AAD - Access using local Certificate

Last updated by | Vitor Tomaz | Feb 18, 2021 at 2:30 AM PST

Contents

AAD - Access using local Certificate

(C# code sample for .Net 4.6 or higher)

By MirekS

Due to the new Azure RM PowerShell (see here) and the new Azure AD portal, the Azure AD certificate based access to SQL DB changed and required some update. This is an updated version of the previous document describing this solution and available here .

(For more information on Azure AD authentication see here)

The solution contains a simple console application that connects to Azure SQL database using a self-signed certificate registered with Azure AD and stored on a local machine. Using such certificate removes the necessity to use other credentials such as user/password, or federated (Windows) authentication.

To build and run this project perform the following steps:

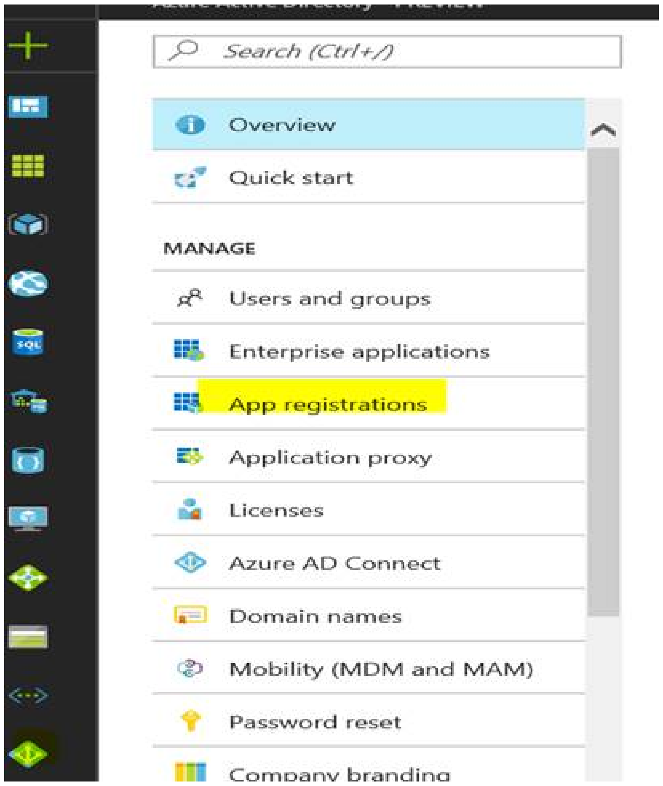
a. **Register the Azure AD Web APP**
b. **Create Azure DB user for the Web APP**
c. **Create a self-signed certificate**
d. **Add a certificate to the Web App**
e. **Configure Console Application**
f. **Run Console Application to verify the connection**

@ 1) Register the Web API

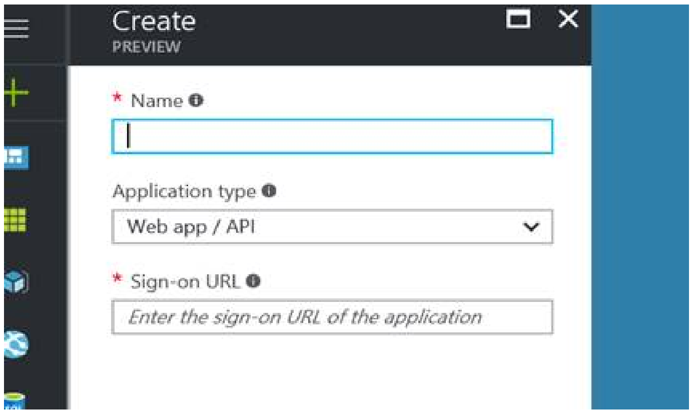
a. 1.Sign in to the [Azure portal](https://portal.azure.com).

b. On the top bar, right side, click on your account and under the **Directory** list, choose the Active Directory tenant where you wish to register your application.
c. Click on **More Services** in the left hand nav, and choose **Azure Active Directory**.
d. Click on **App registrations** and choose **Add** (see Pic 1)
e. Enter a friendly name for the application, for example "**specialtest**" and select 'Web Application and/or Web API' as the Application Type (see Pic 2)
f. Since this application is a daemon and not a web application, it doesn't have a sign-in URL or app ID URI. For these two fields, enter "**http://specialtest**". Click on **Create** to create the application.
g. While still in the Azure portal, choose your application, click on **Settings** and choose **Properties**.
h. Find the Object ID value and copy it to the clipboard (see Pic#3)

