TrustArc

Cookie Consent Manager Professional

Implementation Guide

Revision History

Document Version	Date	Description
3.0	08-12-2025	Updated the Supported Browsers and Versions
3.1		

About This Document

This document provides easy to follow steps to embed the Cookie Consent Manager solution to your website.

Target Audience

This document is intended for implementation managers and/or web development teams.

Disclaimer

Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form by any means, electronic or mechanical, for any purpose, without the express written permission of TrustArc Inc. Moreover, this guide is strictly informational in nature and **is not intended to provide legal advice**, as all legal and compliance obligations and interpretations remain the responsibility of users in coordination with their legal counsel.

Table of Contents

Overview 8
Cookie Consent Manager Implementation 9
- Deploying the Script 9
  - Recommended Locations 10
  - Adding Parameter for pcookie 10
  - Adding Parameters for 'privacypolicylink' and 'cookieLink' 11
  - Adding Parameter for 'locale' 12
- Integrating Consent Manager with Tag Management System (TMS) 14
- Integrating Consent Manager with Wordpress 16

Consent Manager Custom CSS Reference 19
- Banner Customization 19
- Overlay Customization 23
- 2 Step Opt-In Overlay Customization 29
- UI Examples 32
  - Desktop 32
  - Tablet 35
  - Mobile 36

Consent Manager API Functions 38
- getConsent(...) 38
- getConsentDecision(...) 40
- getGDPRConsentDecision(...) 42
- getConsentCategories(...) 43
- getDefaultCategories 44
- setConsentLevel 45
- Event Handlers to Retrieve Notifications from Consent Manager 46
  - Method to Retrieve End User Level Consent 46
- Using TrustArc API from Third-Party iFrame 47
- How to Implement HandleMessage Function 48
- Zero Cookie Load Integration Using Consent Manager API 49
  - Verification and Troubleshooting 49
- Use Case Scenarios 51
- getConsentDecision Results Table 52
- getGDPRConsentDecision Results Table 53
- setConsentLevels Results Table 54

Google Tag Manager 56
- Creating Custom Variable and Triggers in GTM 56

Deploying a Zero Cookie/Tracker Load Integration 58
Zero Cookie Load with Expressed Consent Configuration 59
Adding TrustArc Integration Assets 61
Firing Rules 63
Preference Blocking Triggers 63
Event-based Firing Rules 64
Create GTM Custom Events 64
Apply Custom Events to Your Tags 65
Add Custom Javascript tag to Trigger Events 65
Using Event and Cookie-based Triggers Together 68
Verification and Troubleshooting 69
GTM Preview Mode 69
Summary 69
Verifying a GTM Zero-Tracker Integration 71
Troubleshooting 74
Adobe Dynamic Tag Manager 76
Adding the Cookie Consent Script 76
Adding Page Load Script (Optional) 81
Applying New Condition to All Rules/Tags 82
Zero Cookie Load (Expressed Consent) 82
Regular Load (Implied Consent) 84
Adobe Experience Platform 87
Adding the Cookie Consent Manager Extension 87
Applying Rules to the TrustArc Cookie Consent Manager 90
Adding a Page Reload Script 94
TrustArc Events and Conditions 98
Data Elements 98
Setting up a Functional Rule 106
Setting up an Advertising Rule 108
Implementing ECID Opt-in Services 110
Pre-requisites 110
Setup 112
Using Custom Events for Firing Tags Dynamically 119
Working with Adobe Event Listener 119
Adding a Dynamic Event Generation for Datalayer 120
Event Listener for Dynamic Loading Based on Consent 124
Custom Event for Functional Tags 124
Custom Event for Advertising Tags 125
Tealium Tag Manager 127

Consent Behaviors 127
Deployment Strategies 128
User Flow 128
Consent Manager Assets 130
Consent Manager Code 130
Tealium Event Listener 131
Using Tealium to Deploy CM Assets 133
Tealium Data Sources 136
Tracker Consent 136
Tracker Behavior 137
Tealium Load Rules 138
Expressed Consent 138
Implied Consent 139
Verification and Troubleshooting 140
Verifying a Zero-Tracker Integration 140
Troubleshooting 141
Auto-block Feature 142
Introduction 142
User Flow 143
Auto-block Implementation 144
Setting Up Auto-block Script Tag 144
Manual Block Implementation 150
Setting Up Manual Blocking of Script Tag 150
Manually Blocking a Script Tag with a Domain 151
Ignore Tags for Autoblock 151
Validation 152
Verifying Auto-block Works 152
Page Reload to Block Tags 156
Limitations 157
EasyView 159
Overview 159
Settings 160
Language 161
Country Code 161
CM ID 161
Stealth Auto-Executing Link 162
Appendix 163
Supported Browsers and Versions 163
Visitors to your website 163

Overview

TrustArc's Cookie Consent Manager ("Consent Manager" or "CM") is served via a few lines of JavaScript ("JS") embedded in the HTML of any webpage where a CM user chooses to require end users' consent. While most choose to implement the JS in the footer, it can be placed anywhere within the HTML.

