), jsonlite, rlang

**Suggests** data.table, httpuv, clipr, processx, testthat (>= 3.0.0), vcr (>= 2.0.0)

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Pedro Baltazar [aut, cre]

**Maintainer** Pedro Baltazar <pedrobtz@gmail.com>

**Repository** CRAN

**Date/Publication** 2026-02-17 19:30:02 UTC

## Contents

api_client . . . . .	2
api_resource . . . . .	7
api_service . . . . .	9
api_storage_client . . . . .	10

AuthCodeCredential . . . . .	12
azr_graph_client . . . . .	14
azr_storage_client . . . . .	16
AzureCLICredential . . . . .	17
az_cli_account_show . . . . .	19
az_cli_get_cached_token . . . . .	20
az_cli_get_token . . . . .	22
az_cli_is_login . . . . .	23
az_cli_login . . . . .	24
az_cli_logout . . . . .	25
CachedTokenCredential . . . . .	25
cached_token_credential_chain . . . . .	27
ClientSecretCredential . . . . .	28
credential_chain . . . . .	29
DefaultCredential . . . . .	30
default_azure_client_id . . . . .	33
default_azure_client_secret . . . . .	33
default_azure_config_dir . . . . .	34
default_azure_host . . . . .	34
default_azure_oauth_client . . . . .	35
default_azure_scope . . . . .	35
default_azure_tenant_id . . . . .	36
default_azure_url . . . . .	37
default_credential_chain . . . . .	38
default_non_auth . . . . .	38
default_redirect_uri . . . . .	39
default_refresh_token . . . . .	39
default_response_handler . . . . .	40
DeviceCodeCredential . . . . .	40
get_credential_auth . . . . .	42
get_credential_provider . . . . .	44
get_request_authorizer . . . . .	45
get_token . . . . .	46
get_token_provider . . . . .	48
is_hosted_session . . . . .	49
RefreshTokenCredential . . . . .	50

<b>Index</b>	<b>53</b>
--------------	-----------

---

api\_client

*Azure API Client*

---

### Description

An R6 class that provides a base HTTP client for interacting with Azure APIs. This client handles authentication, request building, retry logic, logging, and error handling for Azure API requests.

## Details

The `api_client` class is designed to be a base class for Azure service-specific clients. It provides:

- Automatic authentication using Azure credentials
- Configurable retry logic with exponential backoff
- Request and response logging
- JSON, XML, and HTML content type handling
- Standardized error handling

## Public fields

`.host_url` Base URL for the API  
`.base_req` Base `httr2` request object  
`.provider` Credential provider R6 object  
`.credentials` Credentials function for authentication  
`.options` Request options (timeout, connecttimeout, max\_tries)  
`.response_handler` Optional callback function to process response content

## Methods

### Public methods:

- `api_client$new()`
- `api_client$.fetch()`
- `api_client$.resp_content()`
- `api_client$.build_request()`
- `api_client$.send_request()`
- `api_client$.resp_body_content()`
- `api_client$.get_token()`
- `api_client$.clone()`

**Method** `new()`: Create a new API client instance

*Usage:*

```
api_client$new(  
  host_url,  
  provider = NULL,  
  credentials = NULL,  
  timeout = 60L,  
  connecttimeout = 30L,  
  max_tries = 5L,  
  response_handler = NULL  
)
```

*Arguments:*

`host_url` A character string specifying the base URL for the API (e.g., "https://management.azure.com").

**provider** An R6 credential provider object that inherits from the `Credential` or `DefaultCredential` class. If provided, the credential's `req_auth` method will be used for authentication. Takes precedence over `credentials`.

**credentials** A function that adds authentication to requests. If both `provider` and `credentials` are `NULL`, uses `default_non_auth()`. The function should accept an `httr2` request object and return a modified request with authentication.

**timeout** An integer specifying the request timeout in seconds. Defaults to 60.

**connecttimeout** An integer specifying the connection timeout in seconds. Defaults to 30.

**max\_tries** An integer specifying the maximum number of retry attempts for failed requests. Defaults to 5.

**response\_handler** An optional function to process the parsed response content. The function should accept one argument (the parsed response) and return the processed content. If `NULL`, uses `default_response_handler()` which converts data frames to `data.table` objects. Defaults to `NULL`.

*Returns:* A new `api_client` object

**Method** `.fetch()`: Make an HTTP request to the API

*Usage:*

```
api_client$.fetch(
  path,
  ...,
  query = NULL,
  body = NULL,
  headers = NULL,
  method = "get",
  verbosity = 0L,
  content = c("body", "headers", "response", "request"),
  content_type = NULL
)
```

*Arguments:*

**path** A character string specifying the API endpoint path. Supports `rlang::engluce()` syntax for variable interpolation using named arguments passed via `...`

`...` Named arguments used for path interpolation with `rlang::engluce()`.

**query** A named list of query parameters to append to the URL.

**body** Request body data. Sent as JSON in the request body. Can be a list or character string (JSON).

**headers** A named list of additional HTTP headers to include in the request.

**method** A character string specifying the HTTP method. One of "get", "post", "put", "patch", or "delete". Defaults to "get".

**verbosity** An integer specifying the verbosity level for request debugging (passed to `httr2::req_perform()`). Defaults to 0.

**content** A character string specifying what to return. One of:

- "body" (default): Return the parsed response body
- "headers": Return response headers
- "response": Return the full `httr2` response object

- "request": Return the prepared request object without executing it
- content\_type A character string specifying how to parse the response body. If NULL, uses the response's Content-Type header. Common values: "application/json", "application/xml", "text/html".

*Returns:* Depends on the content parameter:

- "body": Parsed response body (list, data.frame, or character)
- "headers": List of response headers
- "response": Full `httr2::response()` object
- "request": `httr2::request()` object

**Method** `.resp_content()`: Extract content from a response object

*Usage:*

```
api_client$.resp_content(resp, content, content_type = NULL)
```

*Arguments:*

resp An `httr2::response()` object

content A character string specifying what to return. One of:

- "body": Return the parsed response body
- "headers": Return response headers
- "response": Return the full httr2 response object

content\_type A character string specifying how to parse the response body. Only used when content = "body". If NULL, uses the response's Content-Type header.

*Returns:* Depends on the content parameter:

- "body": Parsed response body (list, data.frame, or character)
- "headers": List of response headers
- "response": Full `httr2::response()` object

**Method** `.build_request()`: Build an HTTP request object

*Usage:*

```
api_client$.build_request(
  path,
  ...,
  query = NULL,
  body = NULL,
  headers = NULL,
  method = "get"
)
```

*Arguments:*

path A character string specifying the API endpoint path. Supports `rlang::engluce()` syntax for variable interpolation using named arguments passed via `...`

`...` Named arguments used for path interpolation with `rlang::engluce()`.

query A named list of query parameters to append to the URL.

body Request body data. Sent as JSON in the request body. Can be a list or character string (JSON).

**headers** A named list of additional HTTP headers to include in the request.  
**method** A character string specifying the HTTP method. One of "get", "post", "put", "patch", or "delete". Defaults to "get".

*Returns:* An `httr2::request()` object ready for execution

**Method** `.send_request()`: Perform an HTTP request and log the results

*Usage:*

```
api_client$.send_request(req, verbosity)
```

*Arguments:*

`req` An `httr2::request()` object to execute

`verbosity` An integer specifying the verbosity level for request debugging (passed to `httr2::req_perform()`). Defaults to 0.

*Returns:* An `httr2::response()` object containing the API response

**Method** `.resp_body_content()`: Extract and parse response content

*Usage:*

```
api_client$.resp_body_content(resp, content_type = NULL)
```

*Arguments:*

`resp` An `httr2::response()` object

`content_type` A character string specifying how to parse the response body. If NULL, uses the response's Content-Type header. Common values: "application/json", "application/xml", "text/html".

*Returns:* Parsed response body. Format depends on content type:

- JSON: List or data.frame
- XML: xml2 document
- HTML: xml2 document
- Other: Character string

**Method** `.get_token()`: Get authentication token from the credential provider

*Usage:*

```
api_client$.get_token()
```

*Returns:* An `httr2::oauth_token()` object if a provider is available, otherwise returns NULL with a warning.

