Understanding Refresh Tokens

Learn about refresh tokens and the role they serve in the authorization process

CONTACT US

WHAT ARE REFRESH TOKENS?

Modern secure applications often use access tokens to ensure a user has access to the appropriate resources, and these access tokens typically have a limited lifetime. This is done for various security reasons: for one, limiting the lifetime of the access token limits the amount of time an attacker can use a stolen token. In addition, the information contained in or referenced by the access token could become stale.

When access tokens expire or become invalid but the application still needs to access a protected resource, the application faces the problem of getting a new access token without forcing the user to once again grant permission. To solve this problem, OAuth 2.0 introduced an artifact called a refresh token. A refresh token allows an application to obtain a new access token without prompting the user.

OBTAINING REFRESH TOKENS

A refresh token can be requested by an application as part of the process of obtaining an access token. Many authorization servers implement the refresh token request mechanism defined in the OpenID Connect specification. In this case, an application must include the `offline_access` scope when initiating a request for an authorization code. After the user successfully authenticates and grants consent for the application to access the protected resource, the application will receive an authorization code that can be exchanged at the token endpoint for both an access and a refresh token.

USING REFRESH TOKENS

When a new access token is needed, the application can make a `POST` request back to the token endpoint using a grant type of `refresh_token` (web applications need to include a client secret). To use a refresh token to obtain a new ID token, the authorization server would need to support OpenID Connect and the scope of the original request would need to include `openid` .

While refresh tokens are often long-lived, the authorization server can invalidate them. Some of the reasons a refresh token may no longer be valid include:

- the authorization server has revoked the refresh token
- the user has revoked their consent for authorization
- the refresh token has expired
- the authentication policy for the resource has changed (e.g., originally the resource only used usernames and passwords, but now it requires MFA)

Because refresh tokens have the potential for a long lifetime, developers should ensure that strict storage requirements are in place to keep them from being leaked. For example, on web applications, refresh tokens should only leave the backend when being sent to the authorization server, and the backend should be secure. The client secret should be protected in a similar fashion. Mobile applications do not require a client secret, but they should still be sure to store refresh tokens somewhere only the client application can access.

REFRESH TOKENS AT AUTH0

With Auth0, you can get a refresh token when using the **Authorization Code Flow** (for regular web or native/mobile apps), the **Device Flow**, or the **Resource Owner Password Grant**. All of Auth0's main SDKs support acquiring, using, and revoking refresh tokens out of the box, without you having to worry about formatting messages. Languages with SDK support include Node.js, .NET, PHP, and many more!

To learn more about refresh tokens at Auth0, including how to revoke them, check out the refresh token documentation.

Contact Us

Name

Company

Email

Role

Software Developer

Message

You hate junk? So do we.

We'd love it if you ticked the box below so we can continue sending this kind of content your way. You can always change your mind and opt out at any time. (And if you think what we're sending IS junk, please let us know so we can fix it.)

○ Email me about product updates, learning resources, feedback surveys, special offers and events.

● No, please just send me this one resource.

SEND

PRODUCT

Pricing

Why Auth0

How It Works

COMPANY

About Us

Blog

Jobs

Press

SECURITY

Availability & Trust

Security

White Hat

LEARN

Help & Support

Documentation

Open Source

EXTEND

Lock

WordPress

API Explorer

CONTACT

10800 NE 8th Street
Suite 600
Bellevue, WA 98004

+1 (888) 235-2699

+1 (425) 312-6521

SUPPORT

+1 (425) 559-9554

Follow @auth0 29.4K followers

in متابعة 42,238

Privacy Policy Terms of Service