Use client assertion to get access tokens from Azure AD

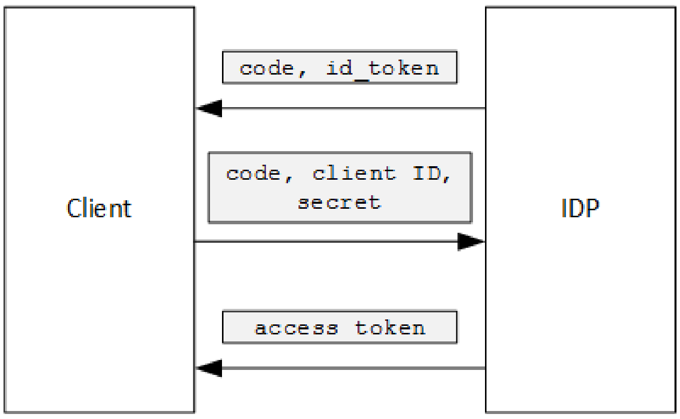
07/21/2017 • 2 minutes to read • Contributors

In this article

Sample code

Background

When using authorization code flow or hybrid flow in OpenID Connect, the client exchanges an authorization code for an access token. During this step, the client has to authenticate itself to the server.



Example: Hybrid flow

One way to authenticate the client is by using a client secret. That's how the Tailspin Surveys application is configured by default.

Here is an example request from the client to the IDP, requesting an access token. Note the `client_secret` parameter.

HTTP

```http
POST https://login.microsoftonline.com/b9bd2162xxx/oauth2/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded

resource=https://tailspin.onmicrosoft.com/surveys.webapi
  &client_id=87df91dc-63de-4765-8701-b59cc8bd9e11
  &client_secret=i3Bf12Dn...
  &grant_type=authorization_code
  &code=PG8wJG6Y...
```

The secret is just a string, so you have to make sure not to leak the value. The best practice is to keep the client secret out of source control. When you deploy to Azure, store the secret in an app setting.

However, anyone with access to the Azure subscription can view the app settings. Furthermore, there is always a temptation to check secrets into source control (for example, in deployment scripts), share them by email, and so on.

For additional security, you can use client assertion instead of a client secret. With client assertion, the client uses an X.509 certificate to prove the token request came from the client. The client certificate is installed on the web server. Generally, it will be easier to restrict access to the certificate, than to ensure that nobody inadvertently reveals a client secret. For more information about configuring certificates in a web app, see Using Certificates in Azure Websites Applications

Here is a token request using client assertion:

HTTP

Copy

```
POST https://login.microsoftonline.com/b9bd2162xxx/oauth2/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded

resource=https://tailspin.onmicrosoft.com/surveys.webapi
  &client_id=87df91dc-63de-4765-8701-b59cc8bd9e11
  &client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer
  &client_assertion=eyJhbGci...
  &grant_type=authorization_code
  &code= PG8wJG6Y...
```

Notice that the `client_secret` parameter is no longer used. Instead, the `client_assertion` parameter contains a JWT token that was signed using the client certificate. The `client_assertion_type` parameter specifies the type of assertion — in this case, JWT token. The server validates the JWT token. If the JWT token is invalid, the token request returns an error.

Note

X.509 certificates are not the only form of client assertion; we focus on it here because it is supported by Azure AD.

At run time, the web application reads the certificate from the certificate store. The certificate must be installed on the same machine as the web app.

The Surveys application includes a helper class that creates a ClientAssertionCertificate that you can pass to the AuthenticationContext.AcquireTokenSilentAsync method to acquire a token from Azure AD.

C#

Copy

```
public class CertificateCredentialService : ICredentialService
{
    private Lazy<Task<AdalCredential>> _credential;

    public CertificateCredentialService(IOptions<ConfigurationOptions> options)
    {
        var aadOptions = options.Value?.AzureAd;
        _credential = new Lazy<Task<AdalCredential>>(() =>
        {
            X509Certificate2 cert = CertificateUtility.FindCertificateByThumbprint(
                aadOptions.Asymmetric.StoreName,
                aadOptions.Asymmetric.StoreLocation,
                aadOptions.Asymmetric.CertificateThumbprint,
                aadOptions.Asymmetric.ValidationRequired);
            string password = null;
            var certBytes = CertificateUtility.ExportCertificateWithPrivateKey(cert, out pass-
word);
            return Task.FromResult(new AdalCredential(new ClientAssertionCertificate(aadOption-
s.ClientId, new X509Certificate2(certBytes, password))));
        });
    }

    public async Task<AdalCredential> GetCredentialsAsync()
    {
        return await _credential.Value;
    }
}
```

For information about setting up client assertion in the Surveys application, see Use Azure Key Vault to protect application secrets .

Next