Package 'azr'

November 4, 2025

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

Version 0.1.0

Description Implements a credential chain for 'Azure OAuth 2.0' authentication based on the package 'httr2"s 'OAuth' framework. Sequentially attempts authentication methods until one succeeds. During development allows interactive browser-based flows ('Device Code' and 'Auth Code' flows) and non-interactive flow ('Client Secret') in batch mode.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

URL https://pedrobtz.github.io/azr/, https://github.com/pedrobtz/azr

BugReports https://github.com/pedrobtz/azr/issues

Imports methods, R6, cli, httr2, jsonlite, rlang

Suggests httpuv

Config/testthat/edition 3

NeedsCompilation no

Author Pedro Baltazar [aut, cre]

Maintainer Pedro Baltazar <pedrobtz@gmail.com>

Repository CRAN

Date/Publication 2025-11-04 18:20:02 UTC

Contents

`AuthCodeCredential` *Authorization code credential authentication*

Description

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

Details

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

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

Super classes

`azr::Credential` -> `azr::InteractiveCredential` -> `AuthCodeCredential`

Methods

Public methods:

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

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

Usage:

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

```
  redirect_uri = default_redirect_uri()
)
```

Arguments:

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

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

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

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

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

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

Returns: A new `AuthCodeCredential` object

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

Usage:

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

Arguments:

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

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

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

Usage:

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

Arguments:

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

Returns: The request object with OAuth authorization code authentication configured

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

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

```
)

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

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

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

## End(Not run)
```

`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

`azr::Credential` -> `AzureCLICredential`

Public fields

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

Methods

Public methods:

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

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

Usage:

```
AzureCLICredential$new(scope = NULL, tenant_id = NULL, process_timeout = 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`.

Returns: A new `AzureCLICredential` object

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

Usage:

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

Arguments:

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

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

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

Usage:

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

Arguments:

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

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

Returns: The request object with authentication header added

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

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

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

# To get a token or authenticate a request it is required that
# 'az login' is successfully executed, otherwise it will return an error.
## Not run:
# Get an access token
```

```
token <- cred$get_token()

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

## End(Not run)
```

`ClientSecretCredential` *Client secret credential authentication*

Description

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

Details

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

Super class

`azr::Credential` -> `ClientSecretCredential`

Methods

Public methods:

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

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

Usage:

```
ClientSecretCredential$validate()
```

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

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

Usage:

```
ClientSecretCredential$get_token()
```

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

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

Usage:

```
ClientSecretCredential$req_auth(req)
```

Arguments:

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

Returns: The request object with OAuth client credentials authentication configured

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

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

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

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

## End(Not run)
```

`default_azure_client_id` *Get default Azure client ID*

Description

Retrieves the Azure client ID from the `AZURE_CLIENT_ID` environment variable, or falls back to the default Azure CLI client ID if not set.

Usage

```
default_azure_client_id()
```

Value

A character string with the client ID

Examples

```
default_azure_client_id()
```

---

default_azure_client_secret

Get default Azure client secret

Description

Retrieves the Azure client secret from the `AZURE_CLIENT_SECRET` environment variable, or returns `NA_character_` if not set.

Usage

```
default_azure_client_secret()
```

Value

A character string with the client secret, or `NA_character_` if not set

Examples

```
default_azure_client_secret()
```

---

default_azure_host

Get default Azure authority host

Description

Retrieves the Azure authority host from the `AZURE_AUTHORITY_HOST` environment variable, or falls back to Azure Public Cloud if not set.

Usage

```
default_azure_host()
```

Value

A character string with the authority host URL

Examples

```
default_azure_host()
```

`default_azure_oauth_client` *Create default Azure OAuth client*

Description

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

Usage

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

Arguments

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

Value

An `httr2::oauth_client()` object

Examples

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

default_azure_scope *Get default Azure OAuth scope*

Description

Returns the default OAuth scope for a specified Azure resource.

Usage

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

Arguments

resource	A character string specifying the Azure resource. Must be one of: "azure_arm" (Azure Resource Manager), "azure_graph" (Microsoft Graph), "azure_storage" (Azure Storage), or "azure_key_vault" (Azure Key Vault). Defaults to "azure_arm".

Value

A character string with the OAuth scope URL

Examples

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

default_azure_tenant_id *Get default Azure tenant ID*

Description

Retrieves the Azure tenant ID from the AZURE_TENANT_ID environment variable, or falls back to the default value if not set.

Usage

```
default_azure_tenant_id()
```

Value

A character string with the tenant ID

Examples

```
default_azure_tenant_id()
```

default_azure_url *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_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()
```

DeviceCodeCredential *Device code credential authentication*

Description

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

Details

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

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

Super classes

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

Methods

Public methods:

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

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

Usage:

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

Arguments:

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

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

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

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

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

Returns: A new `DeviceCodeCredential` object

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

Usage:

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

Arguments:

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

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

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

Usage:

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

Arguments:

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

Returns: The request object with OAuth device code authentication configured

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

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

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

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

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

## End(Not run)
```

---

get_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 = FALSE,
  .chain = default_credential_chain(),
  .verbose = FALSE
)
```

Arguments

scope	Optional character string specifying the authentication scope.
tenant_id	Optional character string specifying the tenant ID for authentication.
client_id	Optional character string specifying the client ID for authentication.
client_secret	Optional character string specifying the client secret for authentication.

use_cache	Character string indicating the caching strategy. Defaults to "disk". Options include "disk" for disk-based caching or "memory" for in-memory caching.
offline	Logical. If TRUE, operates in offline mode. Defaults to FALSE.
.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 detailed diagnostic information during credential discovery and authentication. Defaults to FALSE.

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 = FALSE,
```

```
  .chain = default_credential_chain(),
  .verbose = FALSE
)
```

Arguments

scope	Optional character string specifying the authentication scope.
tenant_id	Optional character string specifying the tenant ID for authentication.
client_id	Optional character string specifying the client ID for authentication.
client_secret	Optional character string specifying the client secret for authentication.
use_cache	Character string indicating the caching strategy. Defaults to "disk". Options include "disk" for disk-based caching or "memory" for in-memory caching.
offline	Logical. If TRUE, operates in offline mode. Defaults to FALSE.
.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 detailed diagnostic information during credential discovery and authentication. Defaults to FALSE.

Value

An httr2::oauth_token() object.

See Also

get_token_provider(), get_request_authorizer()

Examples

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

## End(Not run)
```

`get_token_provider` *Get Default Token Provider Function*

Description

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

Usage

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

Arguments

`scope`	Optional character string specifying the authentication scope.
`tenant_id`	Optional character string specifying the tenant ID for authentication.
`client_id`	Optional character string specifying the client ID for authentication.
`client_secret`	Optional character string specifying the client secret for authentication.
`use_cache`	Character string indicating the caching strategy. Defaults to `"disk"`. Options include `"disk"` for disk-based caching or `"memory"` for in-memory caching.
`offline`	Logical. If `TRUE`, operates in offline mode. Defaults to `FALSE`.
`.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 detailed diagnostic information during credential discovery and authentication. Defaults to `FALSE`.

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)
```

Index