<https://auth0.com/docs/tokens/access-tokens/get-access-tokens#control-access-token-audience>

https://auth0.com/docs/tokens/access-tokens/validate-access-tokens

Validate JSON Web Tokens

All Auth0-issued JWTs have a JSON Web Signature (JWS), meaning they are signed rather than encrypted. If you need to validate a JSON Web Encryption (JWE), see [RFC 7519](https://tools.ietf.org/html/rfc7519#section-7.2) for instructions specific to that type of JWT. Most JWT libraries take care of JWT validation for you. Visit JWT.io to find a JWT library for your platform and programming language.

To validate a JWT, your application needs to:

1. Check that the JWT is well formed.
2. Check the signature.
3. Check the standard claims.

If any of these steps fail, then the associated request must be rejected.

Check that the JWT is well-formed

Ensure that the JWT conforms to the [structure of a JWT](https://auth0.com/docs/tokens/json-web-tokens/json-web-token-structure). If this fails, the token is considered invalid, and the request must be rejected.

1. Verify that the JWT contains three segments, separated by two period ('.') characters.
2. Parse the JWT to extract its three components. The first segment is the Header, the second is the Payload, and the third is the Signature. Each segment is base64url encoded.
3. Base64url-decode the Header, ensuring that no line breaks, whitespace, or other additional characters have been used, and verify that the decoded Header is a valid JSON object.
4. Base64url-decode the Payload, ensuring that no line breaks, whitespace, or other additional characters have been used, and verify that the decoded Payload is a valid JSON object.

### Check signature

The last segment of a JWT is the signature, which is used to verify that the token was signed by the sender and not altered in any way. The Signature is created using the Header and Payload segments, a signing algorithm, and a secret or public key (depending on the chosen signing algorithm).

To verify the signature, you will need to:

1. Check the signing algorithm.
   1. Retrieve the alg property from the decoded Header.
   2. Ensure that it is an allowed algorithm. Specifically, to avoid certain attacks, make sure you disallow none.
   3. Check that it matches the algorithm you selected when you registered your Application or API with Auth0.
2. Confirm that the token is correctly signed using the proper key.

To verify that the signature is correct, you need to generate a new Base64url-encoded signature using the public key (RS256) or secret (HS256) and verify that it matches the original Signature included with the JWT:

* 1. Take the original Base64url-encoded Header and original Base64url-encoded Payload segments (Base64url-encoded Header + "." + Base64url-encoded Payload), and hash them with SHA-256.
  2. Encrypt using either HMAC or RSA (depending on your selected signing algorithm) and the appropriate key.
  3. Base64url-encode the result.
     1. **For RS256**: Retrieve the public key from the JSON web key set (JWKS) located by using the [Auth0 Discovery endpoint](https://auth0.com/docs/tokens/json-web-tokens/json-web-key-sets/locate-json-web-key-sets). For debugging purposes, you can visually inspect your token at jwt.io; for this purpose, you can also locate your public key in the Auth0 Dashboard. Look in **Applications**>**Settings**>**Advanced Settings**>**Certificates** and locate the **Signing Certificate** field.
     2. **For HS256**: Retrieve the client\_secret from Auth0's Management API using the Get a Client endpoint. For debugging purposes, you can visually inspect your token at [jwt.io](https://jwt.io/); for this purpose, you can also locate your secret in the Auth0 Dashboard. For applications, look in **Settings** and locate the **Client Secret** field. For APIs, look in **Settings** and locate the **Signing Secret** field. (Note that this field is only displayed for APIs using the HS256 signing algorithm.)

In the case of RSA, you can also use the following steps:

1. **Calculate current hash value**. A hash value of the signed message is calculated. For this calculation, the same hashing algorithm is used as was used during the signing process. The obtained hash value is called the current hash value because it is calculated from the current state of the message.
2. **Calculate the original hash value**. The digital signature is decrypted with the same encryption algorithm that was used during the signing process. The decryption is done by the public key that corresponds to the private key used during the signing of the message. As a result, we obtain the original hash value that was calculated from the original message during the previous step of the signing process (the original message digests).
3. **Compare the current and the original hash values**. If the two values are identical, the verification is successful and proves that the message has been signed with the private key that corresponds to the public key used in the verification process. If the two values differ, the digital signature is invalid and the verification is unsuccessful.

If the generated signature does not match the original Signature included with the JWT, the token is considered invalid, and the request must be rejected.

### Check standard claims

Before using the token, you should retrieve the following standard claims from the decoded payload and perform the following checks:

* **Token expiration** (exp, Unix timestamp): The expiration date/time must be after the current date/time.
* **Token issuer** (iss, string): The issuing authority inside the token must match the issuing authority (issuer) identified in your Auth0 tenant's discovery document, which exists at https://YOUR\_DOMAIN/.well-known/openid-configuration.

Additional checks are required depending on whether the JWT you are validating is an ID token or an access token.

## **Verify RS256 signed tokens**

1. Go to [Dashboard > Applications](https://manage.auth0.com/#/applications).
2. Go to the **Settings** tab and open **Advanced Settings**.
3. Open the **Certificates** tab to see the **Public Key** in the **Signed Certificate** field.
   1. To use the Public Key to verify a JWT signature on [JWT.io](https://jwt.io/), copy the Public Key and past it in the **Public Key or Certificate** field under **Verify Signature** section on the JWT.io website.
   2. To verify the signature of a token from one of your applications, we recommend that you get the Public Key from your tenant's JWKS here: https://YOUR\_DOMAIN/.well-known/jwks.json

Use access token

Access tokens are used in token-based authentication to allow an application to access an API. For example, a Calendar application needs access to a Calendar API in the cloud so that it can read the user's scheduled events and create new events.

Once an application has received an access token, it will include that token as a credential when making API requests. To do so, it should transmit the access token to the API as a **Bearer** credential in an HTTP **Authorization** header.

Example:

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL2V4YW1wbGUuYXV0aDAuY29tLyIsImF1ZCI6Imh0dHBzOi8vYXBpLmV4YW1wbGUuY29tL2NhbGFuZGFyL3YxLyIsInN1YiI6InVzcl8xMjMiLCJpYXQiOjE0NTg3ODU3OTYsImV4cCI6MTQ1ODg3MjE5Nn0.CA7eaHjIHz5NxeIJoFK9krqaeZrPLwmMmgI\_XiQiIkQ

In this example, the Access Token is a JWT that decodes to the following claims:



Before permitting access to the API using this token, the API must [validate the access token](https://auth0.com/docs/tokens/access-tokens/validate-access-tokens).

Once the Access Token has been successfully validated, the API can be sure that:

* The token was issued by Auth0.
* The token was issued to an application being used by a user with an identifier of usr\_123.
* The user granted the application access to read from and write to their calendar.

The API can now process the request, allowing the application to read from and write to user usr\_123's calendar.