Package 'azr'

November 13, 2026

Title Credential Chain for OAuth 2.0 Authentication to 'Azure'

Version 0.3.7

Description Implements a credential chain for 'Azure' OAuth 2.0 authentication, built on the OAuth framework of the 'httr2' package. Tries authentication methods in turn until one succeeds, so the same code runs on a laptop, in continuous integration and on an 'Azure' host. Supports client secret, workload identity, managed identity and 'Azure CLI' credentials, plus interactive device code and authorization code flows during development.

License MIT + file LICENSE

Encoding UTF-8

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, S7 (>= 0.2.0), cli, httr2 (>= 1.2.2), jsonlite, rlang

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

Config/testthat/edition 3

Config/Needs/website tidyverse/tidytemplate

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Pedro Baltazar [aut, cre, cph]

Maintainer Pedro Baltazar <pedrobtz@gmail.com>

Repository CRAN

Date/Publication 2026-11-13 11:20:02 UTC

Contents

api_client . . . 3
api_log_analytics_client . . . 8
api_resource . . . 11
api_service . . . 12
api_storage_client . . . 13
AuthCodeCredential . . . 16
azr_catalog . . . 18
azr_catalog_read . . . 19
azr_catalog_write . . . 20
azr_dataset . . . 20
azr_dataset_from_uri . . . 21
azr_dataset_manifest . . . 22
azr_dataset_uri . . . 23
azr_graph_client . . . 24
azr_logs_client . . . 25
azr_options . . . 27
azr_resolve_dataset . . . 28
azr_storage_client . . . 29
AzureCLICredential . . . 30
azure_spark_storage_conf . . . 33
az_cli_account_show . . . 35
az_cli_get_cached_token . . . 36
az_cli_get_token . . . 37
az_cli_is_login . . . 38
az_cli_login . . . 38
az_cli_logout . . . 40
az_config . . . 40
CachedTokenCredential . . . 41
cached_token_credential_chain . . . 43
ClientSecretCredential . . . 44
credential_chain . . . 45
DefaultCredential . . . 46
default_azure_client_id . . . 49
default_azure_client_secret . . . 49
default_azure_cli_client_id . . . 50
default_azure_config_dir . . . 50
default_azure_host . . . 51
default_azure_oauth_client . . . 51
default_azure_scope . . . 52
default_azure_tenant_id . . . 53
default_azure_url . . . 53
default_credential_chain . . . 54
default_federated_token_file . . . 55
default_graph_endpoint . . . 56
default_log_analytics_endpoint . . . 56
default_msal_token_cache . . . 57

default_non_auth . . . 57
default_redirect_uri . . . 58
default_refresh_token . . . 58
default_response_handler . . . 59
default_storage_endpoint . . . 59
DeviceCodeCredential . . . 60
get_credential_auth . . . 61
get_credential_provider . . . 63
get_request_authorizer . . . 64
get_token . . . 66
get_token_provider . . . 67
get_user_token . . . 68
is_hosted_session . . . 70
ManagedIdentityCredential . . . 71
parse_storage_path . . . 73
RefreshTokenCredential . . . 75
WorkloadIdentityCredential . . . 77
write_msal_token . . . 79

Index **81**

`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

`.verbose` Logical flag controlling request/response logging in `.send_request()`

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()

`api_client$new()`: Create a new API client instance

Usage:

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

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`.

`verbose` A logical flag controlling request/response logging in `.send_request()`. When `FALSE`, the `>>>` request and `<<<` response `cli` alerts are suppressed. Defaults to the `api_verbose` option, which reads `options(azr.api_verbose = ...)` or the `AZR_API_VERBOSE` environment variable; see `azr_options()`.

Returns: A new `api_client` object

`api_client$.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::englue()` syntax for variable interpolation using named arguments passed via `...`.

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

`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

`api_client$.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

`api_client$.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::englue()` syntax for variable interpolation using named arguments passed via `...`.

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

`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

`api_client$.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

`api_client$.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

`api_client$.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.

`api_client$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)
```

`api_log_analytics_client`

Azure Log Analytics API Class

Description

An R6 class that extends api_client to provide a Kusto Query Language (KQL) `query()` method against the public Azure Log Analytics REST API, bound to a specific Azure subscription and resource group at construction.

Details

The client is bound to `subscription_id` and `resource_id` (the resource group name) at construction. The `$query()` method issues a `POST` to `https://{endpoint}/{api_version}/subscriptions/{subscription_id}` with a JSON body (`{"query": ..., "timespan": ..., "workspaces": [...]}`).

Pass `scope = "hierarchy"` (or any other supported query-string parameter) via `...` on `$query()` to traverse the resource hierarchy.

Super class

`api_client` -> `api_log_analytics_client`

Public fields

`.subscription_id` The Azure subscription ID the client is bound to.

`.resource_id` The Azure resource group name the client is bound to.

`.api_version` The API version segment prepended to all query paths.

Methods

Public methods:

- api_log_analytics_client$new()
- api_log_analytics_client$query()
- api_log_analytics_client$clone()

`api_log_analytics_client$new()`: Create a new Azure Log Analytics API client instance bound to a specific subscription and resource group.

Usage:

```
api_log_analytics_client$new(
  subscription_id,
  resource_id,
  endpoint = default_log_analytics_endpoint(),
  api_version = "v1",
  scope = default_azure_scope("azure_log_analytics"),
  provider = NULL,
  chain = NULL,
  tenant_id = NULL,
  ...
)
```

Arguments:

`subscription_id` A character string specifying the Azure subscription ID (GUID) to bind the client to.

`resource_id` A character string specifying the Azure resource group name to bind the client to.

`endpoint` A character string specifying the Log Analytics query endpoint host (e.g. `"api.loganalytics.io"`). Defaults to `default_log_analytics_endpoint()`. Any leading `https?://` scheme or trailing slashes are stripped.

`api_version` A character string specifying the API version segment prepended to the query path. Defaults to `"v1"`.

`scope` A character string specifying the OAuth2 scope. Defaults to `default_azure_scope("azure_log_analytics")`.

`provider` An optional credential provider object that inherits from `Credential` or `DefaultCredential`. If provided, `chain` is ignored.

`chain` A credential_chain instance for authentication. If `NULL`, a default credential chain is 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_log_analytics_client` object

`api_log_analytics_client$query()`: Issue a KQL query against the bound subscription and resource group.

Usage:

```
api_log_analytics_client$query(
  query,
  date_from = Sys.Date() - 3,
  date_to = Sys.Date() + 1,
  timespan = NULL,
  max_rows = 500001L,
  options = list(truncationMaxSize = 67108864L),
  workspace_filters = list(regions = list()),
  ...,
  raw = FALSE,
  coerce_types = TRUE
)
```

Arguments:

`query` A character string containing the KQL query to execute.

`date_from` Start of the time range as a `Date` or `POSIXct`. When provided together with `date_to`, appends `| where TimeGenerated between(datetime(...), datetime(...))` to the query and sets `timespan` to `NULL`. Defaults to `Sys.Date() - 3`.

`date_to` End of the time range as a `Date` or `POSIXct`. Defaults to `Sys.Date() + 1`.

`timespan` An ISO 8601 duration (e.g. `"PT12H"`) or start/end pair separated by `/` (e.g. `"2024-01-01/2024-01-02"`). Passed as a URL query parameter. Ignored when `date_from` and `date_to` are set. Defaults to `NULL`.

`max_rows` Maximum number of rows to return. Defaults to `500001`.

`options` A named list of query options. Defaults to `list(truncationMaxSize = 67108864)`.

`workspace_filters` A named list of workspace filters. Defaults to `list(regions = list())`.

`...` Additional URL query parameters. Override defaults (e.g. `scope = "resource"` to change from the default `"hierarchy"`).

`raw` If `TRUE`, returns the parsed JSON response as a list. If `FALSE` (the default), returns a named list of `data.frames` — one per table in the response — or the single table directly if only one is returned.

`coerce_types` If `TRUE` (the default), columns are coerced to their native R types based on the Log Analytics schema (e.g. `datetime` → `POSIXct`, `bool` → `logical`). Set to `FALSE` to keep all values as character.

Returns: Either a single `data.frame`, a named list of `data.frames`, or the raw parsed response (when `raw = TRUE`).

`api_log_analytics_client$clone()`: The objects of this class are cloneable with this method.

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
la <- api_log_analytics_client$new(
  subscription_id = "00000000-0000-0000-0000-000000000000",
```

```
  resource_id = "my-resource-group"
)

la$query(
  query = "AzureDiagnostics | take 10",
  timespan = "PT12H",
  scope = "hierarchy"
)

## End(Not run)
```

