JWT SSO for the Embedded Widget

Integrating the widget

Use this guide if you embed the HelpCenter.io Smart Widget inside an authenticated SaaS product and you want your end-users to see your private help-center content without going through a separate login.

How it works

1. Your application authenticates the user the way it always has.
2. When the user lands on a page where you embed the widget, your backend mints a short-lived JWT identifying them.
3. The host snippet hands the JWT to the widget on init.
4. The widget includes Authorization: Bearer <jwt> on every call to HelpCenter.io.
5. HelpCenter.io validates the signature against your shared secret, identifies the user, and serves articles only if Site::visibility allows.

No cookies are involved. No browser-level third-party cookie issues. No login screen.

One-time setup (HelpCenter.io dashboard)

1. Go to **Settings → Security → JWT SSO**.

2. Set your **Login URL** (used by the redirect-based flow if a user lands on the help-center site directly).

3. Click **Generate** to mint a shared secret (or paste your own — minimum 64 characters).

4. Optionally set an **Issuer** and **Audience** claim to enforce. If set, every JWT you sign must include matching iss and aud claims or it is rejected.

5. Optionally set a **Token TTL** (defaults to 300 seconds). The widget surface rejects any JWT whose iat is older than this even if exp is still in the future — defense in depth against leaked tokens.

6. Save.

Backend: minting the JWT

HS256 only. Sign with the shared secret.

Required claims

Claim	Type	Notes
jti	string	Unique per token. Used for replay protection.
iss	string	Your issuer — typically your app's domain. Must match the configured Issuer if set.

iat	int	Issued-at, Unix seconds.
exp	int	Expiry, Unix seconds. Always required. Keep short — 5 minutes is typical.
email	string	The user's email.
name	string	The user's display name.

Optional claims

Claim	Type	Notes
external_id	string	Your stable user ID. Recommended — lets us track the same user across email changes.
lang	string	Two-letter language code. Defaults to the site's default language.
avatar_url	string	Public URL to the user's avatar.

custom_fields	object	Free-form. Surfaced on the User record for analytics.
role	string	viewer, editor, or admin. Defaults to viewer.
aud	string	Audience. Must match the configured Audience if set.

Example (Node / jsonwebtoken)

```
const jwt = require('jsonwebtoken');

function mintHelpcenterJwt(user) {
    return jwt.sign({
        jti: `${user.id}.${Date.now()}.${Math.random().toString(36).slice(2)}`,
        iss: 'app.yourcompany.com',
        iat: Math.floor(Date.now() / 1000),
        exp: Math.floor(Date.now() / 1000) + 300,
        email: user.email,
        name: user.fullName,
        external_id: String(user.id),
        role: 'viewer',
    }, process.env.HELPCENTER_SHARED_SECRET, { algorithm: 'HS256' });
}
```

Example (PHP / firebase/php-jwt)

```
use Firebase\JWT\JWT;

function mint_helpcenter_jwt(User $user): string
```

```
{
    return JWT::encode([
        'jti' => $user->id . '.' . microtime(true),
        'iss' => 'app.yourcompany.com',
        'iat' => time(),
        'exp' => time() + 300,
        'email' => $user->email,
        'name' => $user->name,
        'external_id' => (string) $user->id,
        'role' => 'viewer',
    ], config('helpcenter.shared_secret'), 'HS256');
}
```

Frontend: passing the JWT to the widget

If you already embed the widget, you have a snippet that looks like this:

```
<script>
    window.hcOptions = {
        app_id: 'YOUR_WIDGET_ID',
    };
</script>
<script src="https://helpcenter.io/js/init.js" async></script>
```

To add SSO, set jwt to the token your backend just minted, and define onAuthExpired so the widget can ask for a fresh one when the current one runs out:

```
<script>
    window.hcOptions = {
        app_id: 'YOUR_WIDGET_ID',
        jwt: '{{ minted_jwt }}',
        onAuthExpired: async function () {
            const resp = await fetch('/helpcenter-token', { credentials: 'inc
            const json = await resp.json();
            return json.jwt;
```

```
        },
    };
</script>
<script src="https://helpcenter.io/js/init.js" async></script>
```

The onAuthExpired callback runs whenever the widget receives a 403 SITE_AUTH_REQUIRED response from the API. Return the new JWT (string) — the widget uses it immediately without reloading.

If you need to push a fresh token outside the expiry flow (e.g., user switched accounts), call window.hcWidget.setJwt(newJwt).

Security notes

- **TTL is your friend.** Five minutes is a reasonable default. The shorter the TTL, the smaller the window if a token leaks.
- **Token is in the URL hash, not the query string.** The widget snippet appends #jwt=<token> to the iframe URL. Hashes are not sent to the server, so the token never lands in proxy or access logs. The widget JS reads it on load and immediately strips it from the URL.
- **One use per** jti **on the widget surface.** A replayed token is rejected even if the signature and exp are still valid.
- **Clock skew tolerance is 30 seconds** on both iat and exp. Sync your servers via NTP — anything more than 30 seconds of drift will cause sporadic rejections.

Failure modes

When the widget cannot authenticate, the API returns:

```
{
    "status": "error",
    "code": "SITE_AUTH_REQUIRED",
    "message": "This help center requires authentication."
}
```

...with HTTP 403. The widget triggers onAuthExpired on the host page. If the host returns a fresh JWT, the widget transparently retries. If not, the widget renders an unauthenticated state (no articles visible).

For diagnosis, server-side logs include a widget_jwt.rejected event with a reason field:

Reason	What it means
jwt_invalid_signature	Signature didn't verify. Wrong secret?
jwt_missing_required_claim	One of jti/iss/iat/exp/email/name is missing or empty.
jwt_expired	exp is in the past (past the 30-second skew window).
jwt_iat_in_future	iat is more than 30 seconds in the future. Server clock skew.
jwt_too_old	iat is older than the configured Token TTL.

jwt_issuer_mismatch	iss claim doesn't match the configured Issuer.
jwt_audience_mismatch	aud claim doesn't match the configured Audience.
jwt_replayed	This jti has already been used on this site.
user_banned	The provisioned local user is suspended.

What you don't have to do

- You do NOT need to manage cookies on embed.helpcenter.io. No third-party-cookie configuration. Safari ITP and Firefox don't interfere.
- You do NOT need to expose your shared secret to the browser — it stays on your backend.
- You do NOT need to host a callback URL for HelpCenter.io to redirect to. The widget flow is pure header auth.
- You do NOT need to pre-create users in the HelpCenter.io dashboard. The first successful JWT for a given email/external_id auto-provisions the user as a viewer of your help center.