Application

Allowed Callback URLs

http://localhost:4200

After the user authenticates we will only call back to any of these URLs. You can specify multiple valid URLs by comma-separating them (typically to handle different environments like QA or testing). Make sure to specify the protocol (`https://`) otherwise the callback may fail in some cases. With the exception of custom URI schemes for native clients, all callbacks should use protocol `https://` . You can use Organization URL parameters in these URLs.

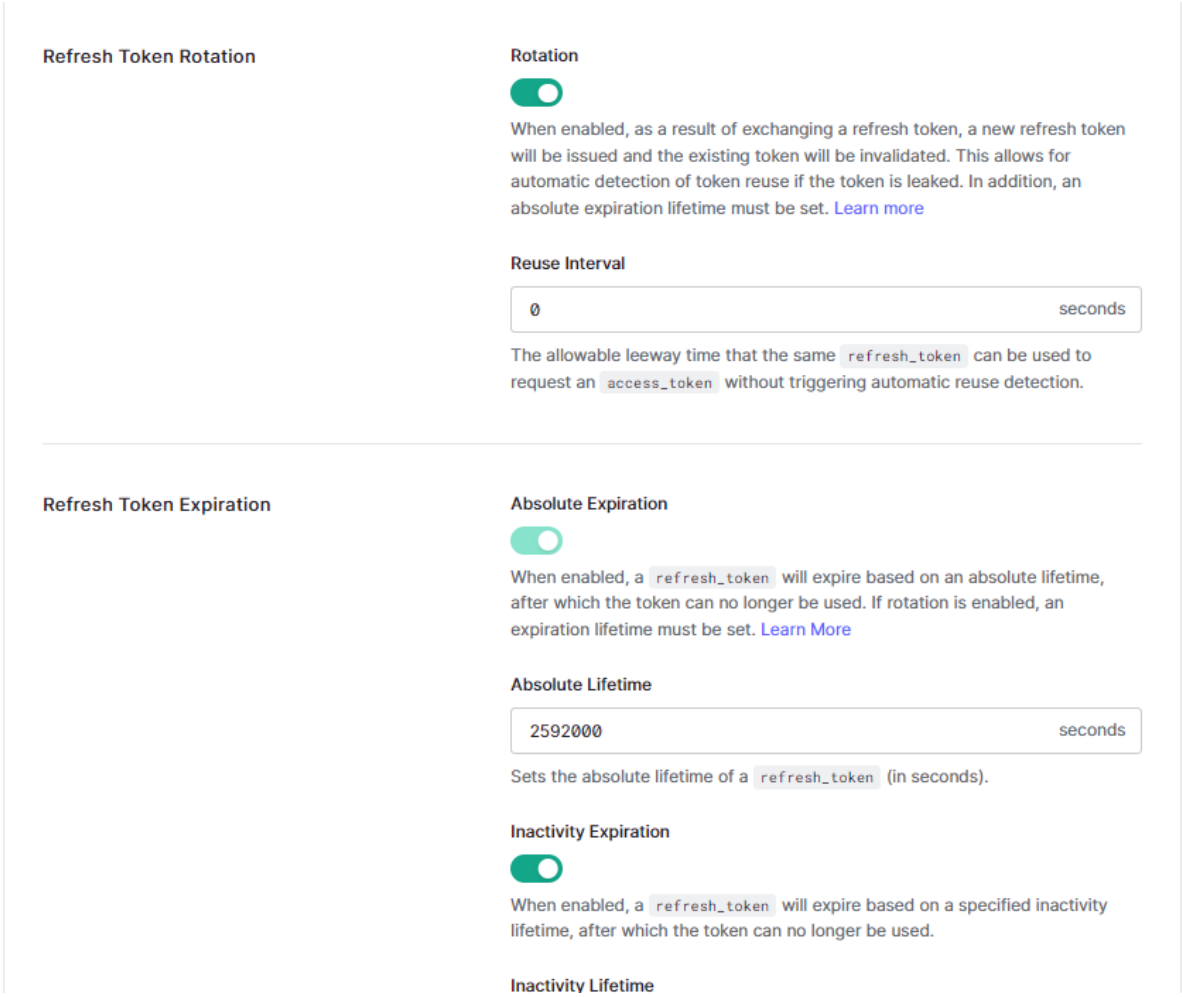
Allowed Logout URLs

http://localhost:4200

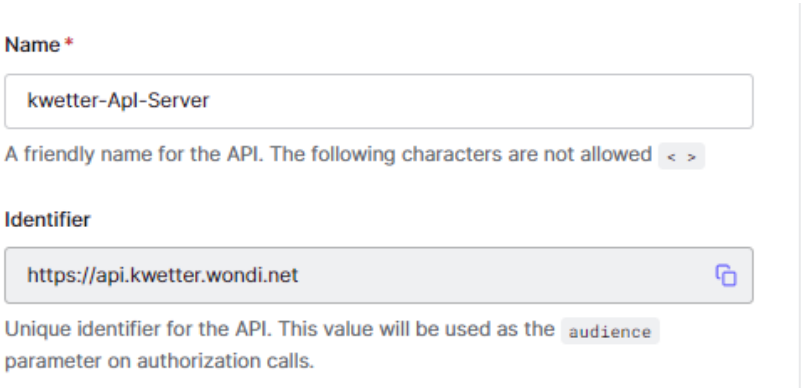
A set of URLs that are valid to redirect to after logout from Auth0. After a user logs out from Auth0 you can redirect them with the `returnTo` query parameter. The URL that you use in `returnTo` must be listed here. You can specify multiple valid URLs by comma-separating them. You can use the star symbol as a wildcard for subdomains (`*.google.com`). Query strings and hash information are not taken into account when validating these URLs. Read more about this at https://auth0.com/docs/authenticate/login/logout