Pic#1

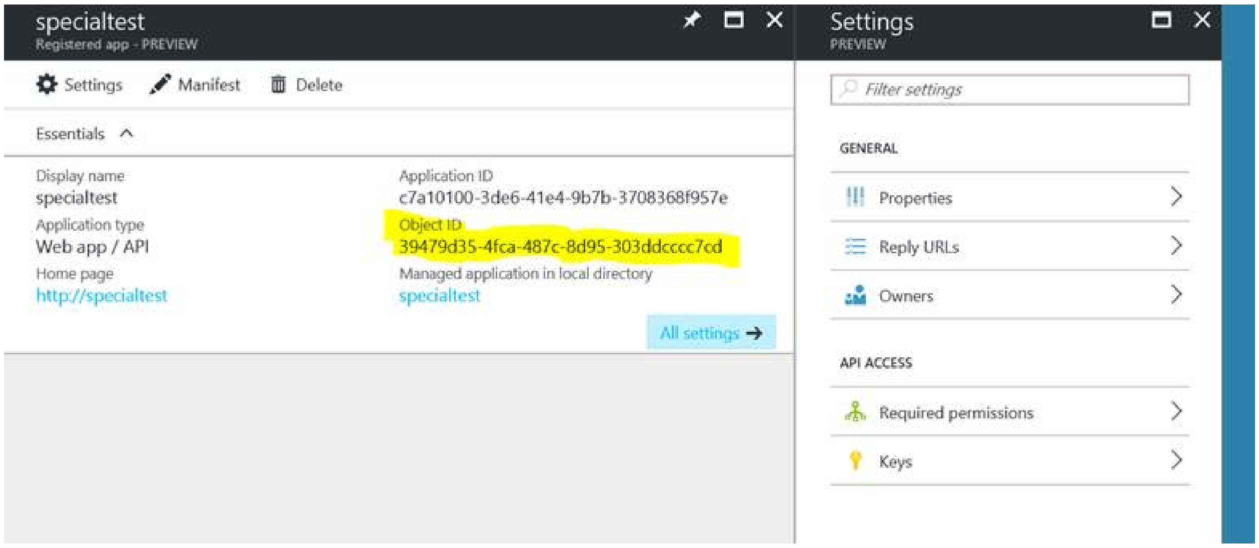


Pic#2



Create a self-signed certificate

Pic#3


specialtest
Registered app - PREVIEW

Display name: specialtest
Application type: Web app / API
Home page: http://specialtest
Application ID: c7a10100-3de6-41e4-9b7b-3708368f957e
Object ID: 39479d35-4fca-487c-8d95-303ddcccc7cd
Managed application in local directory: specialtest

@2) Create Azure DB user for the Web APP

Logon to your Azure SQL Server's user database as an Azure AD admin and using a T-SQL command provision a contained database user for your application principal:

CREATE USER [specialtest] FROM EXTERNAL PROVIDER

See here for more details on how to create an Azure Ad admin and a contained database user.

Note:- Create user will fail if you do not use the exact same name as the application name from step1

@3) Create a self-signed certificate

To complete this step you will need the makecert.exe utilities, which is included in the Windows SDK. You may have installed the Windows SDK with Visual Studio, in which case you will find it if you search your computer for makecert.exe. If not, you can download the SDK [here](http://msdn.microsoft.com/en-US/windows/desktop/aa904949). You only need to install the Windows Software Development Kit part of the SDK. You can find more information about the makecert.exe utility [here](http://msdn.microsoft.com/en-us/library/bfsktky3(v=vs.110).aspx).

