

Content Security Policy (CSP) directives and sources

Content updated: 10/31/2026

 This article is designated for administrators.

About

A Content Security Policy (CSP) uses directives to control which resources a page can load and how other sites can use the page. Most directives list sources, such as domains or special values.

Here are the directives and source values you might see in a policy for your Content Hubs or legacy Video Portal site, or for a Kaltura Application Framework (KAF) integration.

 To generate, test, and apply a policy, see [Configure a Content Security Policy \(CSP\)](#).

Directives

Your policy is made up of **directives**, such as `script-src` and `img-src`. Most directives control one type of content. Others control framing or base addresses.

The table below explains directives you might see in your policy or need to add. The generated policy doesn't necessarily include every directive your site needs. For example, it doesn't include `connect-src`.

Directive	What it controls
<code>default-src</code>	Provides a fallback rule for resource types, such as scripts, images, fonts, and connections, that don't have their own directive in the policy. It doesn't apply to directives that control other things, such as <code>base-uri</code> and <code>frame-ancestors</code> .
<code>script-src</code>	Controls which scripts the browser can run.
<code>style-src</code>	Controls stylesheets and inline styles.
<code>img-src</code>	Controls images, such as thumbnails and logos.
<code>font-src</code>	Controls fonts.

Directive	What it controls
<code>connect-src</code>	Controls browser connections to APIs, analytics services, and other endpoints.
<code>media-src</code>	Controls audio and video sources.
<code>worker-src</code>	Controls scripts that run in the background, separately from the page.
<code>frame-src</code>	Controls what your pages can load in frames, such as YouTube videos. Compare with <code>frame-ancestors</code> .
<code>frame-ancestors</code>	Controls which websites can load your pages in a frame. Compare with <code>frame-src</code> .
<code>base-uri</code>	Controls which addresses a page can use as its base address.
<code>object-src</code>	Controls plugins and other embedded objects. The generated policy sets this to <code>'none'</code> .

Sources

Each directive lists the sources that the browser can use. A source can be your own site, a specific domain, a group of subdomains, a URL scheme such as `blob:`, or a special value such as `'none'`.

The table below explains the sources you might see in your policy.

Value	Meaning
<code>'self'</code>	Allows resources from the same origin as the page, that is, the same scheme, domain, and port. It doesn't include other domains or subdomains, such as other Kaltura domains.
<code>'none'</code>	Prevents the resource type from loading from any source.
<code>https://example.com</code>	Allows resources from one specific HTTPS domain.
<code>https://*.example.com</code>	Allows resources from all subdomains of the specified domain, but not from the domain itself. For example, <code>https://*.example.com</code> doesn't include <code>https://example.com</code> .
<code>https:</code>	Allows resources from any domain over HTTPS.

Value	Meaning
<code>blob:</code>	Allows resources that the browser creates in memory, such as media data the player prepares during playback. This is a URL scheme, not a domain. Add it only to the directive that needs it.
<code>data:</code>	Allows resources that are included directly in the page, such as small images. This is a URL scheme, not a domain. Add it only to the directive that needs it.
<code>kaltura-pc:</code>	Allows your site to open the Kaltura Capture app. The generated policy includes this value in <code>frame-src</code> if Kaltura Capture is enabled on your site.
<code>'unsafe-inline'</code>	Allows inline scripts or styles, depending on the directive it's in, such as <code>script-src</code> or <code>style-src</code> . If the same directive includes a nonce, browsers ignore <code>'unsafe-inline'</code> . This value reduces the protection the policy gives. The generated policy includes this value if Theming isn't enabled on your site. Pages that don't use Theming need it, even on a site where Theming is enabled. Don't add or remove it unless Kaltura Support or your security team confirms that it's needed.
<code>'unsafe-eval'</code>	Allows dynamic code evaluation. This reduces the protection the policy gives. The generated policy includes this value if Theming isn't enabled on your site. Pages that don't use Theming need it, even on a site where Theming is enabled. Don't add or remove it unless Kaltura Support or your security team confirms that it's needed.
<code>'nonce-...'</code>	Allows inline scripts or styles that have the matching nonce. It doesn't allow other inline code, such as inline event handlers. The generated policy includes <code>'nonce-{{scriptNonce}}'</code> and <code>'nonce-{{styleNonce}}'</code> if Theming is enabled on your site. Each time a page loads, Kaltura replaces these tokens with a new one-time value. Keep them in your policy as they are. On pages that don't use Theming, such as the media page or the editor on some sites, nonce tokens can stop inline code from working. If that happens, contact Kaltura Support .



CSP applies to resources and connections made by the browser. If your site connects directly to an API from the browser, add the endpoint to the directive for that type of request. For API connections, this is usually `connect-src`. Server-to-server API calls aren't affected by CSP.