`api_resource` *Azure API Resource*

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()`

`api_resource$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

`api_resource$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()

api_service$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}.{endpoint_suffix}

Super class

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()

`api_storage_client$new()`: Create a new Azure Storage API client instance

Usage:

```
api_storage_client$new(
  storageaccount,
  filesystem,
  scope = default_azure_scope("azure_storage"),
  endpoint_suffix = default_storage_endpoint(),
  provider = NULL,
  chain = NULL,
  tenant_id = NULL,
  client_id = default_azure_cli_client_id(),
  ...
)
```

Arguments:

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

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

`scope` A character string specifying the OAuth2 scope. Defaults to `default_azure_scope("azure_storage")`.

`endpoint_suffix` A character string specifying the Azure Storage DFS endpoint suffix. Defaults to `default_storage_endpoint()`.

`provider` An optional credential provider object that inherits from `Credential` or `DefaultCredential`. If provided, `chain` is ignored.

`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`.

`client_id` A character string specifying the Azure client ID. Passed to DefaultCredential when `chain` is `NULL`. Defaults to `default_azure_cli_client_id()`.

`...` Additional arguments passed to the parent api_client constructor.

Returns: A new `api_storage_client` object

`api_storage_client$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).

`api_storage_client$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.

`api_storage_client$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) with one row per file or directory. Columns include `name`, `contentLength`, `lastModified`, etc. All pages are fetched transparently; the result is the complete listing.

`api_storage_client$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

`Credential` -> `InteractiveCredential` -> `AuthCodeCredential`

Public fields

`.redirect_uri` Character string specifying the redirect URI registered with the application.

Methods

Public methods:

- `AuthCodeCredential$new()`
- `AuthCodeCredential$clone()`

`AuthCodeCredential$new()`**:** Create a new authorization code credential

Usage:

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

```
  use_refresh_token = TRUE,
  interactive = NULL
)
```

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 the Azure CLI public client ID.

`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()`.

`allow_prompt` A logical value indicating whether this credential may prompt the user (vs. only reading cached/refresh tokens). Defaults to `TRUE`.

`use_refresh_token` A logical value indicating whether to use the login flow (acquire tokens via refresh token exchange). Defaults to `TRUE`.

`interactive` Deprecated. Use `allow_prompt` instead.

Returns: A new `AuthCodeCredential` object

`AuthCodeCredential$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_catalog` *Azure Storage dataset catalog*

Description

An S7 class holding an ordered collection of azr_dataset objects with unique `names`.

A catalog can be indexed by dataset name with `[[`, and supports `names()` and `length()`.

Usage

```
azr_catalog(datasets = list())
```

Arguments

`datasets`	A list of azr_dataset objects.

Value

An `azr_catalog` S7 object.

Examples

```
ds <- azr_dataset(
  name = "orders",
  scheme = "abfss",
  container = "raw",
  storage = list(prod = "stprod001"),
  path = "sales/orders",
  format = "delta"
)
catalog <- azr_catalog(datasets = list(ds))

catalog[["orders"]]
names(catalog)
length(catalog)
```

azr_catalog_read *Read a dataset catalog from JSON*

Description

Reads a JSON file describing a collection of datasets and returns an azr_catalog.

The expected JSON shape:

```
{
  "datasets": [
    {
      "name": "sales_orders",
      "scheme": "abfss",
      "container": "raw",
      "storage": { "prod": "stprod001", "preprod": "stpreprod001" },
      "path": "sales/orders",
      "format": "delta"
    }
  ]
}
```

Usage

```
azr_catalog_read(json_file)
```

Arguments

json_file	Path to a JSON file.

Value

An azr_catalog object.

See Also

azr_catalog_write()

azr_catalog_write *Write a dataset catalog to JSON*

Description

Writes an azr_catalog to a JSON file in the shape expected by `azr_catalog_read()`.

Usage

```
azr_catalog_write(catalog, json_file)
```

Arguments

`catalog`	An azr_catalog object.
`json_file`	Path to write the JSON file to.

Value

`json_file`, invisibly.

See Also

`azr_catalog_read()`

azr_dataset *Azure Storage dataset*

Description

An S7 class representing an Azure Storage dataset bound to one or more storage accounts keyed by environment tier (e.g. `"prod"`, `"preprod"`).

Usage

```
azr_dataset(
  name = character(0),
  scheme = character(0),
  container = character(0),
  storage = list(),
  path = character(0),
  format = character(0),
  endpoint_suffix = "core.windows.net"
)
```

Arguments

name	Dataset name. Must match ^[a-z][a-z0-9_]*$.
scheme	Hadoop filesystem scheme: "abfss" or "wasbs".
container	Container (filesystem) name.
storage	Non-empty named list mapping tier name to storage account.
path	Path within the container, without leading or trailing /.
format	Dataset format: "delta", "parquet", "csv", "tsv", "json", "avro", "orc", or "text".
endpoint_suffix	Storage endpoint suffix. Defaults to "core.windows.net".

Details

Only the storage account varies by tier: container, path, endpoint_suffix, and scheme are shared across all tiers in storage. If an environment also needs a different container, path, or sovereign cloud, model it as a separate azr_dataset.

path must be non-empty, so a dataset cannot point at a container root.

Value

An azr_dataset S7 object.

---

azr_dataset_from_uri *Create an* azr_dataset *from a full Azure Storage URI*

Description

Parses an Azure Storage URI using parse_storage_path() and constructs an azr_dataset. The parsed storage account is bound to tier in storage.

Usage

```
azr_dataset_from_uri(
  uri,
  name = NULL,
  format = NULL,
  tier = opts$get("dataset_tier"),
  storage = NULL
)
```

Arguments

uri	Full Azure Storage URI, such as abfss://raw@account.dfs.core.windows.net/path or https://account.dfs.core.windows.net/raw/path.
name	Dataset name. If NULL (the default), derived from the last segment of the URI's path with any file extension removed, e.g. "orders" for .../sales/orders or .../sales/orders.parquet.
format	Dataset format. If NULL, inferred from the URI's file extension (e.g. .parquet, .csv) or _delta_log segment. Defaults to "delta" when uri looks like a directory. Errors only when uri has a file extension that maps to no known format; pass format explicitly then.
tier	Environment tier for the storage account parsed from uri. Defaults to the dataset_tier option (options(azr.dataset_tier = ...) or AZR_DATASET_TIER, default "prod"); see azr_options().
storage	Optional named list mapping additional tiers to storage accounts. The account from uri is bound to tier unless that key is already present.

Value

An azr_dataset object.