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
api_client$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
## Not run:
# Create a client with default credentials
client <- api_client$new(
  host_url = "https://management.azure.com"
)

# Create a client with a credential provider
cred_provider <- get_credential_provider(
  scope = "https://management.azure.com/.default"
)
client <- api_client$new(
  host_url = "https://management.azure.com",
  provider = cred_provider
)

# Create a client with custom credentials function
client <- api_client$new(
  host_url = "https://management.azure.com",
  credentials = my_credential_function,
  timeout = 120,
  max_tries = 3
)

# Create a client with custom response handler
custom_handler <- function(content) {
  # Custom processing logic - e.g., keep data frames as-is
  content
}
client <- api_client$new(
  host_url = "https://management.azure.com",
  response_handler = custom_handler
)

# Make a GET request
response <- client$.fetch(
  path = "/subscriptions/{subscription_id}/resourceGroups",
  subscription_id = "my-subscription-id",
  query = list(`api-version` = "2021-04-01"),
  method = "get"
)

## End(Not run)
```

## Description

An R6 class that wraps an `api_client` and adds an additional path segment (like "beta" or "v1.0") to all requests. This is useful for APIs that version their endpoints or have different API surfaces

under different paths.

## Details

The `api_resource` class creates a modified base request by appending an endpoint path to the client's base request. All subsequent API calls through this resource will automatically include this path prefix.

## Public fields

`.client` The cloned `api_client` instance with modified `base_req`

## Methods

### Public methods:

- `api_resource$new()`
- `api_resource$clone()`

**Method** `new()`: Create a new API resource instance

*Usage:*

```
api_resource$new(client, endpoint)
```

*Arguments:*

`client` An `api_client` object that provides the base HTTP client functionality. This will be cloned to avoid modifying the original.

`endpoint` A character string specifying the API endpoint or path segment to append (e.g., "v1.0", "beta").

*Returns:* A new `api_resource` object

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
api_resource$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
## Not run:
# Create a client
client <- api_client$new(
  host_url = "https://graph.microsoft.com"
)

# Create a resource with v1.0 API endpoint
resource_v1 <- api_resource$new(
  client = client,
  endpoint = "v1.0"
)
```

```

# Create a resource with beta API endpoint
resource_beta <- api_resource$new(
  client = client,
  endpoint = "beta"
)

# Make requests - the endpoint is automatically prepended
response <- resource_v1$.fetch(
  path = "/me",
  method = "get"
)

## End(Not run)

```

---

api\_service

*API Service Base Class*


---

## Description

Base R6 class for creating API service wrappers. This class provides a foundation for building service-specific API clients with authentication, endpoint management, and configuration.

## Public fields

`.client` An [api\\_client](#) instance for making API requests

## Methods

### Public methods:

- [api\\_service\\$new\(\)](#)

**Method** `new()`: Create a new API service instance

*Usage:*

```

api_service$new(
  client = NULL,
  chain = NULL,
  endpoints = list(),
  config = list()
)

```

*Arguments:*

`client` An [api\\_client](#) instance. If NULL, a new client will be created.

`chain` A [credential\\_chain](#) instance for authentication. Optional.

`endpoints` A named list where names are endpoint paths (e.g., "v1.0", "beta") and values are R6 class objects (not instances) to use for creating resources. Defaults to an empty list. If the value is NULL, [api\\_resource](#) will be used.

`config` A list of configuration options. Defaults to an empty list.

*Returns:* A new `api_service` object

---

api\_storage\_client     *Azure Storage API Class*

---

### Description

An R6 class that extends [api\\_client](#) to provide specialized methods for Azure Data Lake Storage Gen2 (ADLS Gen2) REST API operations.

### Details

The base URL is constructed as: `https://{storageaccount}.dfs.core.windows.net`

### Super class

`azr::api_client` -> `api_storage_client`

### Public fields

`.filesystem` The filesystem (container) name

### Methods

#### Public methods:

- `api_storage_client$new()`
- `api_storage_client$download_file()`
- `api_storage_client$get_access_control()`
- `api_storage_client$list_files()`
- `api_storage_client$clone()`

**Method** `new()`: Create a new Azure Storage API client instance

#### Usage:

```
api_storage_client$new(  
  storageaccount,  
  filesystem,  
  scopes = ".default",  
  chain = NULL,  
  tenant_id = NULL,  
  ...  
)
```

#### Arguments:

`storageaccount` A character string specifying the Azure Storage account name.

`filesystem` A character string specifying the filesystem (container) name.

`scopes` A character string specifying the OAuth2 scope suffix. Defaults to `".default"`, which requests all permissions the app has been granted.

*chain* A [credential\\_chain](#) instance for authentication. If NULL, a default credential chain will be created using [DefaultCredential](#).

*tenant\_id* A character string specifying the Azure tenant ID. Passed to [DefaultCredential](#) when *chain* is NULL.

... Additional arguments passed to the parent [api\\_client](#) constructor.

*Returns:* A new `api_storage_client` object

**Method** `download_file()`: Download a file from the filesystem

*Usage:*

```
api_storage_client$download_file(path, dest = NULL)
```

*Arguments:*

*path* A character string specifying the file path within the filesystem.

*dest* A character string specifying the local destination path. Defaults to a temporary file via [tempfile\(\)](#).

*Returns:* The local path the file was written to (invisibly).

**Method** `get_access_control()`: Get the access control list (ACL) for a file or directory

*Usage:*

```
api_storage_client$get_access_control(dataset, upn = FALSE)
```

*Arguments:*

*dataset* A character string specifying the file or directory path within the filesystem.

*upn* A logical value. If TRUE, user principal names (UPN) are returned in the `x-ms-owner`, `x-ms-group`, and `x-ms-acl` response headers instead of object IDs. Defaults to FALSE.

*Returns:* A data.frame with columns `group_id` and `permission`, one row per named group entry in the `x-ms-acl` response header.

**Method** `list_files()`: List files and directories in a path

*Usage:*

```
api_storage_client$list_files(path = "", recursive = FALSE, ...)
```

*Arguments:*

*path* A character string specifying the directory path to list. Use empty string or NULL for the root directory. Defaults to "".

*recursive* A logical value indicating whether to list files recursively. Defaults to FALSE.

... Additional query parameters to pass to the API.

*Returns:* A data.frame (or data.table if available) containing file and directory information with columns such as `name`, `contentLength`, `lastModified`, etc.

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
api_storage_client$clone(deep = FALSE)
```

*Arguments:*

*deep* Whether to make a deep clone.

## Examples

```
## Not run:
# Create a storage client
storage <- api_storage_client$new(
  storageaccount = "mystorageaccount",
  filesystem = "mycontainer"
)

# List files in the root directory
files <- storage$list_files()

# List files in a specific path
files <- storage$list_files(path = "data/folder1")

# List files recursively
files <- storage$list_files(path = "data", recursive = TRUE)

## End(Not run)
```

---

AuthCodeCredential      *Authorization code credential authentication*

---

## Description

Authenticates a user through the OAuth 2.0 authorization code flow. This flow opens a web browser for the user to sign in.

## Details

The authorization code flow is the standard OAuth 2.0 flow for interactive authentication. It requires a web browser and is suitable for applications where the user can interact with a browser window.

The credential supports token caching to avoid repeated authentication. Tokens can be cached to disk or in memory. A redirect URI is required for the OAuth flow to complete.

## Super classes

azr::Credential -> [azr::InteractiveCredential](#) -> AuthCodeCredential

## Methods

### Public methods:

- [AuthCodeCredential\\$new\(\)](#)
- [AuthCodeCredential\\$get\\_token\(\)](#)
- [AuthCodeCredential\\$req\\_auth\(\)](#)
- [AuthCodeCredential\\$clone\(\)](#)

**Method** `new()`: Create a new authorization code credential

*Usage:*

```
AuthCodeCredential$new(
  scope = NULL,
  tenant_id = NULL,
  client_id = NULL,
  use_cache = "disk",
  offline = TRUE,
  redirect_uri = default_redirect_uri(),
  interactive = TRUE
)
```

*Arguments:*

`scope` A character string specifying the OAuth2 scope. Defaults to NULL.

`tenant_id` A character string specifying the Azure Active Directory tenant ID. Defaults to NULL.

`client_id` A character string specifying the application (client) ID. Defaults to NULL.

`use_cache` A character string specifying the cache type. Use "disk" for disk-based caching or "memory" for in-memory caching. Defaults to "disk".

`offline` A logical value indicating whether to request offline access (refresh tokens). Defaults to TRUE.

`redirect_uri` A character string specifying the redirect URI registered with the application. Defaults to `default_redirect_uri()`.

`interactive` A logical value indicating whether this credential requires user interaction. Defaults to TRUE.

*Returns:* A new `AuthCodeCredential` object

**Method** `get_token()`: Get an access token using authorization code flow

*Usage:*

