

EXHIBIT C

[Help](#)[Contact Us](#)[Community](#)

Cache-busting with legacy DFP tags

When you use a browser to view websites for the first time, the browser will save certain items, like images, to a specific folder on your hard drive. The next time you visit that website, the browser will check this folder first to see if it already has some of the items on the page. If so, it won't bother to download them again and instead just loads the files from the folder on your hard drive in order to make the page load faster for you. This process is called *caching*.

For example, if you've gone to Google.com before, your browser may save the Google logo in its cache. The next time you go to Google.com, the browser calls the image file from the cache, instead of calling the Google.com server for the logo image a second time and making you wait while it re-sends the same image you already have.

While browser caching can be helpful for an Internet user, it's problematic when it comes to ad serving. When a user loads a page with ad tags, the browser first checks if it has something in its cache to serve to the ad tag, such as a creative file that was delivered to that tag in the past. If the browser finds a creative in the cache that's served to that tag before, it will serve that creative.

This is a problem because it means that you could be serving old creatives to users, and most importantly, DFP won't count an impression for that creative because no call to the ad server was made when the browser retrieved a creative file from its cache. Browser caching can lead to impression counting discrepancies.

You can defeat browser caching and ensure that a fresh call is always made to the ad server by dynamically generating a part of the ad tags for each ad slot every time a web page is displayed. Doing so ensures that each time users navigate from web page to web page, a call is made to the ad servers rather than to the browser's cache.

We recommend using a dynamically generated random number attribute in the DFP ad tag (`ord=value`) to achieve the goal of forcing the browser not to use its cache. This enables the ad servers to count impressions each time a unique user requests a web page with ads, even if the user requests the same web page more than once.

This section discusses the following topics:

- [Using `ord=value`](#)
- [Example ad tag using `ord=value`](#)

Using `ord=value`

One of the reserved key-values for DFP tags is `ord=value`. Using this key-value and dynamically generating a random number for the value ensures that a fresh call is made to the ad server with each page load. The browser interprets each random number as a new request and consequently requests a new ad from the ad server. Each time the page is loaded, the random number changes, causing the browser to interpret it as a new request, so it skips checking the browser cache for a file to serve to the page.

The `ord=value` **must** be the last key-value in the ad tag and can only be followed by a question mark (?) before the tag is closed. Adding in the question mark at the end is an additional method for forcing the browser to make a fresh call rather than going to its cache.

Additionally, while the value generated for `ord=value` should be different each time the page is refreshed, the value should always remain consistent across different ad tags on the same page. For example, on the first page load, ad tag A and ad tag B will both have `ord=88372?` as the last key-value, while on the second page load when the page is refreshed, ad tag A and ad tag B will both have `ord=35200?` as the last key-value.

The value needs to remain consistent because the other function of `ord=value` is to identify that these ad calls are coming from the same page. If on the same page load, you have ad tags with different `ord` value numbers, the ad server will think it's serving ads to two entirely separate pages, which will interfere with features like roadblocks and competitive ad exclusion labels that depend on knowing which calls are coming from the same page.

There are a few different ways to populate `ord=value` with a random number every time your web page is loaded. DoubleClick recommends that you use the same technology to dynamically generate random numbers as you use to generate HTML pages on your website. For example, if your website is written in ASP, use ASP to generate random numbers for the `ord=value` key-value. Your web developer team should be able to advise you on the best solution for your website's setup.

For standard HTML page, you can use a JavaScript random number generator like this:

```
var ord = Number(ord) || Math.floor(Math.random()*10e12);
```

Since you only want to generate one new random number per page load, make sure you include this code just once on each page. It can be placed in the first ad tag on the page or somewhere in the source code with JavaScript tags around it before the ad tags on the page.

Please note that when you generate an ad tag using a tag generator, the tag may include a random number generator in the tag itself. Check your code so that you don't have multiple ad tags on the same page, each with its own random number generator, resulting in inconsistent numbers for `ord=value` across ad calls in the same page load.

Please also note that including non-integers in random number strings can cause back-end problems in DART. DoubleClick therefore recommends that your random numbers contain only integers. We recommend having at least 8 integers in the random number string.

Example ad tag using ord=value

Here's an example of an ad tag with `ord=value` properly implemented. The example below is of a JavaScript ad tag with a random number generator within the ad tag, but even with iframe and image ad tags, the random number generator and `ord=value` will be set up in a similar manner.

```
<script type="text/javascript">

var ord = Number(ord) || Math.floor(Math.random()*10e12);
document.write('<script type="text/javascript"
src="http://ad.doubleclick.net/Nnetwork_code/adj/first_level_ad_unit/second_level_ad_unit;
```

```
key=value;tile=tile_number;sz=widthxheight;ord=' + ord + '?'></script>');  
</script>  
<noscript>  
  <a  
href="http://ad.doubleclick.net/Nnetwork_code/jump/first_level_ad_unit/second_level_ad_unit;key=value;tile=tile_number;sz=widthxheight;ord=1234567890?" target="_blank">  
    
  </a>  
</noscript>
```

Build and serve legacy DFP tags

Generating ad tags

Examples of JavaScript, iframe, and image DART ad tags

Pass custom targeting criteria into legacy DART ad tags

Cache-busting with legacy DFP tags

Stub files and iframe busters

Using legacy DFP tags in another ad serving system

How helpful is this article:

Not at all helpful

Not very helpful

Somewhat helpful

Very helpful

Extremely helpful



©2013 Google Inc. | DoubleClick is a division of Google Inc.

[Privacy Policy](#)

[Terms of Service](#)

[Program Policies](#)

[DoubleClick Home](#)

English ▼