

Troubleshoot Content Security Policy (CSP) messages in the browser Console

Content updated: 10/31/2026

 This article is designated for administrators.

About

When you test a Content Security Policy (CSP) for Content Hubs, legacy Video Portal, or a Kaltura Application Framework (KAF) integration in report-only mode, the browser **Console** shows a message for each resource your policy would block.

This article explains which messages to act on, how to read them, and how to add a missing source to your policy.



Before you start, add your policy to the `cspHeaderReportOnlyHeader` field, and set up the browser Console. For more information, see [Configure a Content Security Policy \(CSP\)](#).

Which messages to act on

The Console might show many messages. The examples in this section are from Google Chrome and Microsoft Edge. Other browsers word their messages differently.

The Console groups similar messages into one line, with a number next to it, for example, **53**. In a grouped message, each address is replaced with the placeholder `<URL>`. To see the actual addresses, select the number.

 **53** Applying inline style violates the following Content Security Policy directive 'style-src 'self' <URL> <URL> 'nonce-5403242498171404723''. Either the 'unsafe-inline' keyword, a hash ('sha256-47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU='), or a nonce ('nonce-...') is required to enable inline execution. The policy is report-only, so the violation has been logged but no further action has been taken.

Look at the start of each message:

- **The opening words are followed by a web address** (usually starting with `https://`), for example, *Loading the script 'https://www.youtube.com/iframe_api'* or *Connecting to 'https://...'*. In a grouped message, the address shows as `<URL>`. Add the domain to the directive named in the message, as described in **Add a missing source to the policy**, below.

```

▶ 53 Applying inline style violates the following Content Security Policy directive 'style-src 'self' <URL> <URL> 'nonce-7283736289662994193''. Either the 'unsafe-inline' keyword, a hash ('sha256-470E0pJ8H85a-/TIm+5JCeuQeRkmSNMp3JZG3hSuFU='), or a nonce ('nonce-...') is required to enable inline execution. The policy is report-only, so the violation has been logged but no further action has been taken.
Applying inline style violates the following Content Security Policy directive 'style-src 'self' https://*.kaltura.com https://fonts.googleapis.com 'nonce-7283736289662994193''. Either the 'unsafe-inline' keyword, a hash ('sha256-470E0pJ8H85a-/TIm+5JCeuQeRkmSNMp3JZG3hSuFU='), or a nonce ('nonce-...') is required to enable inline execution. The policy is report-only, so the violation has been logged but no further action has been taken.
▶ 53 Connecting to '<URL>' violates the following Content Security Policy directive: "default-src 'self' <URL>". Note that 'connect-src' was not explicitly set, so 'default-src' is used as a fallback. The policy is report-only, so the violation has been logged but no further action has been taken.
▶ 53 Applying inline style violates the following Content Security Policy directive 'style-src 'self' <URL> <URL> 'nonce-7283736289662994193''. Either the 'unsafe-inline' keyword, a hash ('sha256-470E0pJ8H85a-/TIm+5JCeuQeRkmSNMp3JZG3hSuFU='), or a nonce ('nonce-...') is required to enable inline execution. The policy is report-only, so the violation has been logged but no further action has been taken.
Applying inline style violates the following Content Security Policy directive 'style-src 'self' https://*.kaltura.com https://fonts.googleapis.com 'nonce-7283736289662994193''. Either the 'unsafe-inline' keyword, a hash ('sha256-470E0pJ8H85a-/TIm+5JCeuQeRkmSNMp3JZG3hSuFU='), or a nonce ('nonce-...') is required to enable inline execution. The policy is report-only, so the violation has been logged but no further action has been taken.

```

- **The opening words aren't followed by a web address**, for example, *Applying inline style violates...* or *Executing inline script violates...*. You can't fix these messages by adding a source. Don't add `'unsafe-inline'` or `'unsafe-eval'` to remove them. Contact [Kaltura Support](#) before you apply the policy.