```
AuthCodeCredential$get_token(reauth = FALSE)
```

*Arguments:*

`reauth` A logical value indicating whether to force reauthentication. Defaults to FALSE.

*Returns:* An `httr2::oauth_token()` object containing the access token

**Method** `req_auth()`: Add OAuth authorization code authentication to an `httr2` request

*Usage:*

```
AuthCodeCredential$req_auth(req)
```

*Arguments:*

`req` An `httr2::request()` object

*Returns:* The request object with OAuth authorization code authentication configured

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
AuthCodeCredential$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

**Examples**

```

# AuthCodeCredential requires an interactive session
## Not run:
# Create credential with default settings
cred <- AuthCodeCredential$new(
  tenant_id = "your-tenant-id",
  client_id = "your-client-id",
  scope = "https://management.azure.com/.default"
)

# Get an access token (will open browser for authentication)
token <- cred$get_token()

# Force reauthentication
token <- cred$get_token(reauth = TRUE)

# Use with httr2 request
req <- httr2::request("https://management.azure.com/subscriptions")
req <- cred$req_auth(req)

## End(Not run)

```

---

azr\_graph\_client

*Create a Microsoft Graph API Client*


---

**Description**

Creates a configured client for the Microsoft Graph API with authentication and versioned endpoints (v1.0 and beta). This function returns an [api\\_service](#) object that provides access to Microsoft Graph resources through versioned endpoints.

**Usage**

```
azr_graph_client(scopes = ".default", ..., chain = NULL)
```

**Arguments**

scopes	A character string specifying the OAuth2 scope suffix to be appended to the Graph API URL. Defaults to ".default", which requests all permissions the app has been granted. The full scope will be <code>https://graph.microsoft.com/{scopes}</code> .
...	Additional arguments passed to the <a href="#">api_client</a> constructor.
chain	A <a href="#">credential_chain</a> instance for authentication. If NULL, a default credential chain will be created using <a href="#">DefaultCredential</a> .

## Details

The function creates a Microsoft Graph service using these components:

- **api\_client**: A general-purpose API client configured with the Graph API host URL (<https://graph.microsoft.com>) and authentication provider.
- **api\_graph\_resource**: A specialized resource class that extends [api\\_resource](#) with Microsoft Graph-specific methods. Currently implements:
  - `me(select = NULL)`: Fetch the current user's profile. The `select` parameter accepts a character vector of properties to return (e.g., `c("displayName", "mail")`).
- **api\_service**: A service container that combines the client and resources with versioned endpoints (v1.0 and beta). The service is locked using `lockEnvironment()` to prevent modification after creation.

## Value

An [api\\_service](#) object configured for Microsoft Graph API with v1.0 and beta endpoints. The object is locked using `lockEnvironment()` to prevent modification after creation. Access endpoints via `$v1.0` or `$beta`.

## Examples

```
## Not run:
# Create a Graph API client with default credentials
graph <- azr_graph_client()

# Fetch current user profile from v1.0 endpoint
me <- graph$v1.0$me()

# Fetch specific properties using OData $select
me <- graph$v1.0$me(select = c("displayName", "mail", "userPrincipalName"))

# Use beta endpoint for preview features
me_beta <- graph$beta$me(select = c("displayName", "mail"))

# Create with a custom credential chain
custom_chain <- credential_chain(
  AzureCLICredential$new(scope = "https://graph.microsoft.com/.default")
)
graph <- azr_graph_client(chain = custom_chain)

# Use specific scopes instead of .default
graph <- azr_graph_client(scopes = "User.Read Mail.Read")

## End(Not run)
```

---

azr\_storage\_client      *Create an Azure Storage Client*

---

### Description

A convenience wrapper around [api\\_storage\\_client](#) that creates a configured client for Azure Data Lake Storage Gen2 (ADLS Gen2) REST API operations.

### Usage

```
azr_storage_client(  
  storageaccount,  
  filesystem,  
  chain = default_credential_chain(),  
  tenant_id = default_azure_tenant_id(),  
  ...  
)
```

### Arguments

storageaccount	A character string specifying the Azure Storage account name.
filesystem	A character string specifying the filesystem (container) name.
chain	A <a href="#">credential_chain</a> instance for authentication. Defaults to <a href="#">default_credential_chain()</a> .
tenant_id	A character string specifying the Azure tenant ID. Defaults to <a href="#">default_azure_tenant_id()</a> , which reads AZURE_TENANT_ID from the environment.
...	Additional arguments passed to the <a href="#">api_storage_client</a> constructor.

### Value

An [api\\_storage\\_client](#) object.

### Examples

```
## Not run:  
# Create a storage client with default credentials  
storage <- azr_storage_client(  
  storageaccount = "mystorageaccount",  
  filesystem = "mycontainer"  
)  
  
# Create a storage client with a specific tenant  
storage <- azr_storage_client(  
  storageaccount = "mystorageaccount",  
  filesystem = "mycontainer",  
  tenant_id = "00000000-0000-0000-0000-000000000000"  
)  
  
## End(Not run)
```

---

AzureCLICredential      *Azure CLI credential authentication*

---

## Description

Authenticates using the Azure CLI (az) command-line tool. This credential requires the Azure CLI to be installed and the user to be logged in via `az login`.

## Details

The credential uses the `az account get-access-token` command to retrieve access tokens. It will use the currently active Azure CLI account and subscription unless a specific tenant is specified.

## Super classes

`azr::Credential` -> `azr::InteractiveCredential` -> `AzureCLICredential`

## Public fields

`.process_timeout` Timeout in seconds for Azure CLI command execution

## Methods

### Public methods:

- `AzureCLICredential$new()`
- `AzureCLICredential$get_token()`
- `AzureCLICredential$req_auth()`
- `AzureCLICredential$account_show()`
- `AzureCLICredential$login()`
- `AzureCLICredential$logout()`
- `AzureCLICredential$clone()`

**Method** `new()`: Create a new Azure CLI credential

*Usage:*

```
AzureCLICredential$new(  
  scope = NULL,  
  tenant_id = NULL,  
  process_timeout = NULL,  
  interactive = FALSE,  
  use_bridge = FALSE  
)
```

*Arguments:*

`scope` A character string specifying the OAuth2 scope. Defaults to `NULL`, which uses the scope set during initialization.

`tenant_id` A character string specifying the Azure Active Directory tenant ID. Defaults to NULL, which uses the default tenant from Azure CLI.

`process_timeout` A numeric value specifying the timeout in seconds for the Azure CLI process. Defaults to 10.

`interactive` A logical value indicating whether to check if the user is logged in and perform login if needed. Defaults to FALSE.

`use_bridge` A logical value indicating whether to use the device code bridge webpage during login. If TRUE, launches an intermediate local webpage that displays the device code and facilitates copy-pasting before redirecting to the Microsoft device login page. Only used when `interactive = TRUE`. Defaults to FALSE.

*Returns:* A new `AzureCLICredential` object

**Method** `get_token()`: Get an access token from Azure CLI

*Usage:*

```
AzureCLICredential$get_token(scope = NULL)
```

*Arguments:*

`scope` A character string specifying the OAuth2 scope. If NULL, uses the scope specified during initialization.

*Returns:* An `httr2::oauth_token()` object containing the access token

**Method** `req_auth()`: Add authentication to an `httr2` request

*Usage:*

```
AzureCLICredential$req_auth(req, scope = NULL)
```

*Arguments:*

`req` An `httr2::request()` object

`scope` A character string specifying the OAuth2 scope. If NULL, uses the scope specified during initialization.

*Returns:* The request object with authentication header added

**Method** `account_show()`: Show the currently active Azure CLI account information

*Usage:*

```
AzureCLICredential$account_show(timeout = NULL)
```

*Arguments:*

`timeout` A numeric value specifying the timeout in seconds for the Azure CLI command. If NULL, uses the process timeout specified during initialization.

*Returns:* A list containing the account information from Azure CLI

**Method** `login()`: Perform Azure CLI login using device code flow

*Usage:*

```
AzureCLICredential$login()
```

*Returns:* Invisibly returns the exit status (0 for success, non-zero for failure)

**Method** `logout()`: Log out from Azure CLI

*Usage:*

```
AzureCLICredential$logout()
```

*Returns:* Invisibly returns NULL

**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*

```
AzureCLICredential$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

**Examples**

```
# Create credential with default settings
cred <- AzureCLICredential$new()

# Create credential with specific scope and tenant
cred <- AzureCLICredential$new(
  scope = "https://management.azure.com/.default",
  tenant_id = "your-tenant-id"
)

# To get a token or authenticate a request it is required that
# 'az login' is successfully executed, otherwise it will return an error.
## Not run:
# Get an access token
token <- cred$get_token()

# Use with httr2 request
req <- httr2::request("https://management.azure.com/subscriptions")
resp <- httr2::req_perform(cred$req_auth(req))

## End(Not run)
```

---

az\_cli\_account\_show    *Show Azure CLI Account Information*

---

**Description**

Retrieves information about the currently active Azure CLI account and subscription. This function runs `az account show` and parses the JSON output into an R list.