Depending on which version of CM you have purchased or are using, the CM can be integrated with any **Tag Management System** such as Google Tag Manager, Tealium, or Adobe DTM to control the firing of tags based on the level of consent provided by the end-user. Similarly, a Consent Manager API can be used to determine whether a particular vendor domain is approved or denied, which in turn can be used to control the firing of tags that sit outside a Tag Management System. Finally, instead of a tag manager or API integration, there is also the option of integrating with and firing a vendor's opt-out mechanism. Each is discussed further in this guide.

NOTE: Because of the nature of the vendor opt-outs method, using that integration – in contrast to a tag manager or the Consent Manager API – does not allow for cookies or tracker scripts to be suppressed when a new end user accesses a website. Instead, that method allows for opting out of cookies that were initially loaded onto the end-user's browser.

Cookie Consent Manager Implementation

This section covers the step by step process to implement the Cookie Consent Manager (“Consent Manager” or “CM”) solution to your website. Follow the sequence accordingly.

Deploying the Script

Embed TrustArc script within an HTML element on all pages needed.

None

```
<div id="consent-banner"></div>
<div id="teconsent">
<script type="text/javascript" async="async"
src="//consent.trustarc.com/v2/notice/{cmId}"></script>
</div>
```

NOTE: Please ensure that your system has the `'referer'` header enabled; otherwise, the TrustArc system will not recognize the request and mark it as invalid. You can request this information from your IT team if you are unsure of your company network setup.

Breakdown of the Consent Manager Script Elements:

Script	Description
`"consent-banner"`	Controls the banner placement on your website. This `div` element should be placed within the page element you’d like the banner to display.
`"teconsent"`	Controls the “Cookie Preferences” link that a user can click on to load the Consent Manager at any point they would like to set their preferences.
`script` tag	Controls the calls to Consent Manager and can be loaded anywhere.
`{cmId}`	The unique value of your Consent Manager and should be replaced by the script you produce in Step 5 of the Cookie Consent Creation Wizard.

Important: For security purposes, the Trustarc CCM script is designed to only function on the URLs that you indicated in Step 1 of the Cookie Consent Creation Wizard. This helps ensure that your script is not

pulled from your site and then used by another third-party website operator.

NOTE:

- Please refer to the *Appendix* section for details on the latest support matrix for Consent Manager.
- If you need to append the script within HTML block elements, you may switch out the `<div>` element to `<span>` to ensure the tag is appended correctly to the page. Using `<span>` instead of `<div>` is acceptable as well.
- If you would like to implement the banner on the website in a floating permanent location, please add the CSS styling elements to the **consent-banner** div element, for example:

```
<div id="consent-banner" style="position:fixed; bottom:0px; z-index:999999;"></div>
```

Recommended Locations

TrustArc recommends deploying the Consent Manager script in a location that will be visible in 99% of your site. This is to ensure that the Consent Manager script is invoked regardless of how a user accesses your site.

Common locations for such visibility are:

- **Footer**
- **Header**
- **Floating bottom left or right locations on the website**

Adding Parameter for pcookie

Users can add the "pcookie" parameter to indicate which cookies are to be scoped to the parent domain (when a provisioned domain is not the parent domain of the current site). When the "pcookie" is added, TrustArc's cookies will be dropped within the parent domain instead of the root domain. The TrustArc first party cookies are used to store the user's consent choice into their web browser.

Sample script to enable this feature:

```javascript
<div id="consent-banner"></div>
<div id="teconsent">
  <script type="text/javascript" async="async"
src="https://consent.trustarc.com/v2/notice/2auwki?pcookie"></script>
</div>
```

The “pcookie” parameter simplifies consent management across the website. For example, support.trustarc.com is a subdomain of trustarc.com. With “pcookie,” users do not need to repeatedly give their consent on each domain. Instead, their consent will apply to both support.trustarc.com and trustarc.com.

