Azure SSO Login Flow with .NET Core OIDC and Service Fabric Deployment

# 1. Overview

This document outlines the step-by-step process of implementing Azure Single Sign-On (SSO) in a .NET Core application using OpenID Connect (OIDC). The application is deployed using Azure Service Fabric and is accessed via a machine-specific HTTPS URL secured by a certificate.

# 2. Technology Stack

- .NET Core (ASP.NET Core MVC)  
- OpenID Connect (OIDC) Middleware  
- Azure Active Directory (Azure AD)  
- Service Fabric (on-prem or Azure-hosted)  
- HTTPS with certificate-based binding

# 3. Prerequisites

- An Azure AD tenant with registered app  
- SSL certificate installed on the Service Fabric node(s)  
- Machine-specific URL accessible (e.g., https://MACHINE123.domain.com)  
- OIDC middleware configured in the .NET Core app  
- Service Fabric configured for HTTPS endpoint

# 4. Step-by-Step Azure SSO Login Flow

## Step 1: App Registration in Azure AD

Register your application in Azure AD under App registrations.  
- Note down the Client ID and Tenant ID.  
- Set the Redirect URI to https://MACHINE123.domain.com/signin-oidc.

## Step 2: Configure Certificates on Service Fabric

- Upload and install the SSL certificate on all nodes.  
- Configure Service Fabric to bind HTTPS endpoints to the certificate.

## Step 3: Configure OIDC in .NET Core App

In Startup.cs:

services.AddAuthentication(options =>  
{  
 options.DefaultScheme = CookieAuthenticationDefaults.AuthenticationScheme;  
 options.DefaultChallengeScheme = OpenIdConnectDefaults.AuthenticationScheme;  
})  
.AddCookie()  
.AddOpenIdConnect(options =>  
{  
 options.Authority = "https://login.microsoftonline.com/{TenantID}";  
 options.ClientId = "{ClientID}";  
 options.ClientSecret = "{ClientSecret}"; // If using authorization code flow  
 options.ResponseType = "code";  
 options.SaveTokens = true;  
 options.CallbackPath = "/signin-oidc";  
 options.Scope.Add("openid");  
 options.Scope.Add("profile");  
});

## Step 4: HTTPS Configuration in appsettings.json (Optional)

"Kestrel": {  
 "Endpoints": {  
 "Https": {  
 "Url": "https://MACHINE123.domain.com:443",  
 "Certificate": {  
 "Path": "<path-to-pfx-file>",  
 "Password": "<cert-password>"  
 }  
 }  
 }  
}

## Step 5: Deploy to Service Fabric

- Package and deploy your application using sfctl or Visual Studio.  
- Ensure HTTPS endpoint configuration is specified in ServiceManifest.xml.

## Step 6: Access Application and Redirect to Azure Login

- Navigate to https://MACHINE123.domain.com.  
- User is redirected to Microsoft login screen.  
- Upon successful authentication, Azure AD redirects to /signin-oidc with authorization code.

## Step 7: OIDC Middleware Validates Token

- Middleware exchanges the authorization code for tokens.  
- Claims are extracted and user is signed in via cookie authentication.

## Step 8: User Accesses Application

- Authenticated session is maintained using cookies.  
- User gains access to protected resources.