**Usage**

```
az_cli_account_show(timeout = 10L)
```

**Arguments**

timeout            An integer specifying the timeout in seconds for the Azure CLI command. Defaults to 10.

## Details

The function returns details about the current Azure subscription including:

- Subscription ID and name
- Tenant ID
- Account state (e.g., "Enabled")
- User information
- Cloud environment details

## Value

A list containing the account information from Azure CLI

## Examples

```
## Not run:
# Get current account information
account_info <- az_cli_account_show()

# Access subscription ID
subscription_id <- account_info$id

# Access tenant ID
tenant_id <- account_info$tenantId

## End(Not run)
```

---

az\_cli\_get\_cached\_token

*Get Cached Token from MSAL Token Cache*

---

## Description

Reads the MSAL token cache file (msal\_token\_cache.json) from the Azure configuration directory and returns a matching access token as an `httr2::oauth_token()` object.

## Usage

```
az_cli_get_cached_token(
  scope = NULL,
  tenant_id = NULL,
  client_id = NULL,
  config_dir = default_azure_config_dir()
)
```

## Arguments

scope	A character string specifying the OAuth2 scope to filter tokens. If NULL (default), returns the latest-expiring token regardless of scope.
tenant_id	A character string specifying the tenant ID to filter tokens. If NULL (default), matches any tenant.
client_id	A character string specifying the client ID to filter tokens. If NULL (default), matches any client.
config_dir	A character string specifying the Azure configuration directory. Defaults to <code>default_azure_config_dir()</code> .

## Details

The MSAL token cache is a JSON file maintained by the Azure CLI that stores access tokens and refresh tokens. This function reads cached access tokens directly from the file without invoking the Azure CLI, which can be useful in environments where the CLI is slow or unavailable but tokens have been previously cached.

When multiple tokens are found, the function selects the token that expires latest. If scope is provided, only tokens matching that scope/resource are returned.

## Value

An `httr2::oauth_token()` object containing:

- `access_token`: The OAuth2 access token string
- `token_type`: The type of token (typically "Bearer")
- `.expires_at`: POSIXct timestamp when the token expires

## Examples

```
## Not run:
# Get any cached token
token <- az_cli_get_cached_token()

# Get a cached token for a specific scope
token <- az_cli_get_cached_token(
  scope = "https://management.azure.com/.default"
)

# Access the token string
token$access_token

## End(Not run)
```

---

az_cli_get_token	<i>Get Access Token from Azure CLI</i>
------------------	--

---

### Description

Retrieves an access token from Azure CLI using the `az account get-access-token` command. This is a lower-level function that directly interacts with the Azure CLI to obtain OAuth2 tokens.

### Usage

```
az_cli_get_token(scope, tenant_id = NULL, timeout = 10L)
```

### Arguments

scope	A character string specifying the OAuth2 scope for which to request the access token (e.g., "https://management.azure.com/.default").
tenant_id	A character string specifying the Azure Active Directory tenant ID. If NULL, uses the default tenant from Azure CLI. Defaults to NULL.
timeout	A numeric value specifying the timeout in seconds for the Azure CLI process. Defaults to 10.

### Details

This function executes the Azure CLI command and parses the JSON response to create an `httr2::oauth_token()` object. The token includes the access token, token type, and expiration time.

### Value

An `httr2::oauth_token()` object containing:

- `access_token`: The OAuth2 access token string
- `token_type`: The type of token (typically "Bearer")
- `.expires_at`: POSIXct timestamp when the token expires

### Examples

```
## Not run:  
# Get a token for Azure Resource Manager  
token <- az_cli_get_token(  
  scope = "https://management.azure.com/.default"  
)  
  
# Get a token for a specific tenant  
token <- az_cli_get_token(  
  scope = "https://graph.microsoft.com/.default",  
  tenant_id = "your-tenant-id"  
)
```

```
# Access the token string
access_token <- token$access_token

## End(Not run)
```

---

az_cli_is_login	<i>Check if User is Logged in to Azure CLI</i>
-----------------	--

---

### Description

Checks whether the user is currently logged in to Azure CLI by attempting to retrieve account information.

### Usage

```
az_cli_is_login(timeout = 10L)
```

### Arguments

timeout	A numeric value specifying the timeout in seconds for the Azure CLI command. Defaults to 10.
---------	--

### Value

A logical value: TRUE if the user is logged in, FALSE otherwise

### Examples

```
## Not run:
# Check if logged in
if (az_cli_is_login()) {
  message("User is logged in")
} else {
  message("User is not logged in")
}

## End(Not run)
```

---

`az_cli_login`*Azure CLI Device Code Login*

---

**Description**

Performs an interactive Azure CLI login using device code flow. Automatically captures the device code, copies it to the clipboard, and opens the browser for authentication.

**Usage**

```
az_cli_login(tenant_id = NULL, use_bridge = FALSE, verbose = FALSE)
```

**Arguments**

<code>tenant_id</code>	A character string specifying the Azure Active Directory tenant ID to authenticate against. If NULL (default), uses the default tenant from Azure CLI configuration.
<code>use_bridge</code>	A logical value indicating whether to use the device code bridge webpage. If TRUE, launches an intermediate local webpage that displays the device code and facilitates copy-pasting before redirecting to the Microsoft device login page. If FALSE (default), copies the code directly to the clipboard and opens the Microsoft login page.
<code>verbose</code>	A logical value indicating whether to print detailed process output to the console, including error messages from the Azure CLI process. If FALSE (default), only essential messages are displayed.

**Details**

This function runs `az login --use-device-code`, monitors the output to extract the device code, copies it to the clipboard, and opens the authentication URL in the default browser.

**Value**

Invisibly returns the exit status (0 for success, non-zero for failure)

**Examples**

```
## Not run:  
# Perform Azure CLI login with device code flow  
az_cli_login()  
  
# Use the bridge webpage for easier code handling  
az_cli_login(use_bridge = TRUE)  
  
# Login to a specific tenant with verbose output  
az_cli_login(tenant_id = "your-tenant-id", verbose = TRUE)  
  
## End(Not run)
```

---

az_cli_logout	<i>Azure CLI Logout</i>
---------------	-------------------------

---

**Description**

Logs out from Azure CLI by removing all stored credentials and account information. This function runs `az logout`.

**Usage**

```
az_cli_logout()
```

**Details**

After logging out, you will need to run `az_cli_login()` again to authenticate and use Azure CLI credentials.

**Value**

Invisibly returns NULL

**Examples**

```
## Not run:  
# Log out from Azure CLI  
az_cli_logout()  
  
## End(Not run)
```

---

CachedTokenCredential	<i>Cached token credential authentication</i>
-----------------------	---

---

**Description**

A credential class that retrieves tokens from the cache only, without triggering interactive authentication flows. This is useful for non-interactive sessions where you want to use previously cached tokens from DeviceCode or AuthCode credentials.

**Details**

This credential attempts to retrieve cached tokens from a chain of interactive credentials (AuthCode and DeviceCode by default). It will not prompt for new authentication - it only returns tokens that are already cached.

This is particularly useful for:

- Non-interactive R sessions (e.g., scheduled scripts, CI/CD)
- Scenarios where you've previously authenticated interactively and want to reuse those cached tokens

**Public fields**

- .scope Character string specifying the authentication scope.
- .tenant\_id Character string specifying the tenant ID.
- .client\_id Character string specifying the client ID.
- .chain List of credential classes to attempt for cached tokens.