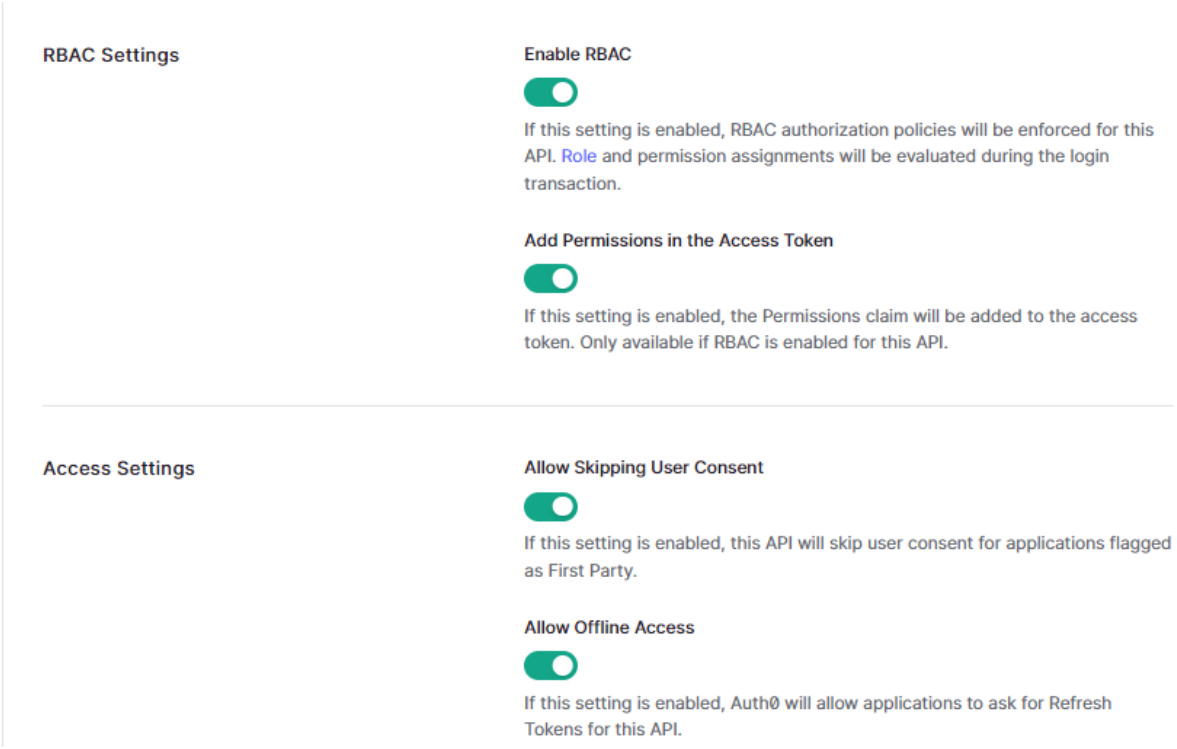
Allowed Web Origins

http://localhost:4200

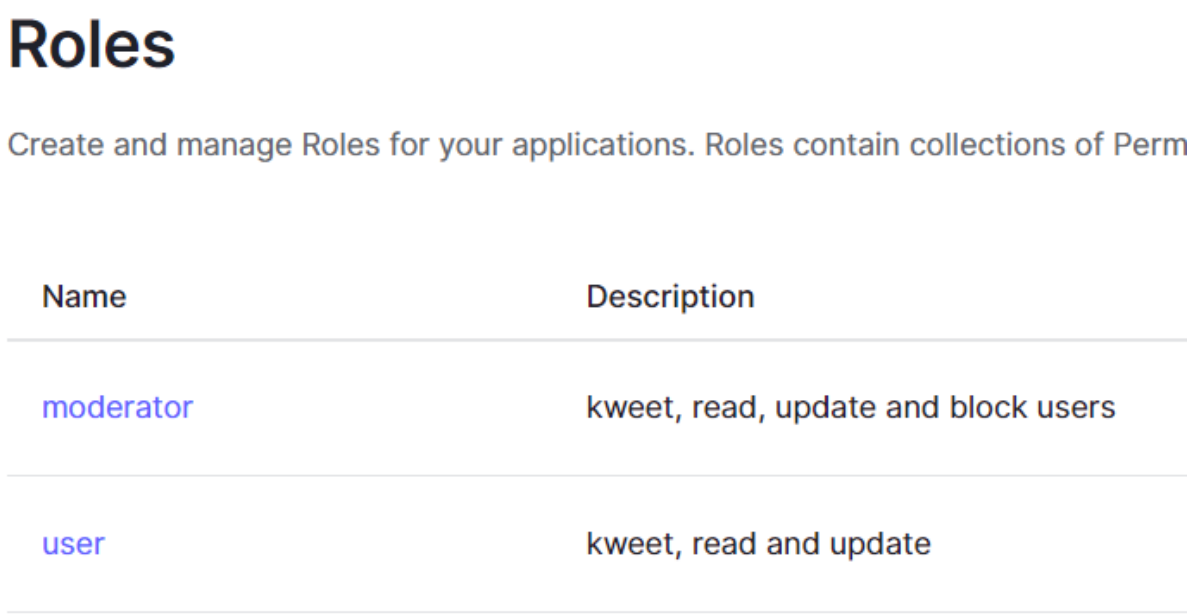

Refresh Token Rotation
Rotation
When enabled, as a result of exchanging a refresh token, a new refresh token will be issued and the existing token will be invalidated. This allows for automatic detection of token reuse if the token is leaked. In addition, an absolute expiration lifetime must be set. Learn more
Reuse Interval
0
seconds
The allowable leeway time that the same refresh_token can be used to request an access_token without triggering automatic reuse detection.
Refresh Token Expiration
Absolute Expiration
When enabled, a refresh_token will expire based on an absolute lifetime, after which the token can no longer be used. If rotation is enabled, an expiration lifetime must be set. Learn More
Absolute Lifetime
2592000
seconds
Sets the absolute lifetime of a refresh_token (in seconds).
Inactivity Expiration
When enabled, a refresh_token will expire based on a specified inactivity lifetime, after which the token can no longer be used.
Inactivity Lifetime

APi


Name *
kwetter-Apl-Server
A friendly name for the API. The following characters are not allowed < >
Identifier
https://api.kwetter.wondi.net
Unique identifier for the API. This value will be used as the audience parameter on authorization calls.


RBAC Settings
Enable RBAC
If this setting is enabled, RBAC authorization policies will be enforced for this API. Role and permission assignments will be evaluated during the login transaction.
Add Permissions in the Access Token
If this setting is enabled, the Permissions claim will be added to the access token. Only available if RBAC is enabled for this API.
Access Settings
Allow Skipping User Consent
If this setting is enabled, this API will skip user consent for applications flagged as First Party.
Allow Offline Access
If this setting is enabled, Auth0 will allow applications to ask for Refresh Tokens for this API.

Roles


Roles
Create and manage Roles for your applications. Roles contain collections of Perm
Name
Description
moderator
kweet, read, update and block users
user
kweet, read and update

Rules:

Script

```
function (user, context, callback) {
