# Okta Login Process using PKCE with Session and Token Management

## 1. Okta Authentication Flow with PKCE

The application uses Authorization Code Flow with PKCE for secure authentication and token exchange. The PKCE mechanism protects the authorization code from interception.

Steps:

1. 1. Generate a `code\_verifier` and a hashed `code\_challenge` (SHA256).
2. 2. Redirect the user to Okta with the `code\_challenge` and client ID.
3. 3. Okta prompts user login and returns an authorization code to the redirect URI.
4. 4. Application exchanges the code + `code\_verifier` for ID token, access token, and refresh token.

## 2. ASP.NET Core Configuration (Startup.cs)

Register authentication with PKCE, session timeout, and sliding expiration:

services.AddAuthentication(options =>  
{  
 options.DefaultScheme = CookieAuthenticationDefaults.AuthenticationScheme;  
 options.DefaultChallengeScheme = OpenIdConnectDefaults.AuthenticationScheme;  
})  
.AddCookie(options =>  
{  
 options.ExpireTimeSpan = TimeSpan.FromMinutes(15); // Session Timeout  
 options.SlidingExpiration = true;  
})  
.AddOpenIdConnect(options =>  
{  
 options.Authority = "https://your-okta-domain.okta.com/oauth2/default";  
 options.ClientId = "your-client-id";  
 options.ClientSecret = "your-client-secret"; // Optional for confidential clients  
 options.ResponseType = "code";  
 options.UsePkce = true;  
 options.Scope.Add("openid");  
 options.Scope.Add("profile");  
 options.Scope.Add("offline\_access"); // Required for refresh token  
 options.SaveTokens = true;  
});

## 3. Frontend Token Refresh and Idle Sign-Out (JavaScript)

Tracks user activity to refresh tokens and trigger logout on idle:

let lastActivity = Date.now();  
let refreshInterval = 13 \* 60 \* 1000; // 13 minutes  
let idleLimit = 16 \* 60 \* 1000; // 16 minutes  
  
document.onclick = document.onmousemove = document.onkeypress = () => {  
 lastActivity = Date.now();  
};  
  
setInterval(() => {  
 if (Date.now() - lastActivity < refreshInterval) {  
 fetch('/account/refresh-token'); // Server-side endpoint to refresh token  
 }  
}, refreshInterval);  
  
setInterval(() => {  
 if (Date.now() - lastActivity > idleLimit) {  
 window.location.href = '/account/logout';  
 }  
}, 10000);

## 4. Token Refresh Endpoint (Server-side)

Use the refresh token to obtain a new access token silently:

[HttpPost]  
public async Task<IActionResult> RefreshToken()  
{  
 var refreshToken = await HttpContext.GetTokenAsync("refresh\_token");  
  
 var client = \_httpClientFactory.CreateClient();  
 var request = new HttpRequestMessage(HttpMethod.Post, "https://your-okta-domain.okta.com/oauth2/default/v1/token");  
 request.Content = new FormUrlEncodedContent(new Dictionary<string, string>  
 {  
 { "grant\_type", "refresh\_token" },  
 { "client\_id", "your-client-id" },  
 { "client\_secret", "your-client-secret" },  
 { "refresh\_token", refreshToken }  
 });  
  
 var response = await client.SendAsync(request);  
 // Handle response and store new tokens  
}

## 5. Summary

• Session Timeout: 15 minutes with sliding expiration.

• Token Refresh: Every 13 minutes if interaction is detected.

• Idle Sign-Out: After 16 minutes of inactivity, user is logged out.

• PKCE: Used to securely exchange authorization codes without exposing client secrets.