InvalidOperationException: No authentication handler is registered for the scheme Bearer.

3

I am trying to implement Aspnet.security.openidconnect (ASOS) with .net core 2.1 I can successfully generate access_token and refreshtoken using ASOS but when I am adding Authorize Attribute on any of my action and try to call that action with postman I am getting following exception:

```
InvalidOperationException: No authentication handler is registered for the scheme
Bearer. The registered schemes are: ASOS. Did you forget to call
AddAuthentication().Add[SomeAuthHandler
```

Here is the code:

```
 services.AddAuthentication(options =>
        {
            options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
            options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
    }).AddOpenIdConnectServer(options =>
    {
        options.AuthorizationEndpointPath = "/connect/authorize";
        // Enable the token endpoint.
        options.TokenEndpointPath = "/connect/token";

        // Implement OnValidateTokenRequest to support flows using the token endpoint.
        options.Provider.OnValidateTokenRequest = context =>
        {
            // Reject token requests that don't use grant_type=password or
grant_type=refresh_token.
            if (!context.Request.IsClientCredentialsGrantType() &&
!context.Request.IsRefreshTokenGrantType())
            {
                context.Reject(
                    error: OpenIdConnectConstants.Errors.UnsupportedGrantType,
                    description: "Only grant_type=password and refresh_token " +
```

```
            // parameter is missing to support unauthenticated token requests.
            // if (string.IsNullOrEmpty(context.ClientId))
            // {
            //     context.Skip();
            //
            //     return Task.CompletedTask;
            // }

            // Note: to mitigate brute force attacks, you SHOULD strongly consider
applying
            // a key derivation function like PBKDF2 to slow down the secret validation
process.
            // You SHOULD also consider using a time-constant comparer to prevent timing
attacks.
            if (string.Equals(context.ClientId, "client_id", StringComparison.Ordinal)
&&
                string.Equals(context.ClientSecret, "client_secret",
StringComparison.Ordinal))
            {
                context.Validate();
            }

            // Note: if Validate() is not explicitly called,
            // the request is automatically rejected.
            return Task.CompletedTask;
        };

        // Implement OnHandleTokenRequest to support token requests.
        options.Provider.OnHandleTokenRequest = context =>
        {
            // Only handle grant_type=password token requests and let
            // the OpenID Connect server handle the other grant types.
            if (context.Request.IsClientCredentialsGrantType())
            {
                // Implement context.Request.Username/context.Request.Password
validation here.
                // Note: you can call context Reject() to indicate that authentication
failed.
                // Using password derivation and time-constant comparer is STRONGLY
recommended.
                //if (!string.Equals(context.Request.Username, "Bob",
StringComparison.Ordinal) ||
                //    !string.Equals(context.Request.Password, "P@ssw0rd",
```

```
                //    return Task.CompletedTask;
                //}

                var identity = new ClaimsIdentity(context.Scheme.Name,
                    OpenIdConnectConstants.Claims.Name,
                    OpenIdConnectConstants.Claims.Role);

                // Add the mandatory subject/user identifier claim.
                identity.AddClaim(OpenIdConnectConstants.Claims.Subject, "[unique id]");

                // By default, claims are not serialized in the access/identity tokens.
                // Use the overload taking a "destinations" parameter to make sure
                // your claims are correctly inserted in the appropriate tokens.
                identity.AddClaim("urn:customclaim", "value",
                    OpenIdConnectConstants.Destinations.AccessToken,
                    OpenIdConnectConstants.Destinations.IdentityToken);

                var ticket = new AuthenticationTicket(
                    new ClaimsPrincipal(identity),
                    new AuthenticationProperties(),
                    context.Scheme.Name);

                // Call SetScopes with the list of scopes you want to grant
                // (specify offline_access to issue a refresh token).
                ticket.SetScopes(
                    OpenIdConnectConstants.Scopes.Profile,
                    OpenIdConnectConstants.Scopes.OfflineAccess);

                context.Validate(ticket);
            }

            return Task.CompletedTask;
        };
    });
```

and in configure method I am calling:

```
app.UseAuthentication();
```

What is missing here? Thanks

Ask
366 3 17

You're specifying the default scheme as Bearer (JwtBearerDefaults.AuthenticationScheme) where did you copy this example from. – davidfowl Jul 17 '18 at 15:35

That was some github link but in that link he just wrote services.AddAuthentication ().AddOpenIdConnectServer (...) But even with that i was getting an exception that no default authentication scheme is defined – Ask Jul 17 '18 at 16:10

1 Answer

3

The snippet you shared only generates tokens: it doesn't validate them. To enable token validation, reference the `AspNet.Security.OAuth.Validation` package and register the aspnet-contrib validation handler:

```
services.AddAuthentication(OAuthValidationDefaults.AuthenticationScheme)
    .AddOAuthValidation();
```

answered Jul 17 '18 at 16:56

Pinpoint
25.1k 4 74 102

Thanks for the answer. It worked – Ask Jul 19 '18 at 5:48