On the client machine (on which, you want to run the project), generate and install a self-signed certificate. In the command-line window, navigate to a folder, where you want to generate a certificate file, and run makecert.exe with the following parameters to create a self-signed certificate on your computer:

```
makecert -r -pe -n "CN=<CertName>" -ss My -len 2048 <CertName>.cer
```

i.e. created directory c:\AAD and execute

c:/"Program Files (x86)/Windows Kits/8.1/bin/x64"/makecert -r -pe -n "CN=mytokentestCert"

-ss My -len 2048 mytokentestCert.cer

Check if mytokentestCert.cer was created unde c:\AAD

@4) Add a certificate to the Web App

Use PowerShell to register a local machine certificate with Azure AD

#Pre -Requisite is to install AzureAdPreview cmdlets and get latest version

#Check the version by executing

(Get-Module AzureADPreview -ListAvailable).Version

Output

```
Major  Minor  Build  Revision

-----  -----  -----  --------

2      0      0      52
```

Initial PowerShell setup

1. Add-AzureAccount | format-list #add your Azure Account
2. Select-AzureSubscription -SubscriptionId <SubscriptionID> #**select your subscription**
3. Get-AzureSubscription -Current | Format-List
   4) Get-AzureSqlDatabaseServer **# make sure your server is listed under this subscription**

Add the certificate as a key for the application in Azure AD in the example above for “**specialtest” app**

Open powershell and navigate to the directory where you created your new certificate. Use the following PowerShell commands to generate a `$base64Value`, a `$base64Thumbprint`, and a `$keyid` that you can use to upload your cert to Azure AD:

$cer = New-Object System.Security.Cryptography.X509Certificates.X509Certificate2

$cer.Import("<**Full path\certname.cert**") #

$bin = $cer.GetRawCertData()

$base64Value = [System.Convert]::ToBase64String($bin)

$bin = $cer.GetCertHash()

$base64Thumbprint = [System.Convert]::ToBase64String($bin)

$keyid = [System.Guid]::NewGuid().ToString()

Connect-AzureAD

#>>You must have Azure AD global admin credential to connect

#>>and to proceed successfully with the command below

$cred=New-AzureADApplicationKeyCredential -ObjectId **<ObjectId>**

    -CustomKeyIdentifier $base64Thumbprint -Type AsymmetricX509Cert -Usage Verify

    -Value $base64Value

i.e.

Connect-AzureAD

#i.e. connect as bob@myadd.onmicrosoft.com with Azure AD password where bob is the Azure Ad admin

$cer = New-Object System.Security.Cryptography.X509Certificates.X509Certificate2 #create a new certificate object

$cer.Import("**c:\aad\mytokentestCert.cer**") # use the local directory that contains the cert created in @3.

$bin = $cer.GetRawCertData()

$base64Value = [System.Convert]::ToBase64String($bin)

$bin = $cer.GetCertHash()

$base64Thumbprint = [System.Convert]::ToBase64String($bin)

$keyid = [System.Guid]::NewGuid().ToString()

# based on Fig. 3 use the right ObjectID below

New-AzureADApplicationKeyCredential -ObjectId **39479d35-4fca-487c-8d95-303ddcccc7cd**

-CustomKeyIdentifier $base64Thumbprint -Type AsymmetricX509Cert -Usage Verify -Value $base64Value

Output

```
CustomKeyIdentifier : {80, 81, 117, 105...}

EndDate             : 4/19/2018 4:35:35 PM

KeyId               : 92c06162-d1b3-432e-9352-2973be552e66

StartDate           : 4/19/2017 4:35:35 PM

Type                : AsymmetricX509Cert

Usage               : Verify

Value               : {77, 73, 73, 68...}
```

@5. Configure Console Application

a. Download and unzip the attached project
b. Make sure the .NET version is 4.6 or higher
c. **Open app.config.**
  - Find the app key ida:Tenant and replace the value with your **AAD tenant name**.
  - Find the app key ida:ClientId and replace the value with the **Application ID** for the app registration from the Azure portal (see **specialtest** AppID Pic.3)
  - Find the app key ida:Cert_Name and replace the value with the subject name of the self-signed certificate you created (see **CN=mytokentestCert at @3(**

i.e.

```
    <add key="ida:Tenant" value="apptest.onmicrosoft.com" /> //this is the AAD domaain

    <add key="ida:ClientId" value="C7A10100-3DE6-41E4-9B7B-3708368F957E"/> //Application ID

 <add key="ida:CertName" value="CN=mytokentestCert"/> //Cert_name used by makecert.exe
```

d. **Open Program.cs**,

Locate the following lines of code and replace the server/database name with your server/database name.

i.e.