Adding Parameters for 'privacypolicylink' and 'cookieLink'

If you are deploying the TrustArc Cookie Consent Manager to multiple websites that have different Privacy Policy or Cookie Policy Links and you would like these links to show in the Consent Manager Banner or modal window, then you can pass these links into the Javascript snippet using the “privacypolicylink” and “cookieLink” parameters. To use these parameters, you first need to enable the links in the *Design Consent Manager* step within the configuration portal.

Sample Script:

```javascript
<div id="consent-banner"></div>
<div id="teconsent">
  <script type="text/javascript" async="async"
src="https://consent.trustarc.com/v2/notice/n4anom?cookieLink=https://www.tru
starc.com/cookie-policy&privacypolicylink=https://www.trustarc.com/privacy-po
licy"></script>
</div>
```

Where the consent manager is loaded on different brand sites that have different Privacy or Cookie Policies, users can inject their policy via the JavaScript for the page being visited so it is then updated in the *Cookie Banner* and *Overlay* window.

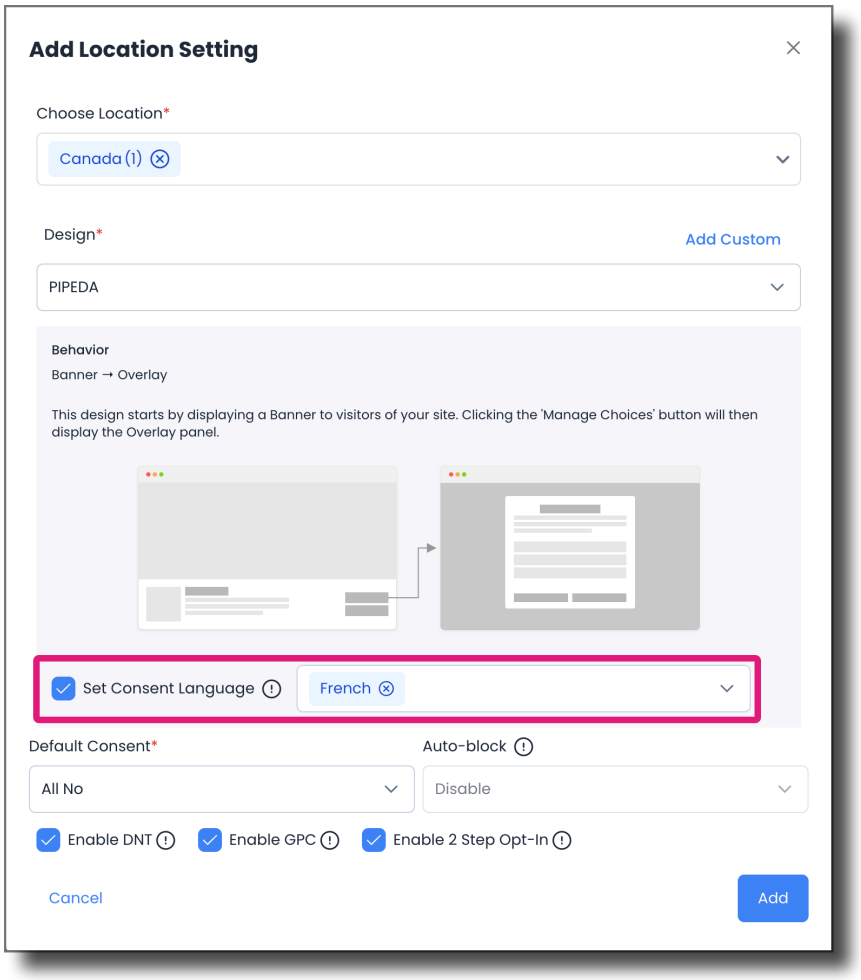
Adding Parameter for 'locale'

Users can override the browser language and translations on each website by using the “&locale=xx” parameter within the JavaScript. This parameter forces the Consent Manager to show in a specific language. This can be useful if certain pages are localized to a specific language and you want the Consent Manager to match that language. When not used, dynamic language detection is done using the user’s browser language choice and the Consent Manager is then localized to that language. Use of the locale parameter of dynamic language detection requires the languages to be configured in the *Configure Consent Manager* step in the TrustArc self-service portal. In the TrustArc portal it is also possible to force a language based on the **Set Consent Language** location setting in the *Configure Consent Manager* step.