Other web addresses later in a message, such as `https://*.kaltura.com`, are sources already in your policy. You can ignore them.

Ignore messages about addresses that start with `chrome-extension://`. They come from your browser extensions, not your site. To avoid them, you can test in a private (incognito) window, where extensions are usually turned off.

Read a CSP message

Here's an example of a message you might see:

```

Loading the script 'https://www.youtube.com/iframe_api' violates the following Content Security Policy directive: "script-src 'self' https://*.kaltura.com 'nonce-1475604276712415323'". Note that 'script-src-elem' was not explicitly set, so 'script-src' is used as a fallback. The policy is report-only, so the violation has been logged but no further action has been taken.

```

The message tells you:

- **What would be blocked** - The address at the start of the message. In this example, `https://www.youtube.com/iframe_api`.
- **Which directive to add it to** - The first word in quotes after the word *directive*. In this example, `script-src`. You can ignore the note about `script-src-elem`.
- **Whether it was blocked** - *The policy is report-only* means the resource was only reported. *The action has been blocked* means an applied policy blocked the resource. Most browsers also show these messages in red.

If a message starts with *Executing inline script*, it doesn't include an address to add. See [If you see messages about inline scripts or styles](#), below.



In the Console, nonce tokens appear as long random strings. Don't copy the policy from the Console. Always edit your own copy in the text editor.

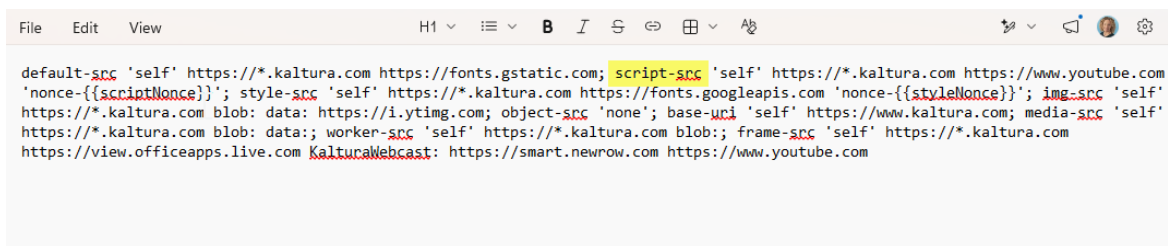


A reported resource isn't necessarily a security risk. Confirm that the resource is trusted and required before adding it to the policy. Don't automatically allow every domain reported in the Console. Ask your security or web development team to review any source you don't recognize.

Add a missing source to the policy

To allow a source, add it to the matching directive in your text editor copy of the policy. The steps below use the YouTube message above as an example.

1. In the Console, find the blocked address in the CSP message. It's the address in single quotes near the start of the message, right after *Loading the script* (or *Loading the image*, *Loading the stylesheet*, and so on). If the blocked address doesn't start with `https://`, for example, `blob:`, `data:`, or a custom scheme such as `kaltura-pc:`, don't guess what to add. Contact [Kaltura Support](#) or your security team. For example: *Loading the script 'https://www.youtube.com/iframe_api' violates the following...*
2. Shorten the address to just the domain. Keep everything up to the first single slash (`/`) after the domain name, including any port number, such as `:8443`, and delete the rest.
For example, `https://www.youtube.com/iframe_api` becomes `https://www.youtube.com`.
3. In the same message, find the directive name. It's the first word inside the double quotes after the word *directive*.
For example: *...violates the following Content Security Policy directive: "script-src 'self' https://*.kaltura.com..."*
4. Open your copy of the policy in your text editor.
The policy appears as one long block of text.
5. Find the directive name in the policy, for example, `script-src`. To find it quickly, press **Ctrl + F** and type the name.



```
default-src 'self' https://*.kaltura.com https://fonts.gstatic.com; script-src 'self' https://*.kaltura.com https://www.youtube.com
'nonce-{{scriptNonce}}'; style-src 'self' https://*.kaltura.com https://fonts.googleapis.com 'nonce-{{styleNonce}}'; img-src 'self'
https://*.kaltura.com blob: data: https://i.ytimg.com; object-src 'none'; base-uri 'self' https://www.kaltura.com; media-src 'self'
https://*.kaltura.com blob: data:; worker-src 'self' https://*.kaltura.com blob:; frame-src 'self' https://*.kaltura.com
https://view.officeapps.live.com KalturaWebcast: https://smart.newrow.com https://www.youtube.com
```