---

azr_dataset_manifest *Azure Storage dataset manifest*

Description

An S7 class representing the resolved information required by an external reader to load an Azure Storage dataset. Use as.list() to convert it to a plain R list.

Usage

```
azr_dataset_manifest(
  name = character(0),
  uri = character(0),
  format = character(0)
)
```

Arguments

name	Dataset name, carried over from the source azr_dataset.
uri	Resolved Azure Storage URI.
format	Dataset format. See azr_dataset for supported values.

Value

An azr_dataset_manifest S7 object.

azr_dataset_uri *Build a URI for an* azr_dataset *or look one up in an* azr_catalog

Description

Build a URI for an azr_dataset or look one up in an azr_catalog

Usage

```
azr_dataset_uri(x, ...)
```

Arguments

x	An azr_dataset or azr_catalog object.
...	Additional arguments passed to methods: tier Environment tier name (a key in the dataset's storage). Defaults to the dataset_tier option (options(azr.dataset_tier = ...) or AZR_DATASET_TIER, default "prod"); see azr_options(). uri_type URI type: "https" or "hadoop" (the Hadoop ABFS URI form scheme://container@accoun used by Spark, Flink, Trino, and any hadoop-azure consumer). name For an azr_catalog only: an optional character scalar selecting a single dataset by name. If omitted, URIs for every dataset are returned.

Value

For an azr_dataset, or an azr_catalog with name supplied, a character scalar URI. For an azr_catalog without name, a named character vector of URIs keyed by dataset name.

Examples

```
ds <- azr_dataset(
  name = "orders",
  scheme = "abfss",
  container = "raw",
  storage = list(prod = "stprod001"),
  path = "sales/orders",
  format = "delta"
)
azr_dataset_uri(ds, tier = "prod")

catalog <- azr_catalog(datasets = list(ds))
azr_dataset_uri(catalog, tier = "prod", name = "orders")
azr_dataset_uri(catalog, tier = "prod")
```

`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",
  endpoint = default_graph_endpoint(),
  ...,
  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 `https://graph.microsoft.com/{scopes}`.
`endpoint`	A character string specifying the Microsoft Graph endpoint host (e.g. `"graph.microsoft.com"`). Defaults to `default_graph_endpoint()`.
`...`	Additional arguments passed to the api_client constructor.
`chain`	A credential_chain instance for authentication. If NULL, a default credential chain will be created using DefaultCredential.

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_logs_client *Create an Azure Log Analytics Client*

Description

A convenience wrapper around api_log_analytics_client that creates a configured client for the Azure Log Analytics query REST API, bound to a specific subscription and resource group.

Usage

```
azr_logs_client(
  subscription_id,
  resource_id,
  endpoint = default_log_analytics_endpoint(),
  api_version = "v1",
  scope = default_azure_scope("azure_log_analytics"),
  provider = NULL,
  chain = default_credential_chain(),
```

```
  tenant_id = default_azure_tenant_id(),
  ...
)
```

Arguments

subscription_id	A character string specifying the Azure subscription ID (GUID) to bind the client to.
resource_id	A character string specifying the Azure resource group name to bind the client to.
endpoint	A character string specifying the Log Analytics query endpoint host. Defaults to default_log_analytics_endpoint().
api_version	A character string specifying the API version segment. Defaults to "v1".
scope	A character string specifying the OAuth2 scope. Defaults to default_azure_scope("azure_log_analy
provider	An optional credential provider object that inherits from Credential or DefaultCredential. If provided, chain is ignored.
chain	A credential_chain instance for authentication. Defaults to default_credential_chain().
tenant_id	A character string specifying the Azure tenant ID. Defaults to default_azure_tenant_id().
...	Additional arguments passed to the api_log_analytics_client constructor.

Value

An api_log_analytics_client object.

Examples

```
## Not run:
la <- azr_logs_client(
  subscription_id = "00000000-0000-0000-0000-000000000000",
  resource_id = "my-resource-group"
)

la$query(
  query = "AzureDiagnostics | take 10",
  timespan = "PT12H",
  scope = "hierarchy"
)

## End(Not run)
```

azr_options *List all azr options and their current values*

Description

Prints every azr option and invisibly returns a data.frame of the same information. The resolution order is: a value set for the session -> `options(azr.*)` -> the option's environment variable -> a built-in default.

Name	R option	Env variable	Default
"chain_verbose"	azr.chain_verbose	AZR_CHAIN_VERBOSE	FALSE
"api_verbose"	azr.api_verbose	AZR_API_VERBOSE	FALSE
"cli_auto_login"	azr.cli_auto_login	AZR_CLI_AUTO_LOGIN	FALSE
"dataset_tier"	azr.dataset_tier	AZR_DATASET_TIER	"prod"
"managed_identity_client_id"	azr.managed_identity_client_id	AZR_MANAGED_IDENTITY_CLIENT_ID	NULL

Usage

```
azr_options(mask = TRUE)
```

Arguments

mask	Logical. When TRUE (default), sensitive option values are shown as "<hidden>" when set.

Value

Invisibly, a data.frame with columns `option`, `value`, `source`, `env_var`, `env_value`, and `default`.

Examples

```
azr_options()

# Set for the session
options(azr.chain_verbose = TRUE)

# Or via environment variable (before starting R)
# AZR_CHAIN_VERBOSE=true
```

azr_resolve_dataset *Build a URI + format manifest for an* azr_dataset *or* azr_catalog

Description

Like azr_dataset_uri(), but each entry also carries the dataset's format, which together are what a reader (e.g. sparklyr::spark_read_source()) needs to load a dataset.

Usage

```
azr_resolve_dataset(x, ...)
```

Arguments

x	An azr_dataset or azr_catalog object.
...	Additional arguments passed to methods:
	tier Environment tier name (a key in the dataset's storage). Defaults to the dataset_tier option (options(azr.dataset_tier = ...) or AZR_DATASET_TIER, default "prod"); see azr_options().
	uri_type URI type: "https" or "hadoop" (the Hadoop ABFS URI form scheme://container@accoun used by Spark, Flink, Trino, and any hadoop-azure consumer).
	name For an azr_catalog only: an optional character scalar selecting a single dataset by name. If omitted, URIs for every dataset are returned.

Value

For an azr_dataset, or an azr_catalog with name supplied, an azr_dataset_manifest. For an azr_catalog without name, a named list of azr_dataset_manifest objects, keyed by dataset name.

Examples

```
ds <- azr_dataset(
  name = "orders",
  scheme = "abfss",
  container = "raw",
  storage = list(prod = "stprod001"),
  path = "sales/orders",
  format = "delta"
)
azr_resolve_dataset(ds, tier = "prod")

catalog <- azr_catalog(datasets = list(ds))
azr_resolve_dataset(catalog, tier = "prod")
```

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,
  endpoint_suffix = default_storage_endpoint(),
  scope = default_azure_scope("azure_storage"),
  provider = NULL,
  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.
endpoint_suffix	A character string specifying the Azure Storage DFS endpoint suffix. Defaults to default_storage_endpoint().
scope	A character string specifying the OAuth2 scope. Defaults to default_azure_scope("azure_storage").
provider	An optional credential provider object that inherits from Credential or DefaultCredential. If provided, chain is ignored.
chain	A credential_chain instance for authentication. Defaults to default_credential_chain().
tenant_id	A character string specifying the Azure tenant ID. Defaults to default_azure_tenant_id(), which reads AZURE_TENANT_ID from the environment.
...	Additional arguments passed to the api_storage_client 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 class

`Credential` -> `AzureCLICredential`

Public fields

`auto_login` Logical indicating whether to check login status and perform login if needed

`use_bridge` Logical indicating whether to use the device code bridge webpage during interactive login

`.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$login()`
- `AzureCLICredential$is_interactive()`
- `AzureCLICredential$logout()`

- AzureCLICredential$clone()

`AzureCLICredential$new()`: Create a new Azure CLI credential

Usage:

```
AzureCLICredential$new(
  scope = NULL,
  tenant_id = NULL,
  process_timeout = NULL,
  auto_login = opts$get("cli_auto_login"),
  use_bridge = FALSE,
  interactive = NULL
)
```

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`.

