



# Widget 2.0 JWT authentication

Last Modified on 05/02/2024 11:09 am EDT

When using the widget with private knowledge bases, or knowledge bases that have restricted content that you want your customers to be able to access, SSO authentication is required.

To implement SSO authentication for your readers in the widget, you may choose between OAuth2 or JSON Web Tokens (JWT).

With JWT authentication, you'll need a JWT client secret to retrieve the token, and your widget embed script will need to request the token to authenticate and display knowledge base content to your readers.

## Generate client secret

To retrieve a JWT token, you will first need to generate the Client Secret for your knowledge base:

1. Go to **Settings > Widget**.
2. Scroll to the **Admin Settings** section.
3. In the **JWT authentication** subsection, click the button to **Generate Client Secret**.

**JWT Authentication**  Secure widget access using JWT

If your site is not public facing, you will need to choose an authentication option for end-users to see article content. For more details, view the [documentation](#).

Generate Client Secret 

4. Once you click this button, your Client Secret will be populated.

**JWT Authentication**  Secure widget access using JWT

If your site is not public facing, you will need to choose an authentication option for end-users to see article content. For more details, view the [documentation](#).

### JWT Secret

b \_\_\_\_\_ e

Do NOT share your JWT secret.

 Regenerate JWT Secret

Now that you have your client secret, you can use it to request an access token from the JWT token endpoint. We recommend modifying the default embed script to include a section that will call the `_ko19.onReady` method and

get the token within that method. See below for overall workflow examples and a sample call to this method.

## Working with JWT authentication

For information about what JWT is and how you can use it, you can refer to this [JWT introduction article](#). For use with the widget, please refer to the following requirements:

### JWT header

```
// Encryption algorithm must be HS256
{
  "alg": "HS256",
  "typ": "JWT"
}
```

### JWT payload



JWT payloads are not encrypted and should **NOT INCLUDE** any sensitive information about the reader.

If you need to pass sensitive information about your readers, use the server side OAuth2 workflow.

```
{
  "iss" => "app.knowledgeowl.com", // Issuer - *required
  "aud" => "YOUR.KNOWLEDGEBASE.URL", // Audience - *required
  "iat" => unixTimestamp, // Issued At - *required
  "nbf" => unixTimestamp - 1000, // Not Before - *required
  "exp" => unixTimestamp + 1000, // Expires - *required
  "reader_ssoId" => endUserID, // Unique ID - *required
  "reader_username" => endUserEmail, // Email or Username - *required
  "reader_groups" => "Support,Admin" // Comma separated list of reader group names - optional
}
```

## Example authentication code

```
var _ko19 = _ko19 || {};
_ko19._pc = '123xxxxxx-123xxxxxx';
_ko19.base_url = '//yourkb.knowledgeowl.com';
!function() {
  var ko = document.createElement('script');
  ko.type = 'text/javascript';
  ko.async = true;

  ko.src = `${_ko19.base_url}/widget-app/assets/js/load.js`;
  document.head.appendChild(ko);

  /**
   * The widget is loaded and ready to receive commands
   */
}
```

```

    _ko19.onReady = function () {
    // Get the JWT token and pass it to the widget
    getToken();
    }

/**
 * Form the authentication object and pass it to the widget for authentication
 */
getToken = function() {
    /** Example of hitting an endpoint to get the JWT token */
    /**
     * $.get('my.jwtendpoint.url', function(response){
     *   if(response.token) {
     *     var JWT = response.token;
     *   }
     * });
     */

    /** Example where the JWT already exists on the page */
    var JWT = preDefinedVar; // 'eyJ0.eyJpc3Mi.dKSb41xr' - Example of shortened value for JWT
    var authToken = {'type': 'jwt', 'token': JWT};

    // Use the formed object with the JWT to authenticate
    authWidget(authToken);
}

/**
 * The widget will pass the response back from the authentication call here.
 * This can be useful for debugging purposes if any errors occur
 */
_ko19.onAuthResp = function(response) {
    // Take some kind of action if the authentication fails
    if(response.error)
        console.log(response);
}

/**
 * The widget will call this function whenever authorization fails
 * This can occur before the user is logged in, and after the authentication expires
 * With most workflows, you will want to re-authorize the end-user
 * when this event occurs
 */
_ko19.onAuthError = function () {
    // Take an action when the user is not logged in
    getToken()
}

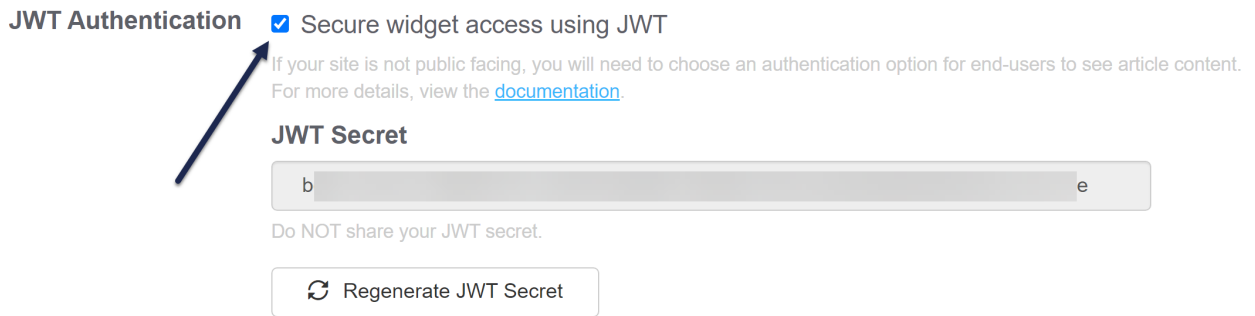
/**
 * Pass your token as JSON into the widget's authenticate function
 * the JSON format should be as follows:
 * {'type': 'jwt|oauth', 'token': token_value}
 */
authWidget = function(authToken) {
    _ko19.authenticate(authToken);
}
}();

```

## Enable JWT

When your script is ready for testing/usage, set the widget to use JWT authentication:

1. Go to **Settings > Widget**.
2. Scroll to the **Admin Settings** section.
3. In the **JWT authentication** subsection, check the box next to "Secure widget access using JWT".




**JWT Authentication**  Secure widget access using JWT

If your site is not public facing, you will need to choose an authentication option for end-users to see article content. For more details, view the [documentation](#).

**JWT Secret**

b e

Do NOT share your JWT secret.

 Regenerate JWT Secret

4. **Save the settings.**

## Generate a new JWT secret

For security reasons, sometimes you may want to generate a new JWT secret. You can do this directly yourself in **Settings > Widget**.

Once you regenerate the JWT secret, the existing JWT secret will no longer function, so your widget authentication will basically be broken until you update it to use the new secret.

To regenerate your secret:

1. Go to **Settings > Widget**.
2. Scroll to the **Admin Settings** section.
3. In the **JWT Authentication** subsection, click the **Regenerate JWT Secret** button.
4. This will open a pop-up confirmation to ensure you want to make this change. As the message states, regenerating the secret will invalidate the old secret (which basically breaks your existing widget authentication until you update it to use the new secret).

## Regenerate JWT Secret



**⚠ This action cannot be undone.**

Regenerating your existing JWT secret will invalidate the existing secret. Any reference to the old secret will need to be updated.

Are you sure you want to regenerate?



Confirmation message for regenerating your JWT secret

5. To proceed with regenerating the secret, select **Regenerate**.

6. The pop-up will disappear and your new JWT secret will be displayed.

You can now update your authentication scripts/processes to use that secret instead.