# Google Chrome 80 Changes and Cross-Site Request Forgery (CSRF) Concerns for Applications Deployed to Oracle Weblogic Server (Doc ID 2637652.1)

**In this Document**


Purpose
  Introduction to the SameSite Cookie Issue
Details
  Use HTTPS instead of HTTP while setting SameSite=None
  How to Test Your Applications With New Cookie Behavior
  How to Explicitly Set SameSite Cookie Attribute in Your Application
  Using a Proxy ( Oracle Traffic Director / Oracle HTTP Server / Apache )
References


## APPLIES TO:

Oracle WebLogic Server - Version 10.3.6 and later
Information in this document applies to any platform.
This document is for custom applications deployed to WebLogic Server. Applications supplied by Oracle may have coordinated advice provided by respective application/product teams.

## PURPOSE

This document outlines an issue that may affect applications deployed to Oracle WebLogic Server.

### Introduction to the SameSite Cookie Issue

The Google Chrome 80 release, scheduled for February 2020, changes the default cross-domain (SameSite) behavior of cookies to enhance security and privacy. Mozilla and Microsoft have also indicated an intent to implement the new model in Firefox and Edge in the future.

This Chrome Platform Status explains the intent of the SameSite attribute:

> "SameSite is a reasonably robust defense against some classes of cross-site request forgery (CSRF) attacks, but developers currently need to opt-into its protections by specifying a SameSite attribute. In other words, developers are vulnerable to CSRF attacks by default. This change would allow developers to be protected by default while allowing sites that require state in cross-site requests to opt-in to the status quo's less-secure model. In addition, forcing sites to opt-in to SameSite=None gives the user agent the ability to provide users more transparency and control over tracking."

With Chrome 80 release in February, by default Chrome will treat cookies that have no declared SameSite value as SameSite=Lax. Up until Chrome 80, the default is SameSite=None. After the Chrome 80 release, developers can still opt into the status quo of unrestricted use by explicitly setting "SameSite=None; Secure" cookies that will be available for external access.

Other browser vendor plans:

- Mozilla has affirmed their support of the new cookie classification model with their intent to implement the "SameSite=None; Secure" requirements for cross-site cookies in Firefox.
- Microsoft recently announced plans to begin implementing the model starting as an experiment in Microsoft Edge 80.

For more information, see this Chromium blog post.

Oracle recommends that application developers and application server administrators assess their situation to see if they will be impacted. Included in this document are some steps to consider.

## DETAILS

### Use HTTPS instead of HTTP while setting SameSite=None

If you manage cross-site cookies, cookies with "SameSite=None" must also specify "Secure", meaning they require a secure context. Besides using a filter to set SameSite attribute, an HTTPS connection is required to access your application. Refer to the WebLogic Server documentation for your version on how to configure HTTPS in WebLogic Server.

### How to Test Your Applications With New Cookie Behavior

This Chromium blog post explains how to test the effect of the new Chrome behavior on your site or cookies you manage by navigating to chrome://flags and enabling the "SameSite by default cookies" and "Cookies without SameSite must be secure" experiments. It is recommended that you use the latest version of Chrome to test your applications with the SameSite cookie setting. See https://www.chromium.org/updates/same-site/incompatible-clients/ for additional information about user agent checking.

### How to Explicitly Set SameSite Cookie Attribute in Your Application

It is recommended instead to update the application. You may need to use a Filter (javax.servlet.Filter) in your web application to explicitly set the value of "SameSite" attribute in the cookies. Refer to WebLogic Server documentation for how to develop and configure Filters in web applications:

Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server
14 Filters
https://docs.oracle.com/en/middleware/fusion-middleware/weblogic-server/12.2.1.4/wbapp/filters.html#GUID-A7441B4A-531A-41B8-844E-0A4A8DC4A6FF

**Note:** This is the only mechanism to manipulate headers (servlet filters) at the Weblogic Server level. It does not require rewriting the application when implementing a Filter.

Below is **sample code** for how to use a filter to set the SameSite attribute. This sample should be updated based on specific requirements and cookies used by the application:

**Download Example SameSiteFilter.java**

It requires knowledge of Java to update and compile the sample code for your application using the above documentation. WLS administrators are encouraged to consult with their application developers to implement this.

Note that explicitly setting "SameSite=None" may make an application less secure under some circumstances, and should be considered a workaround to support the new releases of Chrome and other browsers with the 'SameSite' behavior change. See https://www.chromium.org/updates/same-site/incompatible-clients/ for additional information about user agent checking. Application developers and administrators should read about the different 'SameSite' attributes at https://web.dev/samesite-cookies-explained/ in order to determine which of the settings of None, Lax or Strict is appropriate for the application's intentions.

**Modification History:** April 1, 2020: The original sample has been replaced with an improved sample to consider different versions and a focus on the specific cookie or cookies to help the reader further consider what is best for the application(s) experiencing issues as a result of the browser behavior change. Additional references included helping the reader determine the best 'SameSite' setting for the application.

### Using a Proxy ( Oracle Traffic Director / Oracle HTTP Server / Apache )

If the application cannot be updated, instead of the Filter on WLS, the SameSite attribute may be set at the proxy level. Example proxies include OTD or OHS/Apache.

Below are example configurations for how to use a filter to set **SameSite=None**. Explicitly setting "SameSite=none" may make an application less secure under some circumstances, and should be considered a workaround to support the new releases of Chrome and other browsers with the 'SameSite' behavior change. See https://www.chromium.org/updates/same-site/incompatible-clients/ for additional information about user agent checking. Application developers and administrators should read about the different 'SameSite' attributes at https://web.dev/samesite-cookies-explained/ in order to determine which of the settings of None, Lax or Strict is appropriate for the application's intentions.

If you manage cross-site cookies, cookies with "SameSite=None" must also specify "Secure", meaning they require a secure context. HTTPS should be configured on your proxy and only allow access to your application using https:// through the proxy. Testing with your application is required and further advice should be taken from the application developer to determine the best approach that will work for the application.

**Oracle HTTP Server / Apache**

A common method to set on the proxy level within the httpd.conf is to set the header in this way for all cookies:

```
Header edit Set-Cookie ^(.*)$ $1;SameSite=None;Secure
```

Or, for specific cookie such as JSESSIONID:

```
Header edit Set-Cookie ^(JSESSIONID.*)$ $1;SameSite=None;Secure
```

For more details on OHS, see Note 2635983.1 How to Set a SameSite Attribute for the Set-Cookie Header with Oracle HTTP Server

**Oracle Traffic Director**

A common method is to add the below line under the <Object name="default"> tag in appropriate <vs>-obj.conf / obj.conf file for all cookies

```
Output fn="sed-param-value" pblock="srvhdrs" name="set-cookie"
sed="s/HttpOnly/HttpOnly;Secure;SameSite=None/g"
```

Or, for specific cookie such as JSESSIONID:

```
<If defined defined $cookie{'JSESSIONID'}">
Output fn="sed-param-value" pblock="srvhdrs" name="set-cookie"
sed="s/HttpOnly/HttpOnly;Secure;SameSite=None/g"
</If>
```

For more details on OTD, see Note 2651304.1 How to Set SameSite Attribute in Response When Using Oracle Traffic Director

**References**

Chromium Blog: Developers: Get Ready for New SameSite=None; Secure Cookie Settings

web.dev: SameSite cookies explained

Chrome Platform Status: Cookies default to SameSite=Lax

Chrome Platform Status: Reject insecure SameSite=None cookies

## REFERENCES

Didn't find what you are looking for?