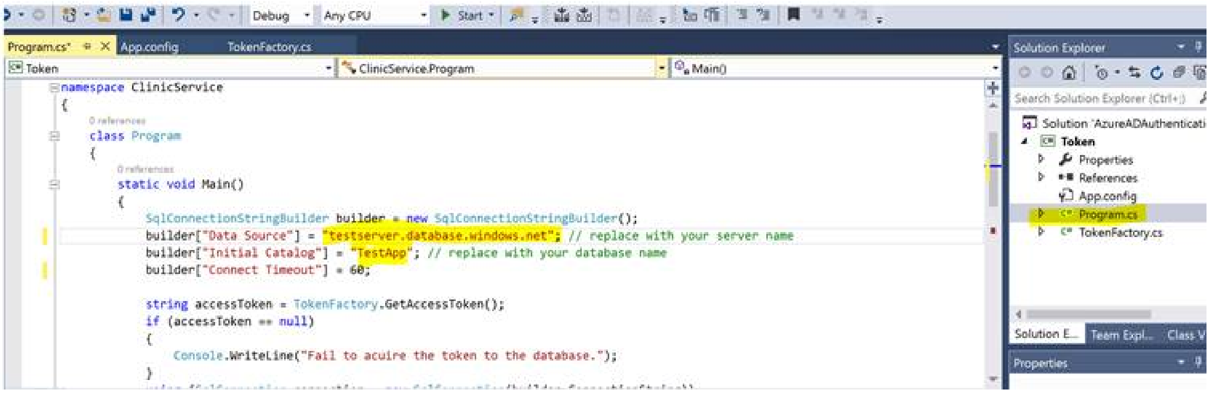
builder["Data Source"] = "**testserver.database.windows.net**"; // replace with your server name