**Active bindings**

provider Lazily initialized credential provider

**Methods****Public methods:**

- [CachedTokenCredential\\$new\(\)](#)
- [CachedTokenCredential\\$get\\_token\(\)](#)
- [CachedTokenCredential\\$req\\_auth\(\)](#)
- [CachedTokenCredential\\$clone\(\)](#)

**Method** `new()`: Create a new `CachedTokenCredential` object

*Usage:*

```
CachedTokenCredential$new(  
  scope = NULL,  
  tenant_id = NULL,  
  client_id = NULL,  
  chain = cached_token_credential_chain()  
)
```

*Arguments:*

scope Optional character string specifying the authentication scope.

tenant\_id Optional character string specifying the tenant ID for authentication.

client\_id Optional character string specifying the client ID for authentication.

chain A list of credential classes to attempt for cached tokens. Defaults to `AuthCodeCredential` and `DeviceCodeCredential`.

*Returns:* A new `CachedTokenCredential` object

**Method** `get_token()`: Get an access token from the cache

*Usage:*

```
CachedTokenCredential$get_token()
```

*Returns:* An `httr2::oauth_token()` object containing the access token

**Method** `req_auth()`: Add authentication to an `httr2` request

*Usage:*

```
CachedTokenCredential$req_auth(req)
```

*Arguments:*

req An `httr2::request()` object

*Returns:* The request object with authentication configured

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
CachedTokenCredential$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

## Examples

```
## Not run:
# Create credential with default settings
cred <- CachedTokenCredential$new(
  scope = "https://graph.microsoft.com/.default",
  tenant_id = "my-tenant-id"
)

# Get a cached token (will fail if no cached token exists)
token <- cred$get_token()

# Use with httr2 request
req <- httr2::request("https://graph.microsoft.com/v1.0/me")
req <- cred$req_auth(req)

## End(Not run)
```

---

cached\_token\_credential\_chain

*Create Cached Token Credential Chain*

---

## Description

Creates the default chain of credentials to attempt for cached token retrieval. The credentials are tried in order until one returns a valid cached token. The default chain includes:

1. Authorization Code Credential - Cached tokens from browser-based authentication
2. Device Code Credential - Cached tokens from device code flow
3. Azure CLI Credential - Cached tokens from Azure CLI authentication

## Usage

```
cached_token_credential_chain()
```

## Value

A `credential_chain` object containing the sequence of credential providers to check for cached tokens.

**See Also**

[CachedTokenCredential](#), [credential\\_chain\(\)](#)

---

ClientSecretCredential

*Client secret credential authentication*

---

**Description**

Authenticates a service principal using a client ID and client secret. This credential is commonly used for application authentication in Azure.

**Details**

The credential uses the OAuth 2.0 client credentials flow to obtain access tokens. It requires a registered Azure AD application with a client secret. The client secret should be stored securely and not hard-coded in scripts.

**Super class**

`azr::Credential -> ClientSecretCredential`

**Methods****Public methods:**

- [ClientSecretCredential\\$validate\(\)](#)
- [ClientSecretCredential\\$get\\_token\(\)](#)
- [ClientSecretCredential\\$req\\_auth\(\)](#)
- [ClientSecretCredential\\$clone\(\)](#)

**Method** `validate()`: Validate the credential configuration

*Usage:*

`ClientSecretCredential$validate()`

*Details:* Checks that the client secret is provided and not NA or NULL. Calls the parent class validation method.

**Method** `get_token()`: Get an access token using client credentials flow

*Usage:*

`ClientSecretCredential$get_token()`

*Returns:* An `httr2::oauth_token()` object containing the access token

**Method** `req_auth()`: Add OAuth client credentials authentication to an httr2 request

*Usage:*

`ClientSecretCredential$req_auth(req)`

*Arguments:*

req An `httr2::request()` object

*Returns:* The request object with OAuth client credentials authentication configured

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
ClientSecretCredential$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

**Examples**

```
# Create credential with client secret
cred <- ClientSecretCredential$new(
  tenant_id = "your-tenant-id",
  client_id = "your-client-id",
  client_secret = "your-client-secret",
  scope = "https://management.azure.com/.default"
)

# To get a token or authenticate a request it requires
# valid 'client_id' and 'client_secret' credentials,
# otherwise it will return an error.
## Not run:
# Get an access token
token <- cred$get_token()

# Use with httr2 request
req <- httr2::request("https://management.azure.com/subscriptions")
resp <- httr2::req_perform(cred$req_auth(req))

## End(Not run)
```

---

credential\_chain

*Create Custom Credential Chain*

---

**Description**

Creates a custom chain of credential providers to attempt during authentication. Credentials are tried in the order they are provided until one successfully authenticates. This allows you to customize the authentication flow beyond the default credential chain.

**Usage**

```
credential_chain(...)
```

### Arguments

... Named credential objects or credential classes. Each element should be a credential class (e.g., `ClientSecretCredential`) or an instantiated credential object that inherits from the `Credential` base class. The names are used for identification purposes.

### Value

A `credential_chain` object containing the specified sequence of credential providers.

### See Also

[default\\_credential\\_chain\(\)](#), [get\\_token\\_provider\(\)](#)

### Examples

```
# Create a custom chain with only non-interactive credentials
custom_chain <- credential_chain(
  client_secret = ClientSecretCredential,
  azure_cli = AzureCLICredential
)

# Use the custom chain to get a token
## Not run:
token <- get_token(
  scope = "https://graph.microsoft.com/.default",
  chain = custom_chain
)

## End(Not run)
```

---

DefaultCredential      *Default credential authentication*

---

### Description

An R6 class that provides lazy initialization of credential providers. The credential provider is created on first access using the default credential chain.

### Details

This class wraps the credential discovery process in an R6 object with a lazily evaluated provider field. The provider is only created when first accessed, using the same logic as [get\\_token\\_provider\(\)](#).

**Public fields**

- .scope Character string specifying the authentication scope.
- .tenant\_id Character string specifying the tenant ID.
- .client\_id Character string specifying the client ID.
- .client\_secret Character string specifying the client secret.
- .use\_cache Character string indicating the caching strategy.
- .offline Logical indicating whether to request offline access.
- .chain A credential chain object for authentication.

**Active bindings**

provider Lazily initialized credential provider

**Methods****Public methods:**

- [DefaultCredential\\$new\(\)](#)
- [DefaultCredential\\$get\\_token\(\)](#)
- [DefaultCredential\\$req\\_auth\(\)](#)
- [DefaultCredential\\$clone\(\)](#)

**Method** `new()`: Create a new DefaultCredential object

*Usage:*

```
DefaultCredential$new(
  scope = NULL,
  tenant_id = NULL,
  client_id = NULL,
  client_secret = NULL,
  use_cache = "disk",
  offline = TRUE,
  chain = default_credential_chain()
)
```

*Arguments:*

scope Optional character string specifying the authentication scope.

tenant\_id Optional character string specifying the tenant ID for authentication.

client\_id Optional character string specifying the client ID for authentication.

client\_secret Optional character string specifying the client secret for authentication.

use\_cache Character string indicating the caching strategy. Defaults to "disk". Options include "disk" for disk-based caching or "memory" for in-memory caching.

offline Logical. If TRUE, adds 'offline\_access' to the scope to request a 'refresh\_token'. Defaults to TRUE.

chain A list of credential objects, where each element must inherit from the Credential base class. Credentials are attempted in the order provided until get\_token succeeds.

*Returns:* A new DefaultCredential object

**Method** `get_token()`: Get an access token using the credential chain

*Usage:*

```
DefaultCredential$get_token()
```

*Returns:* An `httr2::oauth_token()` object containing the access token

**Method** `req_auth()`: Add authentication to an `httr2` request

*Usage:*

```
DefaultCredential$req_auth(req)
```

*Arguments:*

`req` An `httr2::request()` object

*Returns:* The request object with authentication configured

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
DefaultCredential$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
# Create a DefaultCredential object
cred <- DefaultCredential$new(
  scope = "https://graph.microsoft.com/.default",
  tenant_id = "my-tenant-id"
)

## Not run:
# Get a token (triggers lazy initialization)
token <- cred$get_token()

# Authenticate a request
req <- httr2::request("https://management.azure.com/subscriptions")
resp <- httr2::req_perform(cred$req_auth(req))

# Or access the provider directly
provider <- cred$provider

## End(Not run)
```

---

default\_azure\_client\_id  
*Get default Azure client ID*

---

**Description**

Retrieves the Azure client ID from the AZURE\_CLIENT\_ID environment variable, or falls back to the default Azure CLI client ID if not set.

**Usage**

```
default_azure_client_id()
```

**Value**

A character string with the client ID

**Examples**

```
default_azure_client_id()
```

---

default\_azure\_client\_secret  
*Get default Azure client secret*

---

**Description**

Retrieves the Azure client secret from the AZURE\_CLIENT\_SECRET environment variable, or returns NA\_character\_ if not set.

**Usage**

