

# Azure Active Directory authentication for Go

This is a standalone package for authenticating with Azure Active Directory from other Go libraries and applications, in particular the [Azure SDK for Go](#).

Note: Despite the package's name it is not related to other "ADAL" libraries maintained in the [github.com/AzureAD](https://github.com/AzureAD) org. Issues should be opened in [this repo's](#) or [the SDK's](#) issue trackers.

## Install

```
go get -u github.com/Azure/go-autorest/autorest/adal
```

## Usage

An Active Directory application is required in order to use this library. An application can be registered in the [Azure Portal](#) by following these [guidelines](#) or using the [Azure CLI](#).

### Register an Azure AD Application with secret

1. Register a new application with a `secret` credential

```
az ad app create \  
  --display-name example-app \  
  --homepage https://example-app/home \  
  --identifier-uris https://example-app/app \  
  --password secret
```

2. Create a service principal using the `Application ID` from previous step

```
az ad sp create --id "Application ID"
```

- Replace `Application ID` with `appId` from step 1.

### Register an Azure AD Application with certificate

1. Create a private key

```
openssl genrsa -out "example-app.key" 2048
```

2. Create the certificate

```
openssl req -new -key "example-app.key" -subj "/CN=example-app" -out "example-app.csr"  
openssl x509 -req -in "example-app.csr" -signkey "example-app.key" -out  
"example-app.crt" -days 10000
```

3. Create the PKCS12 version of the certificate containing also the private key

```
openssl pkcs12 -export -out "example-app.pfx" -inkey "example-app.key" -in  
"example-app.crt" -passout pass:
```

4. Register a new application with the certificate content from `example-app.crt`

```
certificateContents="$(tail -n+2 "example-app.crt" | head -n-1)"

az ad app create \
  --display-name example-app \
  --homepage https://example-app/home \
  --identifier-uri https://example-app/app \
  --key-usage Verify --end-date 2018-01-01 \
  --key-value "${certificateContents}"
```

5. Create a service principal using the `Application ID` from previous step

```
az ad sp create --id "APPLICATION_ID"
```

- Replace `APPLICATION_ID` with `appId` from step 4.

## Grant the necessary permissions

Azure relies on a Role-Based Access Control (RBAC) model to manage the access to resources at a fine-grained level. There is a set of [pre-defined roles](#) which can be assigned to a service principal of an Azure AD application depending of your needs.

```
az role assignment create --assigner "SERVICE_PRINCIPAL_ID" --role "ROLE_NAME"
```

- Replace the `SERVICE_PRINCIPAL_ID` with the `appId` from previous step.
- Replace the `ROLE_NAME` with a role name of your choice.

It is also possible to define custom role definitions.

```
az role definition create --role-definition role-definition.json
```

- Check [custom roles](#) for more details regarding the content of `role-definition.json` file.

## Acquire Access Token

The common configuration used by all flows:

```
const activeDirectoryEndpoint = "https://login.microsoftonline.com/"
tenantID := "TENANT_ID"
oauthConfig, err := adal.NewOAuthConfig(activeDirectoryEndpoint, tenantID)

applicationID := "APPLICATION_ID"

callback := func(token adal.Token) error {
    // This is called after the token is acquired
}

// The resource for which the token is acquired
resource := "https://management.core.windows.net/"
```

- Replace the `TENANT_ID` with your tenant ID.

- Replace the `APPLICATION_ID` with the value from previous section.

### Client Credentials

```
applicationSecret := "APPLICATION_SECRET"

spt, err := adal.NewServicePrincipalToken(
    *oauthConfig,
    applicationID,
    applicationSecret,
    resource,
    callbacks...)
if err != nil {
    return nil, err
}

// Acquire a new access token
err = spt.Refresh()
if (err == nil) {
    token := spt.Token
}
```

- Replace the `APPLICATION_SECRET` with the `password` value from previous section.

### Client Certificate

```
certificatePath := "./example-app.pfx"

certData, err := ioutil.ReadFile(certificatePath)
if err != nil {
    return nil, fmt.Errorf("failed to read the certificate file (%s): %v",
certificatePath, err)
}

// Get the certificate and private key from pfx file
certificate, rsaPrivateKey, err := decodePkcs12(certData, "")
if err != nil {
    return nil, fmt.Errorf("failed to decode pkcs12 certificate while creating spt:
%v", err)
}

spt, err := adal.NewServicePrincipalTokenFromCertificate(
    *oauthConfig,
    applicationID,
    certificate,
    rsaPrivateKey,
    resource,
    callbacks...)

// Acquire a new access token
err = spt.Refresh()
```

```
if (err == nil) {
    token := spt.Token
}
```

- Update the certificate path to point to the example-app.pfx file which was created in previous section.

## Device Code

```
oauthClient := &http.Client{}

// Acquire the device code
deviceCode, err := adal.InitiateDeviceAuth(
    oauthClient,
    *oauthConfig,
    applicationID,
    resource)
if err != nil {
    return nil, fmt.Errorf("Failed to start device auth flow: %s", err)
}

// Display the authentication message
fmt.Println(*deviceCode.Message)

// Wait here until the user is authenticated
token, err := adal.WaitForUserCompletion(oauthClient, deviceCode)
if err != nil {
    return nil, fmt.Errorf("Failed to finish device auth flow: %s", err)
}

spt, err := adal.NewServicePrincipalTokenFromManualToken(
    *oauthConfig,
    applicationID,
    resource,
    *token,
    callbacks...)

if (err == nil) {
    token := spt.Token
}
```

## Username password authenticate

```
spt, err := adal.NewServicePrincipalTokenFromUsernamePassword(
    *oauthConfig,
    applicationID,
    username,
    password,
    resource,
    callbacks...)

if (err == nil) {
```

```
    token := spt.Token
}
```

### Authorization code authenticate

```
spt, err := adal.NewServicePrincipalTokenFromAuthorizationCode(
    *oauthConfig,
    applicationID,
    clientSecret,
    authorizationCode,
    redirectURI,
    resource,
    callbacks...)

err = spt.Refresh()
if (err == nil) {
    token := spt.Token
}
```

### Command Line Tool

A command line tool is available in `cmd/adal.go` that can acquire a token for a given resource. It supports all flows mentioned above.

```
adal -h

Usage of ./adal:
  -applicationId string
      application id
  -certificatePath string
      path to pk12/PFC application certificate
  -mode string
      authentication mode (device, secret, cert, refresh) (default "device")
  -resource string
      resource for which the token is requested
  -secret string
      application secret
  -tenantId string
      tenant id
  -tokenCachePath string
      location of oath token cache (default "/home/cgc/.adal/accessToken.json")
```

Example acquire a token for `https://management.core.windows.net/` using device code flow:

```
adal -mode device \
  -applicationId "APPLICATION_ID" \
  -tenantId "TENANT_ID" \
  -resource https://management.core.windows.net/
```