builder["Initial Catalog"] = "**TestApp**"; // replace with your database name

builder["Connect Timeout"] = **60;** // connection time out – make it >30 sec

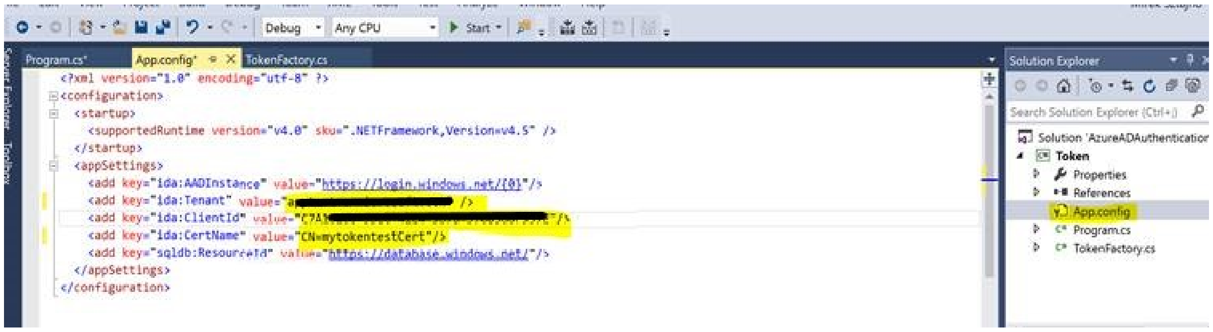
See also Pick.4 and Pick.5 below

Pic.4



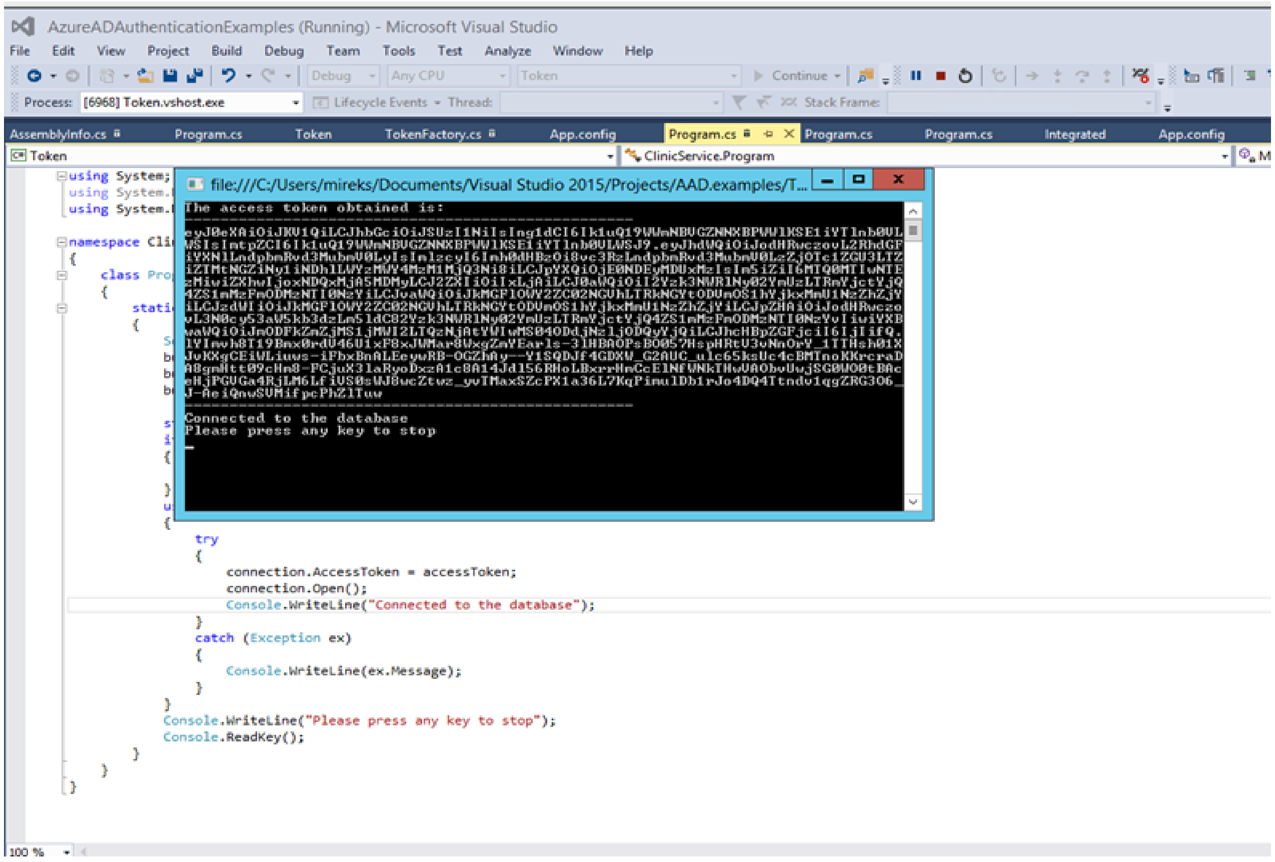
```
namespace ClinicService
{
    0 references
    class Program
    {
        0 references
        static void Main()
        {
            SqlConnectionStringBuilder builder = new SqlConnectionStringBuilder();
            builder["Data Source"] = "testserver.database.windows.net"; // replace with your server name
            builder["Initial Catalog"] = "TestApp"; // replace with your database name
            builder["Connect Timeout"] = 60;

            string accessToken = TokenFactory.GetAccessToken();
            if (accessToken == null)
            {
                Console.WriteLine("Fail to acuire the token to the database.");
            }
```

Pic.5



```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
  </startup>
  <appSettings>
    <add key="ida:AADInstance" value="https://login.windows.net/{0}"/>
    <add key="ida:Tenant" value="a[redacted] />
    <add key="ida:ClientId" value="C7A[redacted]"/>
    <add key="ida:CertName" value="CN=mytokentestCert"/>
    <add key="sqldb:ResourceId" value="https://database.windows.net/"/>
  </appSettings>
</configuration>
```