```
default_azure_client_secret()
```

**Value**

A character string with the client secret, or NA\_character\_ if not set

**Examples**

```
default_azure_client_secret()
```

---

`default_azure_config_dir`*Get default Azure configuration directory*

---

**Description**

Retrieves the Azure configuration directory from the AZURE\_CONFIG\_DIR environment variable, or falls back to the platform-specific default.

**Usage**`default_azure_config_dir()`**Value**

A character string with the Azure configuration directory path

**Examples**`default_azure_config_dir()`

---

`default_azure_host`*Get default Azure authority host*

---

**Description**

Retrieves the Azure authority host from the AZURE\_AUTHORITY\_HOST environment variable, or falls back to Azure Public Cloud if not set.

**Usage**`default_azure_host()`**Value**

A character string with the authority host URL

**Examples**`default_azure_host()`

---

default\_azure\_oauth\_client  
*Create default Azure OAuth client*

---

### Description

Creates an `httr2::oauth_client()` configured for Azure authentication.

### Usage

```
default_azure_oauth_client(  
  client_id = default_azure_client_id(),  
  client_secret = NULL,  
  name = NULL  
)
```

### Arguments

`client_id` A character string specifying the client ID. Defaults to `default_azure_client_id()`.  
`client_secret` A character string specifying the client secret. Defaults to `NULL`.  
`name` A character string specifying the client name. Defaults to `NULL`.

### Value

An `httr2::oauth_client()` object

### Examples

```
client <- default_azure_oauth_client()  
client <- default_azure_oauth_client(  
  client_id = "my-client-id",  
  client_secret = "my-secret"  
)
```

---

default\_azure\_scope *Get default Azure OAuth scope*

---

### Description

Returns the default OAuth scope for a specified Azure resource.

### Usage

```
default_azure_scope(resource = "azure_arm")
```

**Arguments**

resource      A character string specifying the Azure resource. Must be one of: "azure\_arm" (Azure Resource Manager), "azure\_graph" (Microsoft Graph), "azure\_storage" (Azure Storage), or "azure\_key\_vault" (Azure Key Vault). Defaults to "azure\_arm".

**Value**

A character string with the OAuth scope URL

**Examples**

```
default_azure_scope()  
default_azure_scope("azure_graph")
```

---

default\_azure\_tenant\_id

*Get default Azure tenant ID*

---

**Description**

Retrieves the Azure tenant ID from the AZURE\_TENANT\_ID environment variable, or falls back to the default value if not set.

**Usage**

```
default_azure_tenant_id()
```

**Value**

A character string with the tenant ID

**Examples**

```
default_azure_tenant_id()
```

---

default_azure_url	<i>Get default Azure OAuth URLs</i>
-------------------	-------------------------------------

---

### Description

Constructs Azure OAuth 2.0 endpoint URLs for a given tenant and authority host.

### Usage

```
default_azure_url(  
    endpoint = NULL,  
    oauth_host = default_azure_host(),  
    tenant_id = default_azure_tenant_id()  
)
```

### Arguments

endpoint	A character string specifying which endpoint URL to return. Must be one of: "authorize", "token", or "devicecode". If NULL (default), returns a list of all endpoint URLs.
oauth_host	A character string specifying the Azure authority host. Defaults to <code>default_azure_host()</code> .
tenant_id	A character string specifying the tenant ID. Defaults to <code>default_azure_tenant_id()</code> .

### Value

If endpoint is specified, returns a character string with the URL. If endpoint is NULL, returns a named list of all endpoint URLs.

### Examples

```
# Get all URLs  
default_azure_url()  
  
# Get specific endpoint  
default_azure_url("token")  
  
# Custom tenant  
default_azure_url("authorize", tenant_id = "my-tenant-id")
```

---

default\_credential\_chain

*Create Default Credential Chain*

---

### Description

Creates the default chain of credentials to attempt during authentication. The credentials are tried in order until one successfully authenticates. The default chain includes:

1. Client Secret Credential - Uses client ID and secret
2. Authorization Code Credential - Interactive browser-based authentication
3. Azure CLI Credential - Uses credentials from Azure CLI
4. Device Code Credential - Interactive device code flow

### Usage

```
default_credential_chain()
```

### Value

A credential\_chain object containing the default sequence of credential providers.

### See Also

[credential\\_chain\(\)](#), [get\\_token\\_provider\(\)](#)

---

default\_non\_auth

*Default No Authentication*

---

### Description

A pass-through credential function that performs no authentication. This function returns the request object unchanged, allowing API calls to be made without adding any authentication headers or tokens.

### Usage

```
default_non_auth(req)
```

### Arguments

req            An [httr2::request\(\)](#) object

### Value

The same [httr2::request\(\)](#) object, unmodified

---

default\_redirect\_uri *Get default OAuth redirect URI*

---

**Description**

Constructs a redirect URI for OAuth flows. If the provided URI doesn't have a port, assigns a random port using [httpuv::randomPort\(\)](#).

**Usage**

```
default_redirect_uri(redirect_uri = httr2::oauth_redirect_uri())
```

**Arguments**

redirect\_uri A character string specifying the redirect URI. Defaults to [httr2::oauth\\_redirect\\_uri\(\)](#).

**Value**

A character string with the redirect URI

**Examples**

```
default_redirect_uri()
```

---

default\_refresh\_token *Get default Azure refresh token*

---

**Description**

Retrieves the Azure refresh token from the AZURE\_REFRESH\_TOKEN environment variable, or returns NULL if not set.

**Usage**

```
default_refresh_token()
```

**Value**

A character string with the refresh token, or NULL if not set

**Examples**

```
default_refresh_token()
```

default\_response\_handler

*Default Response Handler*

---

### Description

Default callback function for processing API response content. This function converts data frames within lists to `data.table` objects for better performance and functionality, if the `data.table` package is available.

### Usage

```
default_response_handler()
```

### Details

The function recursively processes list responses and converts any `data.frame` objects to `data.table` objects using `data.table::as.data.table()`, but only if the `data.table` package is installed. If `data.table` is not available, data frames are returned unchanged. Non-`data.frame` elements are always returned unchanged.

### Value

A function that accepts parsed response content and returns processed content

### Examples

```
# Get the default handler
handler <- default_response_handler()

# Use with a custom handler
custom_handler <- function(content) {
  # Your custom processing logic
  content
}
```

---

DeviceCodeCredential *Device code credential authentication*

---

### Description

Authenticates a user through the device code flow. This flow is designed for devices that don't have a web browser or have input constraints.

## Details

The device code flow displays a code that the user must enter on another device with a web browser to complete authentication. This is ideal for CLI applications, headless servers, or devices without a browser.

The credential supports token caching to avoid repeated authentication. Tokens can be cached to disk or in memory.

## Super classes

`azr::Credential` -> `azr::InteractiveCredential` -> `DeviceCodeCredential`

## Methods

### Public methods:

- `DeviceCodeCredential$new()`
- `DeviceCodeCredential$get_token()`
- `DeviceCodeCredential$req_auth()`
- `DeviceCodeCredential$clone()`

**Method** `new()`: Create a new device code credential

*Usage:*

```
DeviceCodeCredential$new(  
  scope = NULL,  
  tenant_id = NULL,  
  client_id = NULL,  
  use_cache = "disk",  
  offline = TRUE,  
  interactive = TRUE  
)
```

*Arguments:*

`scope` A character string specifying the OAuth2 scope. Defaults to NULL.

`tenant_id` A character string specifying the Azure Active Directory tenant ID. Defaults to NULL.

`client_id` A character string specifying the application (client) ID. Defaults to NULL.

`use_cache` A character string specifying the cache type. Use "disk" for disk-based caching or "memory" for in-memory caching. Defaults to "disk".

`offline` A logical value indicating whether to request offline access (refresh tokens). Defaults to TRUE.

`interactive` A logical value indicating whether this credential requires user interaction. Defaults to TRUE.

*Returns:* A new `DeviceCodeCredential` object

**Method** `get_token()`: Get an access token using device code flow

*Usage:*

```
DeviceCodeCredential$get_token(reauth = FALSE)
```

*Arguments:*

reauth A logical value indicating whether to force reauthentication. Defaults to FALSE.

*Returns:* An `httr2::oauth_token()` object containing the access token

**Method** `req_auth()`: Add OAuth device code authentication to an `httr2` request

*Usage:*