`auto_login` A logical value indicating whether `get_token()` may launch `az login` when the user is not logged in. Defaults to the `cli_auto_login` option (`options(azr.cli_auto_login = ...)` or `AZR_CLI_AUTO_LOGIN`); see azr_options().

`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 `auto_login = TRUE`. Defaults to `FALSE`. Ignored inside VS Code, which cannot open the local page.

`interactive` Deprecated. Use `auto_login` instead.

Returns: A new `AzureCLICredential` object

`AzureCLICredential$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

`AzureCLICredential$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

`AzureCLICredential$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

`AzureCLICredential$login()`: Perform Azure CLI login using device code flow

Usage:

```
AzureCLICredential$login(force = FALSE)
```

Arguments:

force If FALSE (default), an existing login is reused. If TRUE, always runs `az login`.

Returns: The value of `az_cli_login()`, invisibly.

`AzureCLICredential$is_interactive()`: Check if the credential requires user interaction

Usage:

```
AzureCLICredential$is_interactive()
```

Returns: Logical indicating whether this credential is interactive

`AzureCLICredential$logout()`: Log out from Azure CLI

Usage:

```
AzureCLICredential$logout()
```

Returns: Invisibly returns NULL

`AzureCLICredential$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

```
# 'az login' must have been executed successfully for these examples to work.
## Not run:
# 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"
)

# 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)
```

`azure_spark_storage_conf` *Azure Storage Spark / Hadoop configuration*

Description

Builds the named list of Hadoop `fs.azure.*` configuration keys required to authenticate Apache Spark (or any ABFS-compatible runtime) against Azure Data Lake Storage Gen2. With the default `prefix = "spark.hadoop"` the returned list is ready to pass to `SparkSession.builder.config()`, a Databricks cluster's Spark config, or `spark-defaults.conf` — Spark forwards `spark.hadoop.*` keys to Hadoop at FileSystem-init time. Pass `prefix = NULL` to get the raw `fs.azure.*` form, suitable for `core-site.xml` or a `sparkContext.hadoopConfiguration().set(...)` call.

Five authentication types are supported:

`"client_secret"` Service principal with client secret (`ClientCredsTokenProvider`).

`"refresh_token"` Delegated user identity via a refresh token (`RefreshTokenBasedTokenProvider`).

`"workload_identity"` Kubernetes workload identity via a projected service-account token file (`WorkloadIdentityTokenProvider`). Requires Hadoop 3.4.1+ / 3.5.0+ (HADOOP-18610). Stock Apache Spark 3.5 ships with Hadoop 3.3.4, so this requires Spark 4.x or a runtime that bundles a newer Hadoop (Databricks 14.3+ LTS, Synapse 3.4+).

`"managed_identity"` Azure managed identity via IMDS (`MsiTokenProvider`). For Azure VMs, App Service, Functions, Container Instances, and AKS pods without workload identity. Pass `client_id` to select a user-assigned identity.

`"shared_key"` Storage account key (`SharedKey` auth type). Requires `storage_account` and `account_key`; cannot be configured globally.

Sovereign clouds (Azure US Government, Azure China) are supported by passing the matching `authority_host` (e.g. `"login.microsoftonline.us"`). The storage endpoint suffix used to scope keys to a specific account is derived from the authority host. Alternatively, pass a fully qualified `storage_account` like `"myacct.dfs.core.usgovcloudapi.net"` to override the derivation entirely.

Usage

```
azure_spark_storage_conf(
  auth_type = c("refresh_token", "client_secret", "workload_identity",
    "managed_identity", "shared_key"),
  storage_account = NULL,
  tenant_id = default_azure_tenant_id(),
  client_id = default_azure_client_id(),
  client_secret = default_azure_client_secret(),
  refresh_token = default_refresh_token(),
  account_key = NULL,
  token_file = default_federated_token_file(),
  authority_host = default_azure_host(),
  prefix = "spark.hadoop"
)
```

Arguments

auth_type	Authentication type. One of "refresh_token" (default), "client_secret", "workload_identity", "managed_identity", or "shared_key".
storage_account	Optional storage account, either a short name ("myacct") or a fully qualified host ("myacct.dfs.core.windows.net"). When NULL (default) the keys are applied globally to all accounts. When a short name is supplied the endpoint suffix is derived from authority_host. Required for auth_type = "shared_key".
tenant_id	Azure tenant ID. Defaults to default_azure_tenant_id().
client_id	Azure application (client) ID. Defaults to default_azure_client_id(). For auth_type = "managed_identity" this selects a user-assigned identity (omit or leave at default for system-assigned).
client_secret	Client secret. Required when auth_type = "client_secret". Defaults to default_azure_client_sec
refresh_token	Refresh token. Required when auth_type = "refresh_token". Defaults to default_refresh_token().
account_key	Storage account access key. Required when auth_type = "shared_key".
token_file	Path to the federated service-account token file. Used only when auth_type = "workload_identity". Defaults to default_federated_token_file() (AZURE_FEDERATED_TOKEN_FI
authority_host	Azure authority host (without scheme), e.g. "login.microsoftonline.com". Used to build the OAuth token endpoint for client_secret and refresh_token, the MSI authority for workload_identity and managed_identity, and to derive the storage endpoint suffix for sovereign clouds. Defaults to default_azure_host().
prefix	Optional prefix prepended to every returned key. Defaults to "spark.hadoop", the prefix Spark uses to forward properties to Hadoop. Pass NULL to get the raw fs.azure.* keys (e.g. for core-site.xml or a hadoopConfiguration() call). A trailing dot in prefix is optional.

Value

A named list of class `azure_spark_config`. With the default `prefix`, keys look like `spark.hadoop.fs.azure.account.*`; with `prefix = NULL`, keys look like `fs.azure.account.*`. The `print` method redacts sensitive values (account key, client secret, refresh token).

Examples

```
# Global client-secret config (applies to all storage accounts)
azure_spark_storage_conf(
  auth_type = "client_secret",
  tenant_id = "my-tenant",
  client_id = "my-client",
  client_secret = "my-secret"
)

# Scoped to a specific storage account
azure_spark_storage_conf(
  auth_type = "client_secret",
  storage_account = "mystorageaccount",
  tenant_id = "my-tenant",
  client_id = "my-client",
  client_secret = "my-secret"
)

# Azure US Government sovereign cloud
azure_spark_storage_conf(
  auth_type = "client_secret",
  storage_account = "mystorageaccount",
  tenant_id = "my-tenant",
  client_id = "my-client",
  client_secret = "my-secret",
  authority_host = "login.microsoftonline.us"
)

## Not run:
# Workload identity on AKS (reads env vars automatically)
azure_spark_storage_conf(auth_type = "workload_identity")

# Managed identity on an Azure VM (system-assigned)
azure_spark_storage_conf(auth_type = "managed_identity")

## 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

---

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 default_azure_config_dir().

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

---

az_cli_get_token *Get Access Token from Azure CLI*

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

---

az_cli_is_login *Check if User is Logged in to Azure CLI*

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

---

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,
  force = FALSE
)
```

Arguments

tenant_id	A character string specifying the Azure Active Directory tenant ID to authenticate against. If NULL (default), uses the default tenant from Azure CLI configuration.
use_bridge	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.
verbose	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.
force	A logical value. If FALSE (default), an existing Azure CLI login is reused and no new login is started. If TRUE, always runs `az login`, for example to switch accounts.