This is the established order of priority based on the available settings:

- if **Set Consent Language:** *not enabled*
    a. `&locale` parameter
    b. browser language setting
- if **Set Consent Language:** *enabled*
    a. `&locale` parameter
    b. language override using the "Set Language" option
    c. browser language setting

The `&locale` parameter in the Javascript takes precedence over **Set Consent Language** and browser language settings.



Sample Script

JavaScript

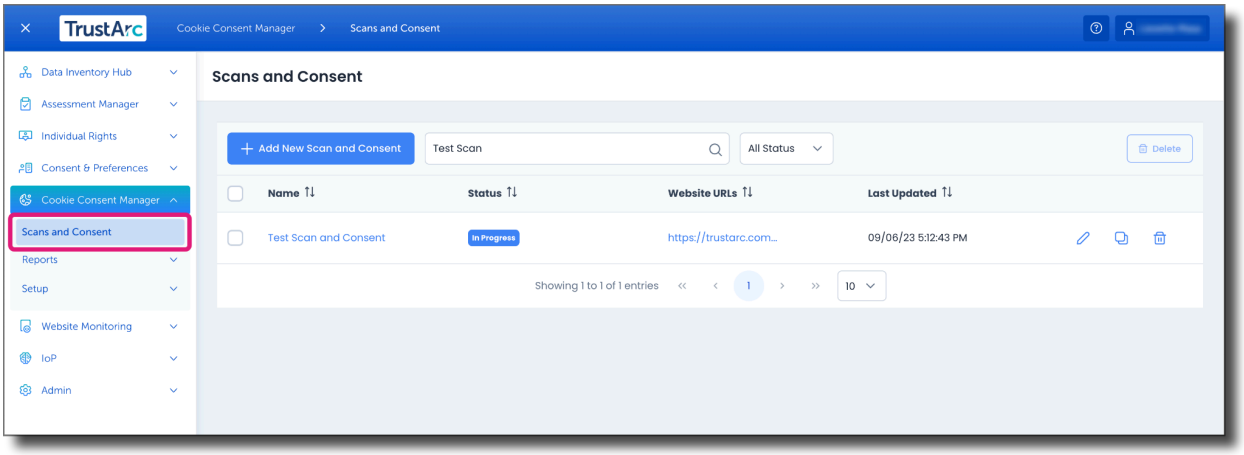
```
<div id="consent-banner"></div>

<div id="teconsent">

 <script type="text/javascript" async="async"
src="https://consent.trustarc.com/v2/notice/ukjtyq&locale=fr"></script>

</div>
```

Integrating Consent Manager with Tag Management System (TMS)

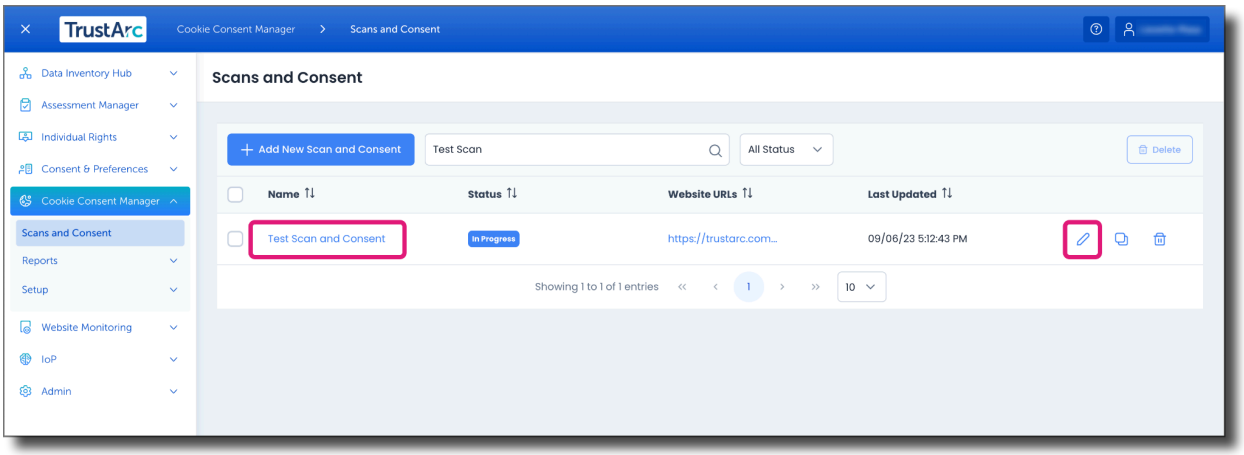
The Consent Manager can be integrated with a **Tag Management System** such as Google Tag Manager, Tealium, or Adobe DTM to control the firing of tags based on the level of consent provided by the end-user. Note that if your website does not use a tag manager, the CM will default to using 3rd party vendor opt-out cookies (where such opt-out cookies are available for a vendor/service provider). Note that, in contrast to integrating the CM with a tag manager, because of the nature of the 3rd party vendor opt-out method, cookies and tracker scripts will not be suppressed upon site load, which could pose regulatory risk in the EU for consideration with your legal counsel.

