

Authorize with a specific scheme in ASP.NET Core

11/08/2019 • 3 minutes to read •  +3

In this article

[Selecting the scheme with the Authorize attribute](#)

[Selecting the scheme with policies](#)

[Use multiple authentication schemes](#)

In some scenarios, such as Single Page Applications (SPAs), it's common to use multiple authentication methods. For example, the app may use cookie-based authentication to log in and JWT bearer authentication for JavaScript requests. In some cases, the app may have multiple instances of an authentication handler. For example, two cookie handlers where one contains a basic identity and one is created when a multi-factor authentication (MFA) has been triggered. MFA may be triggered because the user requested an operation that requires extra security. For more information on enforcing MFA when a user requests a resource that requires MFA, see the GitHub issue [Protect section with MFA](#).

An authentication scheme is named when the authentication service is configured during authentication. For example:

C#	 Copy
<pre>public void ConfigureServices(IServiceCollection services) { // Code omitted for brevity services.AddAuthentication() .AddCookie(options => { options.LoginPath = "/Account/Unauthorized/"; options.AccessDeniedPath = "/Account/Forbidden/"; }) .AddJwtBearer(options => { options.Audience = "http://localhost:5001/"; options.Authority = "http://localhost:5000/"; }); }</pre>	

In the preceding code, two authentication handlers have been added: one for cookies and one for bearer.

❗ Note

Specifying the default scheme results in the `HttpContext.User` property being set to that identity. If that behavior isn't desired, disable it by invoking the parameterless form of `AddAuthentication`.

Selecting the scheme with the `Authorize` attribute

At the point of authorization, the app indicates the handler to be used. Select the handler with which the app will authorize by passing a comma-delimited list of authentication schemes to `[Authorize]`. The `[Authorize]` attribute specifies the authentication scheme or schemes to use regardless of whether a default is configured. For example:


C#

 Copy

```
[Authorize(AuthenticationSchemes = AuthSchemes)]
public class MixedController : Controller
{
    // Requires the following imports:
    // using Microsoft.AspNetCore.Authentication.Cookies;
    // using Microsoft.AspNetCore.Authentication.JwtBearer;
    private const string AuthSchemes =
        CookieAuthenticationDefaults.AuthenticationScheme + "," +
        JwtBearerDefaults.AuthenticationScheme;
}
```

In the preceding example, both the cookie and bearer handlers run and have a chance to create and append an identity for the current user. By specifying a single scheme only, the corresponding handler runs.

C#


 Copy

```
[Authorize(AuthenticationSchemes =
    JwtBearerDefaults.AuthenticationScheme)]
public class MixedController : Controller
{
}
```

In the preceding code, only the handler with the "Bearer" scheme runs. Any cookie-based identities are ignored.

Selecting the scheme with policies

If you prefer to specify the desired schemes in [policy](#), you can set the `AuthenticationSchemes` collection when adding your policy:

C#	 Copy
<pre>services.AddAuthorization(options => { options.AddPolicy("Over18", policy => { policy.AuthenticationSchemes.Add(JwtBearerDefaults.AuthenticationScheme); policy.RequireAuthenticatedUser(); policy.Requirements.Add(new MinimumAgeRequirement()); }); });</pre>	

In the preceding example, the "Over18" policy only runs against the identity created by the "Bearer" handler. Use the policy by setting the `[Authorize]` attribute's `Policy` property:

C#	 Copy
<pre>[Authorize(Policy = "Over18")] public class RegistrationController : Controller</pre>	

Use multiple authentication schemes

Some apps may need to support multiple types of authentication. For example, your app might authenticate users from Azure Active Directory and from a users database. Another example is an app that authenticates users from both Active Directory Federation Services and Azure Active Directory B2C. In this case, the app should accept a JWT bearer token from several issuers.

Add all authentication schemes you'd like to accept. For example, the following code in `Startup.ConfigureServices` adds two JWT bearer authentication schemes with different issuers:

C#	 Copy
<pre>public void ConfigureServices(IServiceCollection services) { // Code omitted for brevity</pre>	

```

services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddJwtBearer(options =>
    {
        options.Audience = "https://localhost:5000/";
        options.Authority = "https://localhost:5000/identity/";
    })
    .AddJwtBearer("AzureAD", options =>
    {
        options.Audience = "https://localhost:5000/";
        options.Authority =
"https://login.microsoftonline.com/eb971100-6f99-4bdc-8611-
1bc8edd7f436/";
    });
}

```

❗ Note

Only one JWT bearer authentication is registered with the default authentication scheme `JwtBearerDefaults.AuthenticationScheme`. Additional authentication has to be registered with a unique authentication scheme.

The next step is to update the default authorization policy to accept both authentication schemes. For example:

C#

 Copy

```

public void ConfigureServices(IServiceCollection services)
{
    // Code omitted for brevity

    services.AddAuthorization(options =>
    {
        var defaultAuthorizationPolicyBuilder = new
        AuthorizationPolicyBuilder(
            JwtBearerDefaults.AuthenticationScheme,
            "AzureAD");
        defaultAuthorizationPolicyBuilder =

        defaultAuthorizationPolicyBuilder.RequireAuthenticatedUser();
        options.DefaultPolicy =
        defaultAuthorizationPolicyBuilder.Build();
    });
}

```

As the default authorization policy is overridden, it's possible to use the `[Authorize]` attribute in controllers. The controller then accepts requests with JWT issued by the first or second issuer.

See [this GitHub issue](#) on using multiple authentication schemes.

Is this page helpful?

 Yes  No
