The complete guide to integrating the Smart Widget

Integrating the widget

The Smart Widget drops a searchable help center into the corner of any page. Most teams install it in two minutes from the dashboard and never touch the code again. But the widget can do a lot more than sit in the corner: you can open a specific article from a question mark next to a heading, run it inside a private app behind your own login, hand search dead-ends to your own contact form, and more.

This guide walks through every supported way to integrate it. Each part stands on its own, so jump to what you need.

Install it

Your dashboard hands you the snippet. Go to Settings, then Widget, activate the widget if you have not already, and copy the code from the Installing the widget section. It looks like this:

```
<script type="text/javascript">(function(){
  var s = document.createElement('script');
  s.src = "//helpcenter.io/js/init.js";
  s.async = true;
  window.hcOptions = { app_id: 'YOUR_WIDGET_ID' };
  document.body.appendChild(s);
})();</script>
```

Paste it right before the closing </body> tag. One more step that trips people up: in the widget settings, set the Site URL field to the domain where you placed it (you can list several, separated by commas, like localhost for testing and example.com for production). Without that, the widget will not appear.

Everything else in this guide is something you add to that window.hcOptions object, or a method you call after the widget loads.

The options you can set

hcOptions accepts more than the dashboard snippet shows. The ones you will reach for most:

- app_id – your widget id. Required.
- position – set it to 'bottom-left' to move the launcher to the other corner. The default is bottom-right.
- lang – a two-letter code to force a language, like 'es'. Otherwise the widget picks one for you.
- category – a category id to preload that category's articles when the widget opens.
- href – a /content/<slug> path to open a specific article on load.
- showButton – set it to false to hide the default launcher and open the widget yourself.

A note on looks: the launcher's color, icon, and heading come from your dashboard widget settings, not from the snippet. From the page you control position and language, not styling.

Control it from your own code

Once the widget loads, it puts a small API on the page at window.hcWidget. Use it to wire the widget to your own buttons and links.

```
window.hcWidget.show();      // open the panel
window.hcWidget.hide();      // collapse it back to the launcher
window.hcWidget.hideAll();   // remove the widget from the page
window.hcWidget.setCategory(801);                // load a category's articl
window.hcWidget.setHrefAndOpen('/content/slug');  // open a specific article
```

A common pattern is to hide the default button with showButton: false and open the widget from your own "Help" link with show().

Open a specific article from a link

This is the question mark next to a heading that opens the matching help article right there, without sending the reader off to your help center.

Give any link a /content/<slug> URL and a data-hc-article attribute. The widget intercepts the click and opens that article inside the panel (or in a side panel, if you are running in stealth mode):

```
<h2>Billing
  <a href="https://acme.helpcenter.io/content/how-billing-works-AbCdE"
     data-hc-article="1" aria-label="Open help">?</a>
</h2>
```

You can do the same thing in code with window.hcWidget.setHrefAndOpen('/content/how-billing-works-AbCdE'). Articles are targeted by their URL or slug, the part after /content/, not by a numeric id.

Run it without the default launcher

When you want the widget to feel built into your own interface, hide its button and drive it yourself. Set showButton: false and the widget loads invisibly, opening as a full-height side panel when you call show() or when a data-hc-article link is clicked.

```
window.hcOptions = { app_id: 'YOUR_WIDGET_ID', showButton: false };
```

Hand off to your own contact form

When a visitor searches and finds nothing, you might want to open your own ticket form or live chat rather than the widget's built-in contact step. Define onContactsRequest and the widget calls it at that moment:

```
window.hcOptions = {
  app_id: 'YOUR_WIDGET_ID',
  onContactsRequest: function () {
    openMySupportForm();
  }
};
```

Put it behind your login with SSO

If your help center is private and lives inside an authenticated product, you do not want your users to hit a second login. Your app already knows who they are. It mints a short-lived JSON Web Token, hands it to the widget, and the widget trusts that signed token.

You give the widget two things together: the first token in jwt, and a way to get more in onAuthExpired. Both are required. Widget tokens are single-use, so a jwt on its own authenticates only the very first request and everything after it fails. onAuthExpired is what the widget calls to mint each fresh token:

```
window.hcOptions = {
  app_id: 'YOUR_WIDGET_ID',
  jwt: 'THE_TOKEN_YOUR_BACKEND_MINTED',
  onAuthExpired: async function () {
    const r = await fetch('/helpcenter-token', { credentials: 'include' });
    const { jwt } = await r.json();
    return jwt;
```

```
  }
};
```

The token travels to the widget in the iframe's URL fragment, never the query string, so it stays out of server and proxy logs. If you need to push a new token outside the expiry flow, for example when a user switches accounts, call window.hcWidget.setJwt(newToken).

The full reference for minting tokens, the required and optional claims, token lifetime, and replay protection lives in its own article: "JWT SSO for the Embedded Widget." Start there for the backend side.

AI answers

On the Catalyst plan, or with the AI add-on on Growth and up, the widget's search answers questions in plain language instead of only listing articles. There is nothing to switch on from the page. Load the widget as usual, open it, and start typing a question. The AI answer streams in for plans that include it.

Embeddable FAQ sections

Separate from the widget, you can drop a curated block of FAQs straight onto a page. Build a section in your dashboard under Settings, then Widget, in the Your Embeddable FAQ Sections part, and you get a snippet like this:

```
<div class="hc-faq-section" data-id="YOUR_SECTION_KEY" data-type="full"></div>
<script src="https://helpcenter.io/js/faqs.js" async></script>
```

data-type="full" renders a section grouped by category, either with category tabs or as a stacked accordion depending on the layout you pick in the dashboard. Leave it as default for a simple curated list. To render in another language, set window.hcOptions = { lang: 'es' } before the script.

FAQ sections are curated, so they do not include a search box. When you want visitors to search your whole knowledge base from inside your page, embed the full help center with JavaScript instead, or use the Smart Widget.

A note on Content-Security-Policy

If your site sends a Content-Security-Policy header, allow the widget's domains or it will be blocked from loading. You need the loader script, the widget's iframe, and its data calls:

- script-src https://helpcenter.io
- frame-src https://embed.helpcenter.io
- connect-src https://embed.helpcenter.io

Swap helpcenter.io for your help center's domain. Help centers on a custom domain still load the loader script from helpcenter.io.

When the widget does not show up

Almost always it is the Site URL setting. The domain where you embedded the widget has to be listed in the widget's settings, or the widget stays hidden. Add the exact host, save, and reload. If you are testing locally, add localhost to the list.

That covers the full surface. For a hands-on version you can paste your own values into and click through, ask us for the Widget Integration Kit, two self-contained HTML pages that demonstrate every example above.