To enable the **TMS integration** feature, follow these steps:

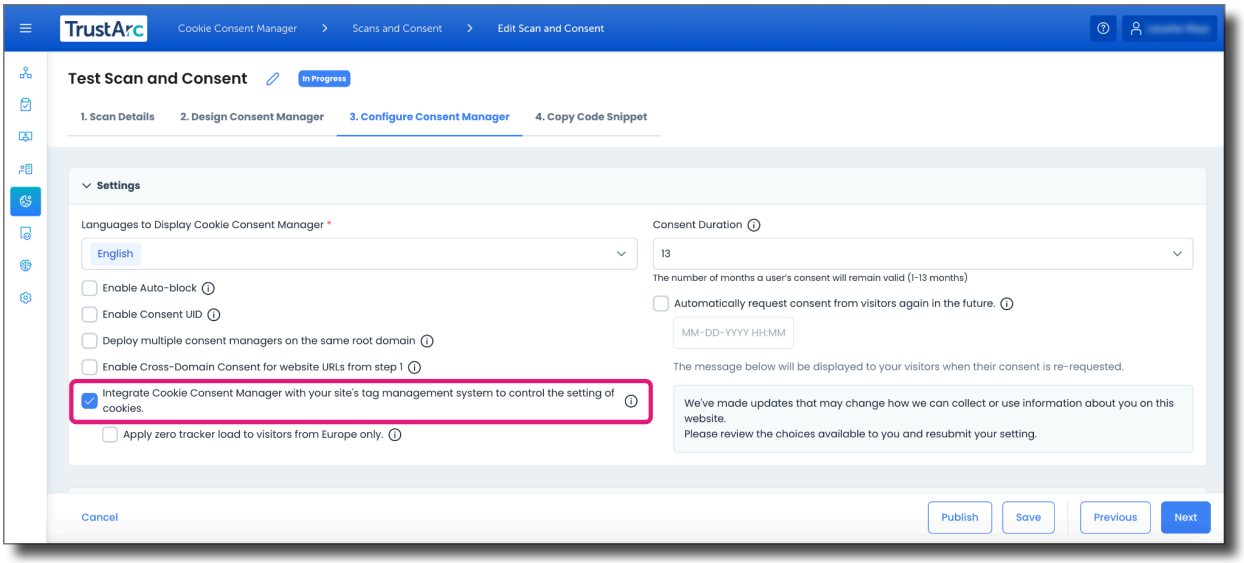
1. From the Cookie Consent Manager platform navigation panel, select **Scans and Consent.**



2. Select a cookie consent manager to update or click the **Edit** icon.



3. In the *Configure Consent Manager* page, select the **Integrate Cookie Consent Manager with your site's tag management system to control the setting of cookies** checkbox.



Selecting the **Apply zero tracker load to visitors from Europe only** checkbox enables the `notice_behavior` cookie. This enables you to isolate the user to an EU location where non-essential tags are not immediately fired on first page load if the end-user is in the EU region and has not yet consented to some or all non-essential trackers.

Disclaimer: Plugins and browser extensions can block your tag managers and then, block CCM. Please check if your tag manager works with the popular Adblockers. If your tag manager is blocked, you may consider firing the TrustArc scripts outside of the tag manager in the HEAD tag.

