



Widget 2.0 OAuth2 authentication

Last Modified on 06/23/2025 12:49 pm 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 JWT.

To use OAuth2, you'll need to generate a client ID and secret within KnowledgeOwl, and your widget embed script will need to use that secret to authenticate and display knowledge base content to your readers.

Generate Client Secret

To retrieve an OAuth2 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 **OAuth authentication** subsection, click the button to **Generate Client Secret**.

OAuth Authentication ☒ Secure widget access using OAuth2

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** and **Client ID** will be populated.

OAuth Authentication ☐ Secure widget access using OAuth 2.0

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](#).

Client ID

6 4

Client Secret

c 1

Do NOT share your client secret.

Regenerate OAuth Secret

Now that you have your client secret and client ID, you can use them to request an access token from the OAuth

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.

Request Token

```
//curl request for token with reader information
curl -u clientID:clientSecret https://app.knowledgeowl.com/oauth2/token
-d
"grant_type=client_credentials&reader[ssoid]=UID&reader[username]=reader@mysite.com&reader[groups]=Group1,Group2"
```

Example Response

```
{ "access_token": "1234567890987654321234567890987654321234", "expires_in": 3600, "token_type": "Bearer", "scope": null
}
```

Widget Embed Code Example

```
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.onLoad = function () {
    // Retrieve the OAuth2 token from a server side endpoint
    getToken();
  }

  /**
   * Form the authentication object and pass it to the widget for authentication
   */
  getToken = function() {
    // Get the OAuth2 token from a server side endpoint
    $.get('my.oauth2.endpoint', function(response){
      if(response.token) {
        // Example value for response.token = '1234567890987654321234567890987654321234'
        var authToken = { 'type': 'oauth', 'token': response.access_token };

        // Send the object for authentication
        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 an action of 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 () {
    console.log("not authorized trigger");
}

/**
 * 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 OAuth

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

1. Go to **Settings > Widget**.
2. Scroll to the **Admin Settings** section.
3. In the **OAuth authentication** subsection, check the box next to "Secure widget access using OAuth2".

OAuth Authentication

☒ Secure widget access using OAuth 2.0

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](#).


Client ID

6 4

Client Secret

c 1

Do NOT share your client secret.

 Regenerate OAuth Secret

4. **Save the settings.**

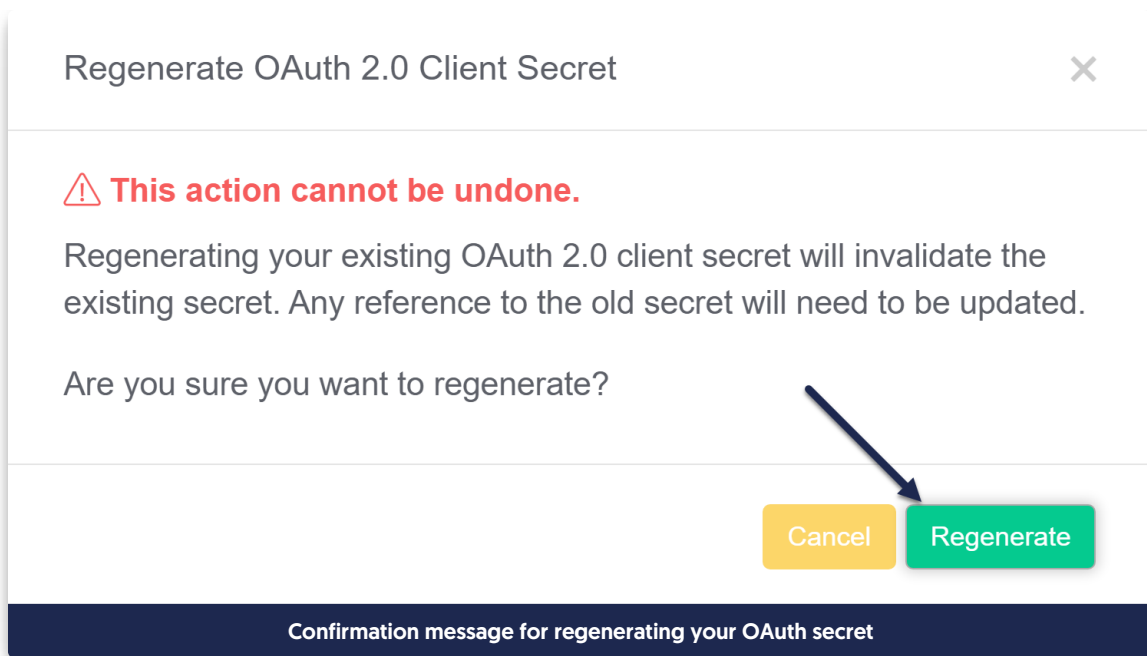
Regenerate OAuth Secret

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

Once you regenerate the OAuth secret, the existing OAuth 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 **OAuth Authentication** subsection, click the **Regenerate OAuth 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).



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

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

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