Details

Unless `force = TRUE`, the function first runs `az account show` and, when the Azure CLI is already logged in (to `tenant_id`, if given), returns that account without starting a new login. Otherwise it 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, the current account from `az account show` when an existing login is reused; otherwise the accounts reported by `az login` on success, or its non-zero exit status on failure.

Examples

```
# Reuses the current login if there is one
az_cli_login()

# Always start a new login
az_cli_login(force = TRUE)
```

az_cli_logout *Azure CLI Logout*

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`

az_config *Report the Azure environment configuration*

Description

Reports the Azure environment variables used during credential discovery, the value each one resolves to, and where that value came from. This is the same information shown in the package startup banner.

`AZURE_CLIENT_SECRET` is never reported verbatim: when it is set, its value is returned as `<<REDACTED>>`.

Usage

```
az_config()
```

Value

A `data.frame` with one row per environment variable and the columns:

`variable` Name of the environment variable.

`value` The resolved value, or `NA` when the variable is unset and has no built-in fallback.

`source` Where the value came from: `"env"` when read from the environment, `"default"` when it falls back to a built-in value, or `"unset"` when the variable is not set and has no fallback.

See Also

default_azure_tenant_id(), default_azure_client_id(), default_azure_config_dir()

Examples

```
az_config()

# which variables are actually set in the environment?
config <- az_config()
config[config$source == "env", ]
```

CachedTokenCredential *Cached token credential authentication*

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()

`CachedTokenCredential$new()`: Create a new CachedTokenCredential object

Usage:

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

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

`CachedTokenCredential$get_token()`: Get an access token from the cache

Usage:

```
CachedTokenCredential$get_token()
```

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

`CachedTokenCredential$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

`CachedTokenCredential$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(scope = NULL, tenant_id = NULL, client_id = NULL)
```

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.

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

`Credential` -> `ClientSecretCredential`

Methods

Public methods:

- `ClientSecretCredential$validate()`
- `ClientSecretCredential$get_token()`
- `ClientSecretCredential$req_auth()`
- `ClientSecretCredential$clone()`

`ClientSecretCredential$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.

`ClientSecretCredential$get_token()`: Get an access token using client credentials flow

Usage:

```
ClientSecretCredential$get_token()
```

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

`ClientSecretCredential$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

`ClientSecretCredential$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 chain entries. Each entry must be either a credential class (e.g., `ClientSecretCredential`) or an already-constructed object that inherits from the `Credential` base class. Class entries receive the context passed to `get_credential_provider()`. Constructed instances are used as-is. The names are used for identification purposes. Constructing a chain performs no authentication.

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.

`.verbose` Logical indicating whether to print the resolved provider class.

Active bindings

`provider` Lazily initialized credential provider

Methods

Public methods:

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

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

Usage:

```
DefaultCredential$new(
  scope = NULL,
  tenant_id = NULL,
  client_id = NULL,
  client_secret = NULL,
  use_cache = c("disk", "memory"),
  offline = TRUE,
  chain = default_credential_chain(),
  verbose = opts$get("chain_verbose")
)
```

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.

`verbose` Logical. If `TRUE`, prints the resolved credential provider on first access. Defaults to the `chain_verbose` option (`options(azr.chain_verbose = ...)` or `AZR_CHAIN_VERBOSE`); see `azr_options()`.

Returns: A new `DefaultCredential` object

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

Usage:

```
DefaultCredential$get_token()
```

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

`DefaultCredential$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

`DefaultCredential$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 in priority order:

1. `AZURE_CLIENT_ID` environment variable
2. Built-in fallback (Microsoft's public Azure CLI client ID)

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_cli_client_id

Get the Azure CLI public client ID

Description

Returns Microsoft's public Azure CLI client ID (`04b07795-8ddb-461a-bbee-02f9e1bf7b46`). This is the default `client_id` used by interactive credentials when no application-specific client ID is configured.

Usage

```
default_azure_cli_client_id()
```

Value

A character string with the Azure CLI client ID

Examples

```
default_azure_cli_client_id()
```

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 in priority order:

1. `AZURE_AUTHORITY_HOST` environment variable
2. Built-in fallback (`login.microsoftonline.com`)

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. Accepts both the full name (e.g. `"azure_arm"`) and the short name without the `azure_` prefix (e.g. `"arm"`). Must be one of: `"azure_arm"` / `"arm"` (Azure Resource Manager), `"azure_graph"` / `"graph"` (Microsoft Graph), `"azure_storage"` / `"storage"` (Azure Storage), `"azure_key_vault"` / `"key_vault"` (Azure Key Vault), `"azure_openai"` / `"openai"` (Azure OpenAI), `"azure_log_analytics"` / `"log_analytics"` (Azure Log Analytics), `"azure_app_insights"` / `"app_insights"` (Azure Application Insights), `"azure_databricks"` / `"databricks"` (Azure Databricks), `"azure_sql"` / `"sql"` (Azure SQL / Synapse), or `"azure_service_bus"` / `"service_bus"` (Azure Service Bus). Defaults to `"azure_arm"`.

Value

A character string with the OAuth scope URL

Examples

```
default_azure_scope()
default_azure_scope("azure_graph")
default_azure_scope("graph")
default_azure_scope("storage")
```

default_azure_tenant_id *Get default Azure tenant ID*

Description

Retrieves the Azure tenant ID in priority order:

1. `AZURE_TENANT_ID` environment variable
2. Built-in fallback (`"common"`)

Usage

```
default_azure_tenant_id()
```

Value

A character string with the tenant ID

Examples

```
default_azure_tenant_id()
```

default_azure_url *Get default Azure OAuth URLs*

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 `default_azure_host()`.
`tenant_id`	A character string specifying the tenant ID. Defaults to `default_azure_tenant_id()`.

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. Workload Identity Credential - Federated token, e.g. an AKS pod
3. Managed Identity Credential - The host's managed identity
4. Azure CLI Credential - Uses credentials from Azure CLI
5. Authorization Code Credential - Interactive browser-based authentication
6. Device Code Credential - Interactive device code flow

Usage

```
default_credential_chain()
```

Value

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

Narrowing the chain with `AZURE_TOKEN_CREDENTIALS`

Like `DefaultAzureCredential` in the Azure SDKs, the chain honours the `AZURE_TOKEN_CREDENTIALS` environment variable, read each time the chain is built. Matching is case-insensitive:

Value	Entries kept
`"dev"`	`azure_cli`, `auth_code`, `device_code`

"prod"	client_secret, workload_identity, managed_identi
"EnvironmentCredential" or "ClientSecretCredential"	client_secret
"WorkloadIdentityCredential"	workload_identity
"ManagedIdentityCredential"	managed_identity
"AzureCliCredential"	azure_cli
"InteractiveBrowserCredential" or "AuthCodeCredential"	auth_code
"DeviceCodeCredential"	device_code

Setting "dev" is how to keep a person's Azure CLI login from losing to a workload or managed identity the host also offers. Unset or empty keeps the whole chain. Any other value is an error, including the Azure SDK names for credentials azr does not implement (AzureDeveloperCliCredential, AzurePowerShellCredential, VisualStudioCodeCredential). Chains built with credential_chain() are never filtered.

See Also

credential_chain(), get_token_provider()

---

default_federated_token_file

Get default federated token file path

Description

Retrieves the path to the federated identity token file from the AZURE_FEDERATED_TOKEN_FILE environment variable, or returns NULL if not set. Used by WorkloadIdentityCredential.

Usage