  const assignedRoles = (context.authorization || {}).roles;
  const idTokenClaims = context.idToken || {};

  idTokenClaims['https://dev-fuh1akv6.eu.auth0.com/roles'] = assignedRoles;

  callback(null, user, context);
}
```

Startup

```
public static IServiceCollection AddCustomAuthentication(this IServiceCollection services,
    IConfiguration configuration)
{



    services.AddAuthentication(options =>
    {
        options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
        options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

    }).AddJwtBearer(options =>
    {
        options.Authority = $"https://{configuration["Auth0:authZeroDomain"]}/";
        options.Audience = configuration["Auth0:authZeroAudience"];
        options.TokenValidationParameters = new TokenValidationParameters
        {
            NameClaimType = "Roles",
            RoleClaimType = configuration.GetValue<string>("https://dev-fuh1akv6.eu.auth0.com/roles")
        };
    });

    return services;
}
```

App setting file in Backend

```
"AllowedHosts": "*",
"Auth0": {
  "authZeroDomain": "dev-fuh1akv6.eu.auth0.com",
  "authZeroAudience": "https://api.kwetter.wondi.net"
},
```

Client-side credential

```
window.env.authZeroDomain = 'dev-fuh1akv6.eu.auth0.com';
window.env.authZeroEndpoint = 'https://dev-vabm4wba.eu.auth0.com/';
window.env.authZeroClientId = 'xYho03V3p8OSKkG97rhs0swEU8Ww3KZi';
window.env.authZeroRoleNamespace = 'https://dev-fuh1akv6.eu.auth0.com/roles';
window.env.authZeroAudience = 'https://api.kwetter.wondi.net';
window.env.gatewayUrl = 'https://localhost:5001';
```

Meta Data

app_metadata

```
{
  "roles": "user"
}
```

Resource

```
[HttpPut]
[Authorize(Roles = "user")]
[ProducesResponseType(StatusCodes.Status204NoContent)]
[ProducesResponseType(StatusCodes.Status403Forbidden)]
0 references
public async Task<IActionResult> EditKweet(TextKweetViewModel textKweet)
{
    var currentUser = User.FindFirst(ClaimTypes.NameIdentifier).Value;
    if (currentUser != textKweet.UserId)
    {
        _logger.LogInformation($"User {currentUser} attempted to post a kweet as user {
        return Unauthorized($"User {currentUser} attempted to post a kweet as user {tex
    }
```