```
DeviceCodeCredential$req_auth(req)
```

*Arguments:*

req An `httr2::request()` object

*Returns:* The request object with OAuth device code authentication configured

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
DeviceCodeCredential$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

**Examples**

```
# DeviceCodeCredential requires an interactive session
## Not run:
# Create credential with default settings
cred <- DeviceCodeCredential$new()

# Get an access token (will prompt for 'device code' flow)
token <- cred$get_token()

# Force re-authentication
token <- cred$get_token(reauth = TRUE)

# Use with httr2 request
req <- httr2::request("https://management.azure.com/subscriptions")
req <- cred$req_auth(req)

## End(Not run)
```

---

get\_credential\_auth    *Get Credential Authentication Function*

---

**Description**

Creates a function that retrieves authentication tokens and formats them as HTTP Authorization headers. This function handles credential discovery and returns a callable method that generates Bearer token headers when invoked.

**Usage**

```
get_credential_auth(
  scope = NULL,
  tenant_id = NULL,
  client_id = NULL,
  client_secret = NULL,
  use_cache = "disk",
  offline = TRUE,
  chain = default_credential_chain()
)
```

**Arguments**

scope	Optional character string specifying the authentication scope.
tenant_id	Optional character string specifying the tenant ID for authentication.
client_id	Optional character string specifying the client ID for authentication.
client_secret	Optional character string specifying the client secret for authentication.
use_cache	Character string indicating the caching strategy. Defaults to "disk". Options include "disk" for disk-based caching or "memory" for in-memory caching.
offline	Logical. If TRUE, adds 'offline_access' to the scope to request a 'refresh_token'. Defaults to TRUE.
chain	A list of credential objects, where each element must inherit from the <code>Credential</code> base class. Credentials are attempted in the order provided until <code>get_token</code> succeeds.

**Value**

A function that, when called, returns a named list with an `Authorization` element containing the Bearer token, suitable for use with `httr2::req_headers()`.

**See Also**

[get\\_token\(\)](#), [get\\_request\\_authorizer\(\)](#), [get\\_token\\_provider\(\)](#)

**Examples**

```
## Not run:
# Create an authentication function
auth_fn <- get_credential_auth(
  scope = "https://graph.microsoft.com/.default"
)

# Call it to get headers
auth_headers <- auth_fn()

# Use with httr2
req <- httr2::request("https://graph.microsoft.com/v1.0/me") |>
  httr2::req_headers(!!!auth_headers)
```

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

---

```
get_credential_provider
    Get Credential Provider
```

---

### Description

Discovers and returns an authenticated credential object from a chain of credential providers. This function attempts each credential in the chain until one successfully authenticates, returning the first successful credential object.

### Usage

```
get_credential_provider(
  scope = NULL,
  tenant_id = NULL,
  client_id = NULL,
  client_secret = NULL,
  use_cache = "disk",
  offline = TRUE,
  oauth_host = NULL,
  oauth_endpoint = NULL,
  chain = NULL,
  interactive = TRUE,
  verbose = getOption("azr.verbose", FALSE)
)
```

### Arguments

scope	Optional character string specifying the authentication scope.
tenant_id	Optional character string specifying the tenant ID for authentication.
client_id	Optional character string specifying the client ID for authentication.
client_secret	Optional character string specifying the client secret for authentication.
use_cache	Character string indicating the caching strategy. Defaults to "disk". Options include "disk" for disk-based caching or "memory" for in-memory caching.
offline	Logical. If TRUE, adds 'offline_access' to the scope to request a 'refresh_token'. Defaults to TRUE.
oauth_host	Optional character string specifying the OAuth host URL.
oauth_endpoint	Optional character string specifying the OAuth endpoint.
chain	A list of credential objects, where each element must inherit from the <code>Credential</code> base class. Credentials are attempted in the order provided until <code>get_token</code> succeeds. If NULL, uses <code>default_credential_chain()</code> .

interactive	A logical value indicating whether interactive credentials are allowed. Defaults to TRUE.
verbose	A logical value indicating whether to print verbose messages during credential discovery. Defaults to <code>getOption("azr.verbose", FALSE)</code> .

**Value**

A credential object that inherits from the `Credential` class and has successfully authenticated.

**See Also**

[get\\_token\\_provider\(\)](#), [get\\_request\\_authorizer\(\)](#), [default\\_credential\\_chain\(\)](#)

**Examples**

```
## Not run:
# Get a credential provider with default settings
cred <- get_credential_provider(
  scope = "https://graph.microsoft.com/.default",
  tenant_id = "my-tenant-id"
)

# Use the credential to get a token
token <- cred$get_token()

## End(Not run)
```

---

get\_request\_authorizer

*Get Default Request Authorizer Function*

---

**Description**

Creates a request authorizer function that retrieves authentication credentials and returns a callable request authorization method. This function handles the credential discovery process and returns the request authentication method from the discovered credential object.

**Usage**

```
get_request_authorizer(
  scope = NULL,
  tenant_id = NULL,
  client_id = NULL,
  client_secret = NULL,
  use_cache = "disk",
  offline = TRUE,
  chain = default_credential_chain()
)
```

**Arguments**

scope	Optional character string specifying the authentication scope.
tenant_id	Optional character string specifying the tenant ID for authentication.
client_id	Optional character string specifying the client ID for authentication.
client_secret	Optional character string specifying the client secret for authentication.
use_cache	Character string indicating the caching strategy. Defaults to "disk". Options include "disk" for disk-based caching or "memory" for in-memory caching.
offline	Logical. If TRUE, adds 'offline_access' to the scope to request a 'refresh_token'. Defaults to TRUE.
chain	A list of credential objects, where each element must inherit from the <code>Credential</code> base class. Credentials are attempted in the order provided until <code>get_token</code> succeeds.

**Value**

A function that authorizes HTTP requests with appropriate credentials when called.

**See Also**

[get\\_token\\_provider\(\)](#), [get\\_token\(\)](#)

**Examples**

```
# In non-interactive sessions, this function will return an error if the
# environment is not setup with valid credentials. And in an interactive session
# the user will be prompted to attempt one of the interactive authentication flows.
## Not run:
req_auth <- get_request_authorizer(
  scope = "https://graph.microsoft.com/.default"
)
req <- req_auth(httr2::request("https://graph.microsoft.com/v1.0/me"))

## End(Not run)
```

---

get\_token

*Get Authentication Token*

---

**Description**

Retrieves an authentication token using the default token provider. This is a convenience function that combines credential discovery and token acquisition in a single step.

**Usage**

```
get_token(
  scope = NULL,
  tenant_id = NULL,
  client_id = NULL,
  client_secret = NULL,
  use_cache = "disk",
  offline = TRUE,
  chain = default_credential_chain()
)
```

**Arguments**

scope	Optional character string specifying the authentication scope.
tenant_id	Optional character string specifying the tenant ID for authentication.
client_id	Optional character string specifying the client ID for authentication.
client_secret	Optional character string specifying the client secret for authentication.
use_cache	Character string indicating the caching strategy. Defaults to "disk". Options include "disk" for disk-based caching or "memory" for in-memory caching.
offline	Logical. If TRUE, adds 'offline_access' to the scope to request a 'refresh_token'. Defaults to TRUE.
chain	A list of credential objects, where each element must inherit from the <code>Credential</code> base class. Credentials are attempted in the order provided until <code>get_token</code> succeeds.

**Value**

An `httr2::oauth_token()` object.

**See Also**

[get\\_token\\_provider\(\)](#), [get\\_request\\_authorizer\(\)](#)

**Examples**

```
# In non-interactive sessions, this function will return an error if the
# environment is not setup with valid credentials. And in an interactive session
# the user will be prompted to attempt one of the interactive authentication flows.
## Not run:
token <- get_token(
  scope = "https://graph.microsoft.com/.default",
  tenant_id = "my-tenant-id",
  client_id = "my-client-id",
  client_secret = "my-secret"
)

## End(Not run)
```

---

get\_token\_provider      *Get Default Token Provider Function*

---

### Description

Creates a token provider function that retrieves authentication credentials and returns a callable token getter. This function handles the credential discovery process and returns the token acquisition method from the discovered credential object.

### Usage