```
default_federated_token_file()
```

Value

A character string with the file path, or NULL if not set

Examples

```
default_federated_token_file()
```

default_graph_endpoint

Get default Microsoft Graph endpoint

Description

Returns the default host used to construct Microsoft Graph API URLs (`graph.microsoft.com`).

Usage

```
default_graph_endpoint()
```

Value

A character string with the Microsoft Graph endpoint host.

Examples

```
default_graph_endpoint()
```

default_log_analytics_endpoint

Get default Azure Log Analytics query endpoint

Description

Returns the default host used to construct Azure Log Analytics query URLs (`api.loganalytics.io`).

Usage

```
default_log_analytics_endpoint()
```

Value

A character string with the Log Analytics query endpoint host.

Examples

```
default_log_analytics_endpoint()
```

default_msal_token_cache *Get default MSAL token cache path*

Description

Returns the path to the MSAL token cache file shared by the Azure CLI and Azure SDKs. Defaults to `msal_token_cache.json` inside the Azure config directory (see `default_azure_config_dir()`).

Usage

```
default_msal_token_cache()
```

Value

A character string with the path to the MSAL token cache file.

See Also

`default_azure_config_dir()`, `write_msal_token()`

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

Converts `data.frame` results in the parsed response to `data.table` objects when the `data.table` package is available. Applied automatically by api_client unless overridden via the `response_handler` argument.

Usage

```
default_response_handler(content)
```

Arguments

`content`	Parsed response content from an API call.

Value

The processed content, with any `data.frame` objects converted to `data.table` if the `data.table` package is installed.

default_storage_endpoint

Get default Azure Storage DFS endpoint suffix

Description

Returns the default endpoint suffix used to construct Azure Data Lake Storage Gen2 DFS URLs.

Usage

```
default_storage_endpoint()
```

Value

A character string with the DFS endpoint suffix.

Examples

```
default_storage_endpoint()
```

`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

`Credential` -> `InteractiveCredential` -> `DeviceCodeCredential`

Methods

Public methods:

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

`DeviceCodeCredential$new()`**:** Create a new device code credential

Usage:

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

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 the Azure CLI public client ID.

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.

allow_prompt A logical value indicating whether this credential may prompt the user (vs. only reading cached/refresh tokens). Defaults to TRUE.

use_refresh_token A logical value indicating whether to use the login flow (acquire tokens via refresh token exchange). Defaults to TRUE.

interactive Deprecated. Use allow_prompt instead.

Returns: A new DeviceCodeCredential object

DeviceCodeCredential$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 Credential base class. Credentials are attempted in the order provided until get_token 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,
  allow_interactive = rlang::is_interactive(),
  verbose = opts$get("chain_verbose"),
  interactive = NULL
)
```

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 `Credential` base class. Credentials are attempted in the order provided until `get_token` succeeds. If `NULL`, uses `default_credential_chain()`.

allow_interactive
: A logical value indicating whether interactive credentials are allowed. Defaults to rlang::is_interactive().

verbose
: A logical value indicating whether to print verbose messages during credential discovery. Defaults to the chain_verbose option, which reads options(azr.chain_verbose = ...) or the AZR_CHAIN_VERBOSE environment variable; see azr_options().

interactive
: Deprecated. Use allow_interactive instead.

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 Credential base class. Credentials are attempted in the order provided until get_token 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 Credential base class. Credentials are attempted in the order provided until get_token succeeds.

Value

An httr2::oauth_token() object.

See Also

get_user_token(), 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 `Credential` base class. Credentials are attempted in the order provided until `get_token` 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)
```

---

get_user_token *Get an Access Token for the Signed-In User*

Description

Retrieves an authentication token for the person signed in on this machine, skipping the service identities that `get_token()` would try first.

Usage

```
get_user_token(
  scope = NULL,
  tenant_id = NULL,
  client_id = default_azure_cli_client_id(),
  use_cache = "disk",
  offline = TRUE
)
```

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 to authenticate against. Defaults to default_azure_cli_client_id(), the public Azure CLI client, rather than to AZURE_CLIENT_ID: on a host configured for a service identity that variable holds the *workload's* application, which is not the application a person signs in to.
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.

Details

get_token() walks the whole default_credential_chain(), where workload_identity and managed_identity come before azure_cli. On a host configured for a service identity - an AKS pod setting AZURE_FEDERATED_TOKEN_FILE, a CI runner with OIDC - that means get_token() hands back the *workload's* identity even when a person is sitting at the console. get_user_token() restricts the chain to the credentials that represent a person, tried in the same order the default chain uses them:

1. Azure CLI Credential - the current `az login` session
2. Authorization Code Credential - interactive browser sign-in
3. Device Code Credential - interactive device code flow

The two interactive flows are skipped in non-interactive sessions, so on an unattended host this resolves to the Azure CLI login or fails. client_secret is deliberately absent from the arguments: it authenticates an application, not a user.

Value

An httr2::oauth_token() object.

See Also

get_token(), az_cli_get_token(), default_credential_chain()

Examples

```
## Not run:
scope <- "https://management.azure.com/.default"

# on a host configured for workload identity, this is the workload
get_token(scope = scope)

# ...while this is whoever ran `az login`
get_user_token(scope = scope)
```

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

`is_hosted_session` *Detect if running in a hosted session*

Description

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

Usage

```
is_hosted_session()
```

Details

This function checks for (in order):

- Option override: if `azr.hosted` option is set, returns `isTRUE()` of its value
- Google Colab: presence of the `COLAB_RELEASE_TAG` environment variable
- VS Code: presence of the `VSCODE_INJECTION` or `VSCODE_PROXY_URI` environment variable
- Kubernetes: presence of the `KUBERNETES_SERVICE_HOST` 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, VS Code, Kubernetes, or remote RStudio Server), `FALSE` otherwise.

Examples

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

`ManagedIdentityCredential` *Managed identity credential authentication*

Description

Authenticates using an Azure managed identity. Supports both system-assigned and user-assigned managed identities. This credential works when code is running inside an Azure environment that has a managed identity configured. No credentials need to be stored — the identity is granted by the Azure platform.

Details

The token endpoint depends on the host, which the credential detects from environment variables the platform sets, in the same order as the Azure SDKs:

Source	Hosts	Detected by
`"app_service"`	App Service, Azure Functions, Container Apps	`IDENTITY_ENDPOINT` and `IDENTITY_HEADE`
`"azure_arc"`	Azure Arc-enabled servers	`IDENTITY_ENDPOINT` and `IMDS_ENDPOINT`
`"app_service_2017"`	App Service (legacy API)	`MSI_ENDPOINT` and `MSI_SECRET`
`"cloud_shell"`	Azure Cloud Shell	`MSI_ENDPOINT` alone
`"imds"`	VMs, VM scale sets, Container Instances, AKS nodes	none of the above

The Instance Metadata Service (IMDS) at `http://169.254.169.254/metadata/identity/oauth2/token` is the fallback. Service Fabric (`IDENTITY_SERVER_THUMBPRINT` alongside `IDENTITY_ENDPOINT` and `IDENTITY_HEADER`) is detected but not supported: `get_token()` fails with a classed error and the credential chain moves on. Azure Arc and Cloud Shell support only the system-assigned identity.

The detected source is stored in the `.msi_source` field and shown by `print()` and in verbose credential-chain output.

To use a **system-assigned** managed identity, leave `managed_identity_client_id` as `NULL`. To use a **user-assigned** managed identity, supply its client ID as `managed_identity_client_id`, or set the `managed_identity_client_id` option (`options(azr.managed_identity_client_id = ...)` or `AZR_MANAGED_IDENTITY_CLIENT_ID`) so the credential chain in `get_token()` picks it up.

A user-assigned identity is a different principal from an app registration, so the `client_id` given to `get_token()` and friends is not forwarded to this credential: it identifies the app registration that ClientSecretCredential and the interactive credentials authenticate as.

Against IMDS this credential fails fast (2-second timeout) when not running inside Azure, so it is safe to include early in a credential chain.

Super class

`Credential` -> `ManagedIdentityCredential`

Public fields

`.msi_client_id` Client ID for user-assigned managed identity, or `NULL` for system-assigned.

`.msi_source` Token source detected from the environment: one of `"app_service"`, `"azure_arc"`, `"app_service_2017"`, `"cloud_shell"`, `"service_fabric"` or `"imds"`.

Methods

Public methods:

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

`ManagedIdentityCredential$new()`: Create a new managed identity credential

Usage:

```
ManagedIdentityCredential$new(
  scope = NULL,
  managed_identity_client_id = opts$get("managed_identity_client_id"),
  ...
)
```

Arguments:

`scope` A character string specifying the OAuth2 scope. Defaults to the Azure Resource Manager scope.

`managed_identity_client_id` A character string specifying the client ID of a user-assigned managed identity. Defaults to the `managed_identity_client_id` option; when that is unset, `NULL` selects the system-assigned managed identity.

`...` Accepts the deprecated `client_id` argument, an alias for `managed_identity_client_id`. It is taken through `...` so that the credential chain, which forwards only named constructor arguments, never passes an app registration's client ID here.

Returns: A new `ManagedIdentityCredential` object

`ManagedIdentityCredential$get_token()`: Get an access token from the managed identity endpoint

Usage:

```
ManagedIdentityCredential$get_token()
```

Details: Returns a valid in-object cached token immediately if one exists. Otherwise requests a new token from the endpoint of the detected source (see Details above).

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

`ManagedIdentityCredential$req_auth()`: Add managed identity authentication to an httr2 request

Usage:

```
ManagedIdentityCredential$req_auth(req)
```

Arguments:

req An httr2::request() object

Returns: The request object with a Bearer token authorization header

`ManagedIdentityCredential$clone()`: The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

Examples

```
## Not run:
# System-assigned managed identity
cred <- ManagedIdentityCredential$new(
  scope = "https://management.azure.com/.default"
)

# User-assigned managed identity
cred <- ManagedIdentityCredential$new(
  scope = "https://management.azure.com/.default",
  managed_identity_client_id = "your-user-assigned-client-id"
)

token <- cred$get_token()

## End(Not run)
```

---

parse_storage_path *Parse an Azure Storage path*

Description

Splits an Azure Storage URL or Hadoop filesystem path into its constituent parts. Supports all common Azure Storage path formats:

`abfss://` / `abfs://` Azure Data Lake Storage Gen2 (DFS endpoint), used by Spark / Hadoop.

`wasbs://` / `wasb://` Legacy Azure Blob filesystem scheme, used by older Spark / Hadoop integrations.

`https://` / `http://` Standard Azure Blob or DFS REST endpoint, optionally with a SAS token query string.

`az://` / `azure://` Non-standard aliases used by some Python tools (e.g. `adlfs`); parsed using the same `container@account.host/path` shape as `abfs://`.

The `format` field is inferred from the path on a best-effort basis: `"delta"` when the path contains `_delta_log`; a file extension name (`"parquet"`, `"csv"`, `"json"`, `"avro"`, `"orc"`, `"text"`) when the last path segment has a recognised extension; `"folder"` when there is no extension; and `NA` for unrecognised extensions.

Usage

```
parse_storage_path(path)
```

Arguments

path	A character string containing the Azure Storage path to parse.

Value

An azure_storage_path object (a named list) with the fields:

scheme URL scheme, e.g. "abfss", "wasbs", "https".

storage_account Storage account name.

endpoint Storage endpoint type: "dfs" or "blob".

endpoint_suffix Host suffix after the endpoint label, e.g. "core.windows.net" (Azure public), "core.usgovcloudapi.net" (US Government), "core.chinacloudapi.cn" (China), or "storage.azure.net" (DNS-zone endpoints). Use this to distinguish sovereign clouds from the public cloud.

container Container name. Called "filesystem" in ADLS Gen2 / ABFS contexts; both refer to the same underlying object.

path Path within the container, without a leading /. Empty string if the URL points to the container root.

format Inferred dataset or file format (see above).

query Named list of query parameters (e.g. a parsed SAS token), or NULL if none.

original The original input string.

Examples

```
parse_storage_path(
  "abfss://mycontainer@myaccount.dfs.core.windows.net/data/sales/2024"
)

parse_storage_path(
  "https://myaccount.blob.core.windows.net/mycontainer/data/events.parquet"
)

parse_storage_path(
  "wasbs://mycontainer@myaccount.blob.core.windows.net/data/delta_table"
)
```

`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

`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()`

`RefreshTokenCredential$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

`RefreshTokenCredential$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.

`RefreshTokenCredential$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

`RefreshTokenCredential$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

`RefreshTokenCredential$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)
```

`WorkloadIdentityCredential` *Workload Identity credential authentication*

Description

Authenticates using Azure Workload Identity by reading a federated token from a file and exchanging it for an Azure AD access token. This is commonly used in Kubernetes environments (AKS) where a service account token is mounted into the pod.

Details

The credential implements the OAuth 2.0 client credentials flow with a JWT bearer assertion (`client_assertion`). It reads the federated identity token from a file on each call to `get_token()` so that token rotation by the runtime (e.g., Kubernetes) is automatically picked up.

The following environment variables are used when parameters are not provided:

- `AZURE_CLIENT_ID`: Client (application) ID of the Azure AD application
- `AZURE_TENANT_ID`: Azure AD tenant ID
- `AZURE_FEDERATED_TOKEN_FILE`: Path to the file containing the federated token

Super class

`Credential` -> `WorkloadIdentityCredential`

Public fields

`.token_file_path` Path to the file containing the federated identity token

Methods

Public methods:

- `WorkloadIdentityCredential$new()`
- `WorkloadIdentityCredential$validate()`
- `WorkloadIdentityCredential$get_token()`
- `WorkloadIdentityCredential$req_auth()`
- `WorkloadIdentityCredential$clone()`

`WorkloadIdentityCredential$new()`: Create a new Workload Identity credential

Usage:

```
WorkloadIdentityCredential$new(
  scope = NULL,
  tenant_id = Sys.getenv(environment_variables$azure_tenant_id, unset = NA_character_),
  client_id = Sys.getenv(environment_variables$azure_client_id, unset = NA_character_),
  token_file_path = default_federated_token_file()
)
```

Arguments:

`scope` A character string specifying the OAuth2 scope. Defaults to the Azure Resource Manager scope.

`tenant_id` A character string specifying the Azure AD tenant ID. Defaults to the `AZURE_TENANT_ID` environment variable.

`client_id` A character string specifying the client (application) ID. Defaults to the `AZURE_CLIENT_ID` environment variable.

`token_file_path` A character string specifying the path to the file containing the federated identity token. Defaults to the `AZURE_FEDERATED_TOKEN_FILE` environment variable.

