Suppress Campaign Display Based on Page Elements or API Call

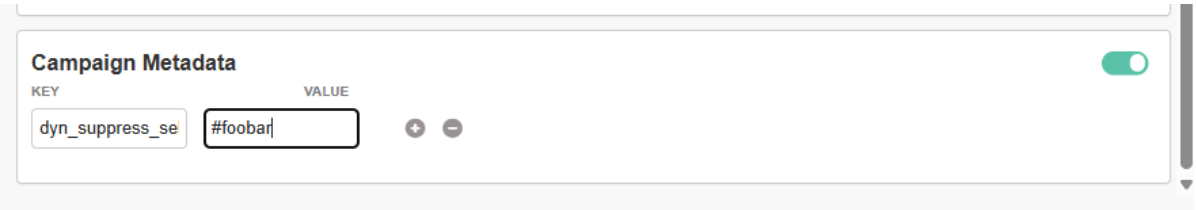
Last Modified on 08/17/2026 8:49 am EDT

If you need to suppress Digioh campaigns based on specific HTML elements being on the page, the activities of other JavaScript on your site, or the result of a real-time API call, you can do this with the Digioh Extension **Dynamic Suppression with Selector or Promise**. From the profile menu top right, click "Extension", then find and install this one.

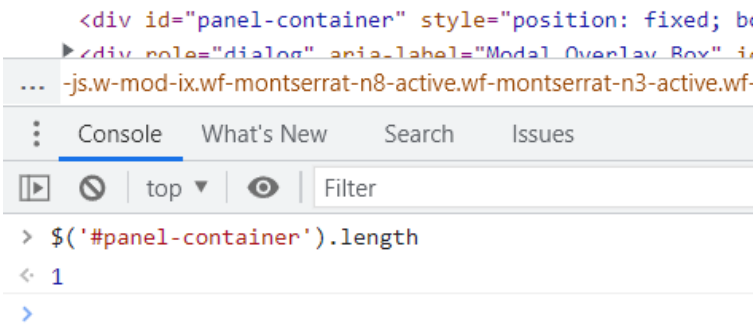
1. Page Element Use Case (Selectors)

Digioh Campaign Conditions evaluate at the time of page load (DOM Ready). You can use the HTML Exists Condition to target (or anti-target) Campaigns based on whether the page contains an HTML Element, using a jQuery Selector. However, this only works if the HTML in question is present *at the time of DOM Ready*. If you have a single page app (SPA), or JavaScript on your page that dynamically renders HTML, then you may encounter a race condition between your code and Digioh Conditions.

Use the metadata command **dyn_suppress_selector** with a value that is the relevant jQuery selector:



Be sure to test in devtools console that your selector finds the HTML, before configuring on your campaign.



Also, when viewed in boxqamode, the extension will notify you when it takes action to suppress.

New: Multiple-selector support

You can now specify **multiple** jQuery selectors in the same metadata value for `dyn_suppress_selector`. Separate selectors with a pipe character (`|`). If **any** of the selectors matches an element on the page, the box will be suppressed.

Example usage: dyn_suppress_selector : "#promo-banner | .user-logged-in | [data-hide-popup='true']"

In this example, if any element matching `#promo-banner`, or any element with class `.user-logged-in`, or any element with the attribute `data-hide-popup="true"` is present on the page, the Campaign will be suppressed.

Notes:

- Each selector is evaluated independently. Invalid selectors are logged (in QA mode) but do not break processing of the rest.
- This only applies to selector-based suppression — it does **not** affect the `dyn_suppress_promise` mode.
- Use this feature when you want to cover multiple possible page states under which you want to suppress a campaign

2. Asynchronous JavaScript and API Use Case

This is a more advanced use case that will require you to write a few lines of JavaScript on your site, or with Custom JS in your Digioh account. If you are not a developer and need help, let us know at support@digioh.com.

An example use case here might be if you have asynchronous JavaScript running on your site that must complete before knowing whether to show a campaign. Here, you would configure all the conditions on the campaign except for a condition related to that JavaScript. If your JavaScript was synchronous on page load, you could just use a "JavaScript Var" or "JavaScript Function" Condition, but because it is async it might not be available to Digioh Conditions when page load completes (a so called "race condition"). Instead, you can use this extension which will wait for your asynchronous JavaScript to complete before displaying (or suppressing) the Campaign.

Use the metadata command **dyn_suppress_promise** to name a function that returns a JavaScript Promise object. The Promise can implement any asynchronous business logic you need, but must resolve to Boolean true (suppress the campaign) or false (show the campaign).



The function must be declared in either the (top level) window object, or Digioh api or campaignapi objects, return a Promise, and must not expect any parameters. The syntax here is to identify a function, not a function call, but for ease of use parenthesis are optional. Examples:

- dyn_suppress_promise : window.getIsPremiumUserPromise

- dyn_suppress_promise : api.getIsPremiumUserPromise()
- dyn_suppress_promise : campaignapi.getShouldSuppressPromise

Here's an example Promise that uses the query string to determine whether to suppress the Campaign:

```
boxapi.fTestPromise = function() {
   return new Promise((resolve, reject) => {
      const suppress = api.getUrlParameter('suppress');
      if(suppress == 'true') resolve(true);
      else if(suppress == 'false') resolve(false);
      else reject('some error');
   });
};
```

While this is not really asynchronous, the concept for async operations is the same:

```
boxapi.fAsyncPromise = function() {
   return new Promise((resolve, reject) => {
      $.ajax({url: "https://customer.com/api"})
      .done(function(data) {
         if(data.suppress) resolve(true)
         else resolve(false);
  });
};
```

The extension will aggressively validate configuration and data types involved with the Promise. If testing on your live site, use boxqamode to see error and activity notifications.

Questions? Comments? Let us know at support@digioh.com