6. Execute Program.cs


AzureADAuthenticationExamples (Running) - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Debug Any CPU Token Continue
Process: [6968] Token.vshost.exe Lifecycle Events Thread: Stack Frame:
AssemblyInfo.cs Program.cs Token TokenFactory.cs App.config Program.cs Program.cs Program.cs Integrated App.config
Token
ClinicService.Program
file:///C:/Users/mireks/Documents/Visual Studio 2015/Projects/AAD.examples/T...
100 %

```
using System;
using System.
using System.I

namespace Cli
{
    class Pro
    {
        stati
        {
            S
            b
            b
            b

            s
            i
            {
            }
            u
            {
                try
                {
                    connection.AccessToken = accessToken;
                    connection.Open();
                    Console.WriteLine("Connected to the database");
                }
                catch (Exception ex)
                {
                    Console.WriteLine(ex.Message);
                }
            }
            Console.WriteLine("Please press any key to stop");
            Console.ReadKey();
        }
    }
}
```

```
The access token obtained is:
--------------------------------------------------------------------------------
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6Ik1uQ19WWmNBUGZNNXBPWW1KSE1iYTlnb0VL
WSIsImtpZCI6Ik1uQ19WWmNBUGZNNXBPWW1KSE1iYTlnb0VLWSJ9.eyJhdWQiOiJodHRwczovL2RhdGF
iYXNlLndpbmRvd3MubmV0LyIsImlzcyI6Imh0dHBzOi8vc3RzLndpbmRvd3MubmV0LzZjOTc1ZGU3LTZ
iZTMtNGZiNy1iNDh1LWYzMWY4MzM1MjQ3NiI8iLCJpYXQiOjE0NDEyMDUxMzIsIm5iZiI6MTQ0MTIwNTE
zMiwiZXhwIjoxNDQxMjA5MDMyLCJ2ZXIiOiIxLjAiLCJ0aWQiOiI2Yzk3NWR1Ny02YmUzLTRmYjctYjQ
4ZS1mMzFmODMzNTI0NzYiLCJvaWQiOiJkMGF1OWY2ZC02NGVhLTRkNGYtODUmOS1hYjkxMmU1NzZhZjY
iLCJzdWIiOiJkMGF1OWY2ZC02NGVhLTRkNGYtODUmOS1hYjkxMmU1NzZhZjYiLCJpZHAiOiJodHRwczo
vL3N0cy53aW5kb3dzLm51dC82Yzk3NWR1Ny02YmUzLTRmYjctYjQ4ZS1mMzFmODMzNTI0NzYvIiwiYXB
waWQiOiJmODFkZmZjMS1jMWI2LTQzNjAtYWIwMS04ODdjNzljODQyYjQiLCJhcHBpZGFjciI6IjIifQ.
lYImvh8T19Bmx0rdU46UixF8xJWMar8WxgZmYEarls-3lHBAOPsBO057HspHRtU3vNnOrY_1TTHsh01X
JvKXgCEiWLiuws-iFbxBnALEeywRB-OGZhAy--Y1SQDJf4GDXW_G2AUC_ulc65ksUc4cBMTnoKKrcraD
A8gmHtt09cHm8-PCjuX3laRyoDxzA1c8A14Jd156RHoLBxrrHmCcE1NfWNkTHwUAObvUwjSG0WO0tBAc
eHjPGUGa4RjLM6LfiUS0sWJ8ucZtwz_yvTMaxSZcPX1a36L7KqPimu1Db1rJo4DQ4Ttndv1qgZRG3O6_
J-AeiQnwSUMifpcPhZlTuw
--------------------------------------------------------------------------------
Connected to the database
Please press any key to stop
_
```

During the execution, an Azure AD token will be displayed in the execution window indicating successful connection

Program.cs (example)

```
using System;
using System.Data;
using System.Data.SqlClient;
namespace ClinicService
{
    class Program
    {
        static void Main()
     {
        SqlConnectionStringBuilder builder = new SqlConnectionStringBuilder();
        builder["Data Source"] = "testserver.database.windows.net";//repl. your server
        builder["Initial Catalog"] = "TestApp"; // replace: your database
        builder["Connect Timeout"] = 120;

        string accessToken = TokenFactory.GetAccessToken();
          if (accessToken == null)
            {
              Console.WriteLine("Fail to acuire the token to the database.");
            }
            using (SqlConnection connection = new
                   SqlConnection(builder.ConnectionString))
            {   try
                {
                    connection.AccessToken = accessToken;
                    connection.Open();
                    Console.WriteLine("Connected to the database");
                }
                catch (Exception ex)
                {
                    Console.WriteLine(ex.Message);
                }
            }
            Console.WriteLine("Please press any key to stop");
            Console.ReadKey();
        }
    }
}
```

App.config (example)

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
  </startup>
  <appSettings>
    <add key="ida:AADInstance" value="https://login.windows.net/{0}"/>
    <add key="ida:Tenant" value="apptest.onmicrosoft.com" />
    <add key="ida:ClientId" value="C7A10100-3DE6-41E4-9B7B-3708368F957E"/>
    <add key="ida:CertName" value="mytokentestCert"/>
    <add key="sqldb:ResourceId" value="https://database.windows.net/"/>
  </appSettings>
</configuration>
```

How good have you found this content?