6. Place the cursor directly after the directive name, right after the last letter of `script-`
`src`.

7. Type a space, then the domain. For example, type a space, then

```
https://www.youtube.com
```

The domain is added to the directive.

Before:

```
script-src 'self' https://*.kaltura.com 'nonce-{{scriptNonce}}';
```

After:

```
script-src https://www.youtube.com 'self' https://*.kaltura.com 'nonce-{{scriptNonce}}';
```

8. Save the file.



Adding the domain, such as `https://www.youtube.com`, allows everything from that domain, not just one file. This is usually needed, because a resource often loads other files from the same domain, and a full address, such as `https://www.youtube.com/iframe_api`, stops working if it changes. Add only domains you trust. If your organization's security policy requires narrower sources, check with your security or web development team.

To allow all subdomains of a domain, use an asterisk, for example, `https://*.kaltura.com`. Some organizations' security policies don't allow these wildcards. In that case, list each domain separately. Check with your security team.

Test again after each change

1. Paste the updated policy into the **cspHeaderReportOnlyHeader** field, replacing the previous policy.
2. Select **Save**.
3. In your site's browser tab, clear the Console and refresh the page.
4. Test the feature again.

Fixing one message can reveal a new one. For example, a script you allowed might load another file from a different domain. This is normal. Repeat the process until no unexpected messages appear for any of the features you tested.

If you can't resolve a message, contact [Kaltura Support](#).

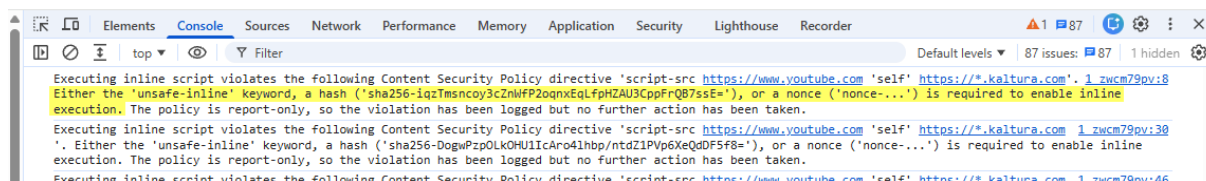
If you see messages about inline scripts or styles

A **script** is code that runs on a webpage. Some scripts are included directly in the page

instead of being loaded from a separate file. These are called inline scripts. They often come from custom JavaScript added to your site's header or footer.

The browser reports a message when an inline script isn't allowed by the `script-src` directive. Inline styles work the same way, with the `style-src` directive.

The browser might suggest allowing inline scripts, adding a script hash, or using a nonce. Before changing the policy, ask your security or web development team which option is appropriate.



If **Theming** is enabled on your site, the generated policy already includes nonce tokens. Even so, some pages, such as the media page or the editor, might not use Theming. On **non-Theming** pages, these messages can mean that something won't work correctly after you apply the policy.

If you see these messages when you test the generated policy without changes, contact [Kaltura Support](#) before you apply the policy.

When no unexpected messages appear, apply the policy. For more information, see **Step 3** in [Configure a Content Security Policy \(CSP\)](#).

Troubleshooting

Issue	What to do
The same message still appears after you add the source.	Check that you added the source to the directive named in the message, that the domain starts with <code>https://</code> and has no quotes, and that you saved the policy. Then clear the Console and refresh the page.
A new message appears for a different address after you add a source.	This is normal. The resource you allowed is loading other files. Add each required source, using the domain instead of the full address where possible.