Returns: A new `WorkloadIdentityCredential` object

`WorkloadIdentityCredential$validate()`: Validate the credential configuration

Usage:

```
WorkloadIdentityCredential$validate()
```

Details: Checks that `token_file_path` is provided and not NA. Calls the parent class validation method.

`WorkloadIdentityCredential$get_token()`: Get an access token by exchanging the federated token

Usage:

```
WorkloadIdentityCredential$get_token()
```

Details: Returns a valid in-object cached token immediately if one exists. Otherwise reads the federated token from the file and exchanges it for a new access token so that token rotation performed by the runtime is automatically reflected.

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

`WorkloadIdentityCredential$req_auth()`: Add authentication to an httr2 request

Usage:

```
WorkloadIdentityCredential$req_auth(req)
```

Arguments:

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

Returns: The request object with a Bearer token authorization header

`WorkloadIdentityCredential$clone()`: The objects of this class are cloneable with this method.

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
# Create credential using environment variables
# (requires AZURE_CLIENT_ID, AZURE_TENANT_ID, AZURE_FEDERATED_TOKEN_FILE)
cred <- WorkloadIdentityCredential$new(
  scope = "https://management.azure.com/.default"
)

# Or supply parameters directly
cred <- WorkloadIdentityCredential$new(
  tenant_id = "your-tenant-id",
  client_id = "your-client-id",
  token_file_path = "/var/run/secrets/azure/tokens/azure-identity-token",
  scope = "https://management.azure.com/.default"
)

# 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)
```

---

write_msal_token *Write an httr2 Token to the MSAL Token Cache*

Description

Writes an `httr2::oauth_token()` object into the MSAL token cache JSON file (`msal_token_cache.json`) shared by the Azure SDK and Azure CLI. The resulting entry is readable by other Azure tools (Python SDK, Azure CLI, and the rest of this package via `az_cli_get_cached_token()`).

Usage

```
write_msal_token(token, cache_file = default_msal_token_cache())
```

Arguments

token	An `httr2::oauth_token()` object. Must contain `access_token`, `token_type`, and `.expires_at`. May optionally contain `refresh_token` and `scope`. All cache fields (`home_account_id`, `tenant_id`, `username`, `client_id`, `scope`, `environment`) are derived from the JWT claims (`oid`, `tid`, `upn`/`preferred_username`, `appid`/`azp`, `scp`/`scope`, `iss`) and the token object itself.
cache_file	Path to the MSAL token cache JSON file. Defaults to `default_msal_token_cache()`.

Details

The function adds or overwrites `AccessToken`, `RefreshToken` (when the token carries a refresh token), `Account`, and `AppMetadata` sections. Existing entries for other accounts or clients are preserved.

The `home_account_id` follows the MSAL convention `"<object_id>.<tenant_id>"` where `object_id` is the Azure AD OID of the authenticated principal. Cache entry keys are built in the same format used by the Azure CLI and MSAL Python:

- AccessToken: `<home_account_id>-<environment>-accesstoken-<client_id>-<realm>-<target>`
- RefreshToken: `<home_account_id>-<environment>-refreshtoken-<client_id>--`
- Account: `<home_account_id>-<environment>-<realm>`
- AppMetadata: `appmetadata-<environment>-<client_id>`

Value

Invisibly returns the path to the cache file.

See Also

`az_cli_get_cached_token()`, `httr2::oauth_token()`

Index

api_client, 3, *8, 9*, *12–14*, *24*, *59*
api_graph_resource, *24*
api_log_analytics_client, 8, *25, 26*
api_resource, 11, *13*, *24*
api_service, 12, *24, 25*
api_storage_client, 13, *29*
as.list(), *22*
AuthCodeCredential, 16
az_cli_account_show, 35
az_cli_get_cached_token, 36
az_cli_get_cached_token(), *79, 80*
az_cli_get_token, 37
az_cli_get_token(), *69*
az_cli_is_login, 38
az_cli_login, 38
az_cli_login(), *32*, *40*
az_cli_logout, 40
az_config, 40
azr_catalog, 18, *19, 20*, *23*, *28*
azr_catalog_read, 19
azr_catalog_read(), *20*
azr_catalog_write, 20
azr_catalog_write(), *19*
azr_dataset, *18*, 20, *21–23*, *28*
azr_dataset_from_uri, 21
azr_dataset_manifest, 22, *28*
azr_dataset_uri, 23
azr_dataset_uri(), *27, 28*
azr_graph_client, 24
azr_logs_client, 25
azr_options, 27
azr_options(), *4*, *22, 23*, *28*, *31*, *48*, *64*
azr_resolve_dataset, 28
azr_storage_client, 29
azure_spark_storage_conf, 33
AzureCLICredential, 30

cached_token_credential_chain, 43
CachedTokenCredential, 41, *43*
ClientSecretCredential, 44, *71*
credential_chain, *9*, *13, 14*, *24*, *26*, *29*, 45
credential_chain(), *43*, *55*

data.frame, *27*
default_azure_cli_client_id, 50
default_azure_cli_client_id(), *14*, *69*
default_azure_client_id, 49
default_azure_client_id(), *34*, *41*, *51*
default_azure_client_secret, 49
default_azure_client_secret(), *34*
default_azure_config_dir, 50
default_azure_config_dir(), *37*, *41*, *57*
default_azure_host, 51
default_azure_host(), *34*, *53*
default_azure_oauth_client, 51
default_azure_scope, 52
default_azure_tenant_id, 53
default_azure_tenant_id(), *26*, *29*, *34*, *41*, *53*
default_azure_url, 53
default_credential_chain, 54
default_credential_chain(), *26*, *29*, *46*, *63, 64*, *69*
default_federated_token_file, 55
default_federated_token_file(), *34*
default_graph_endpoint, 56
default_graph_endpoint(), *24*
default_log_analytics_endpoint, 56
default_log_analytics_endpoint(), *9*, *26*
default_msal_token_cache, 57
default_msal_token_cache(), *79*
default_non_auth, 57
default_non_auth(), *4*
default_redirect_uri, 58
default_redirect_uri(), *17*
default_refresh_token, 58
default_refresh_token(), *34*, *76*
default_response_handler, 59
default_response_handler(), *4*
default_storage_endpoint, 59

default_storage_endpoint(), *14*, *29*
DefaultCredential, *9*, *14*, *24*, 46
DeviceCodeCredential, 60

get_credential_auth, 61
get_credential_provider, 63
get_credential_provider(), *46*
get_request_authorizer, 64
get_request_authorizer(), *62*, *64*, *66*, *68*
get_token, 66
get_token(), *62*, *65*, *68*, *69*, *71*
get_token_provider, 67
get_token_provider(), *46*, *55*, *62*, *64–66*
get_user_token, 68
get_user_token(), *66*

httpuv::randomPort(), *58*
httr2::oauth_client(), *51*, *52*
httr2::oauth_redirect_uri(), *58*
httr2::oauth_token(), *7*, *31*, *36–38*, *42*, *44*, *48*, *66*, *69*, *72*, *76*, *78–80*
httr2::req_headers(), *62*
httr2::req_perform(), *5*, *6*
httr2::request(), *5*, *6*, *31*, *42*, *44*, *48*, *57*, *73*, *76*, *78*
httr2::response(), *5–7*

InteractiveCredential, *16*, *60*
is_hosted_session, 70

lockEnvironment(), *24*, *25*

ManagedIdentityCredential, *27*, 71

parse_storage_path, 73
parse_storage_path(), *21*

RefreshTokenCredential, 75
rlang::englue(), *5*, *6*
rlang::is_interactive(), *64*

tempfile(), *15*

WorkloadIdentityCredential, *55*, 77
write_msal_token, 79
write_msal_token(), *57*