```
get_token_provider(  
    scope = NULL,  
    tenant_id = NULL,  
    client_id = NULL,  
    client_secret = NULL,  
    use_cache = "disk",  
    offline = TRUE,  
    chain = default_credential_chain()  
)
```

### Arguments

scope	Optional character string specifying the authentication scope.
tenant_id	Optional character string specifying the tenant ID for authentication.
client_id	Optional character string specifying the client ID for authentication.
client_secret	Optional character string specifying the client secret for authentication.
use_cache	Character string indicating the caching strategy. Defaults to "disk". Options include "disk" for disk-based caching or "memory" for in-memory caching.
offline	Logical. If TRUE, adds 'offline_access' to the scope to request a 'refresh_token'. Defaults to TRUE.
chain	A list of credential objects, where each element must inherit from the <code>Credential</code> base class. Credentials are attempted in the order provided until <code>get_token</code> succeeds.

### Value

A function that retrieves and returns an authentication token when called.

### See Also

[get\\_request\\_authorizer\(\)](#), [get\\_token\(\)](#)

## Examples

```
# In non-interactive sessions, this function will return an error if the
# environment is not set up with valid credentials. In an interactive session
# the user will be prompted to attempt one of the interactive authentication flows.
## Not run:
token_provider <- get_token_provider(
  scope = "https://graph.microsoft.com/.default",
  tenant_id = "my-tenant-id",
  client_id = "my-client-id",
  client_secret = "my-secret"
)
token <- token_provider()

## End(Not run)
```

---

is_hosted_session	<i>Detect if running in a hosted session</i>
-------------------	--

---

## Description

Determines whether the current R session is running in a hosted environment such as Google Colab or RStudio Server (non-localhost).

## Usage

```
is_hosted_session()
```

## Details

This function checks for:

- Google Colab: presence of the COLAB\_RELEASE\_TAG environment variable
- RStudio Server: RSTUDIO\_PROGRAM\_MODE is "server" and RSTUDIO\_HTTP\_REFERER does not contain "localhost"

## Value

A logical value: TRUE if running in a hosted session (Google Colab or remote RStudio Server), FALSE otherwise.

## Examples

```
if (is_hosted_session()) {
  message("Running in a hosted environment")
}
```

---

RefreshTokenCredential

*Refresh token credential authentication*

---

### Description

Authenticates using an existing refresh token. This credential is useful when you have obtained a refresh token through another authentication flow and want to use it to get new access tokens without interactive authentication.

### Details

The refresh token credential uses the OAuth 2.0 refresh token flow to obtain new access tokens. It requires a valid refresh token that was previously obtained through an interactive flow (e.g., authorization code or device code).

This is particularly useful for:

- Non-interactive sessions where you have a pre-obtained refresh token
- Long-running applications that need to refresh tokens automatically
- Scenarios where you want to avoid repeated interactive authentication

### Super class

azr::Credential -> RefreshTokenCredential

### Public fields

.refresh\_token Character string containing the refresh token.

### Methods

#### Public methods:

- [RefreshTokenCredential\\$new\(\)](#)
- [RefreshTokenCredential\\$validate\(\)](#)
- [RefreshTokenCredential\\$get\\_token\(\)](#)
- [RefreshTokenCredential\\$req\\_auth\(\)](#)
- [RefreshTokenCredential\\$clone\(\)](#)

**Method new():** Create a new refresh token credential

*Usage:*

```
RefreshTokenCredential$new(  
  refresh_token = default_refresh_token(),  
  scope = NULL,  
  tenant_id = NULL,  
  client_id = NULL  
)
```

*Arguments:*

`refresh_token` A character string containing the refresh token. Defaults to `default_refresh_token()` which reads from the `AZURE_REFRESH_TOKEN` environment variable.

`scope` A character string specifying the OAuth2 scope. Defaults to `NULL`.

`tenant_id` A character string specifying the Azure Active Directory tenant ID. Defaults to `NULL`.

`client_id` A character string specifying the application (client) ID. Defaults to `NULL`.

*Returns:* A new `RefreshTokenCredential` object

**Method** `validate()`: Validate the credential configuration

*Usage:*

```
RefreshTokenCredential$validate()
```

*Details:* Checks that the refresh token is provided and not `NA` or `NULL`. Calls the parent class validation method.

**Method** `get_token()`: Get an access token using the refresh token flow

*Usage:*

```
RefreshTokenCredential$get_token()
```

*Returns:* An `httr2::oauth_token()` object containing the access token

**Method** `req_auth()`: Add OAuth refresh token authentication to an `httr2` request

*Usage:*

```
RefreshTokenCredential$req_auth(req)
```

*Arguments:*

`req` An `httr2::request()` object

*Returns:* The request object with OAuth refresh token authentication configured

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
RefreshTokenCredential$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

**Examples**

```
## Not run:
# Create credential with a refresh token
cred <- RefreshTokenCredential$new(
  refresh_token = "your-refresh-token",
  scope = "https://management.azure.com/.default",
  tenant_id = "your-tenant-id",
  client_id = "your-client-id"
)

# Get an access token
```

```
token <- cred$get_token()

# Use with httr2 request
req <- httr2::request("https://management.azure.com/subscriptions")
resp <- httr2::req_perform(cred$req_auth(req))

## End(Not run)
```

# Index

api\_client, [2](#), [9–11](#), [14](#), [15](#)  
api\_graph\_resource, [15](#)  
api\_resource, [7](#), [9](#), [15](#)  
api\_service, [9](#), [14](#), [15](#)  
api\_storage\_client, [10](#), [16](#)  
AuthCodeCredential, [12](#)  
az\_cli\_account\_show, [19](#)  
az\_cli\_get\_cached\_token, [20](#)  
az\_cli\_get\_token, [22](#)  
az\_cli\_is\_login, [23](#)  
az\_cli\_login, [24](#)  
az\_cli\_login(), [25](#)  
az\_cli\_logout, [25](#)  
azr::api\_client, [10](#)  
azr::InteractiveCredential, [12](#), [17](#), [41](#)  
azr\_graph\_client, [14](#)  
azr\_storage\_client, [16](#)  
AzureCLICredential, [17](#)

cached\_token\_credential\_chain, [27](#)  
CachedTokenCredential, [25](#), [28](#)  
ClientSecretCredential, [28](#)  
credential\_chain, [9](#), [11](#), [14](#), [16](#), [29](#)  
credential\_chain(), [28](#), [38](#)

data.table::as.data.table(), [40](#)  
default\_azure\_client\_id, [33](#)  
default\_azure\_client\_id(), [35](#)  
default\_azure\_client\_secret, [33](#)  
default\_azure\_config\_dir, [34](#)  
default\_azure\_config\_dir(), [21](#)  
default\_azure\_host, [34](#)  
default\_azure\_host(), [37](#)  
default\_azure\_oauth\_client, [35](#)  
default\_azure\_scope, [35](#)  
default\_azure\_tenant\_id, [36](#)  
default\_azure\_tenant\_id(), [16](#), [37](#)  
default\_azure\_url, [37](#)  
default\_credential\_chain, [38](#)  
default\_credential\_chain(), [16](#), [30](#), [44](#),  
[45](#)  
default\_non\_auth, [38](#)  
default\_non\_auth(), [4](#)  
default\_redirect\_uri, [39](#)  
default\_redirect\_uri(), [13](#)  
default\_refresh\_token, [39](#)  
default\_refresh\_token(), [51](#)  
default\_response\_handler, [40](#)  
default\_response\_handler(), [4](#)  
DefaultCredential, [11](#), [14](#), [30](#)  
DeviceCodeCredential, [40](#)  
get\_credential\_auth, [42](#)  
get\_credential\_provider, [44](#)  
get\_request\_authorizer, [45](#)  
get\_request\_authorizer(), [43](#), [45](#), [47](#), [48](#)  
get\_token, [46](#)  
get\_token(), [43](#), [46](#), [48](#)  
get\_token\_provider, [48](#)  
get\_token\_provider(), [30](#), [38](#), [43](#), [45–47](#)  
httpuv::randomPort(), [39](#)  
httr2::oauth\_client(), [35](#)  
httr2::oauth\_redirect\_uri(), [39](#)  
httr2::oauth\_token(), [6](#), [13](#), [18](#), [20–22](#), [26](#),  
[28](#), [32](#), [42](#), [47](#), [51](#)  
httr2::req\_headers(), [43](#)  
httr2::req\_perform(), [4](#), [6](#)  
httr2::request(), [5](#), [6](#), [13](#), [18](#), [27](#), [29](#), [32](#),  
[38](#), [42](#), [51](#)  
httr2::response(), [5](#), [6](#)  
is\_hosted\_session, [49](#)  
lockEnvironment(), [15](#)  
RefreshTokenCredential, [50](#)  
rlang::englue(), [4](#), [5](#)  
tempfile(), [11](#)