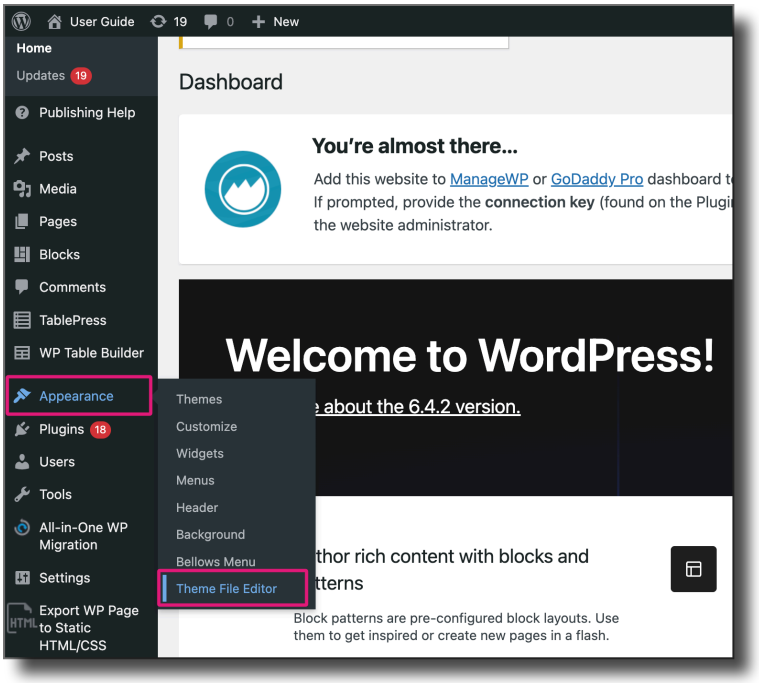
Integrating Consent Manager with Wordpress

The Consent Manager can be implemented in **Wordpress**. There are some additional steps to consider when implementing the Consent Manager in Wordpress as this requires specific placement of the HTML elements via Wordpress Theme Editor.

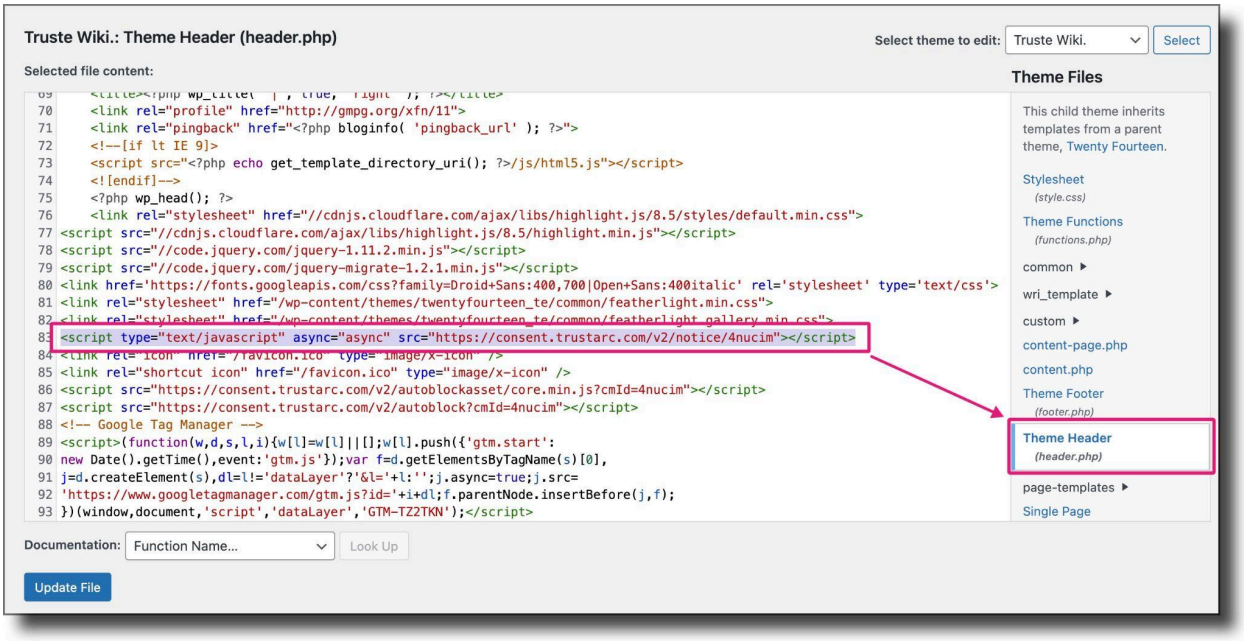
Before you proceed, ensure that you are logged into the WordPress admin section. The changes may vary depending on your theme or the plugin used to control your CSS. However, the steps will be similar.

To implement the Consent Manager in Wordpress, follow the steps outlined below:

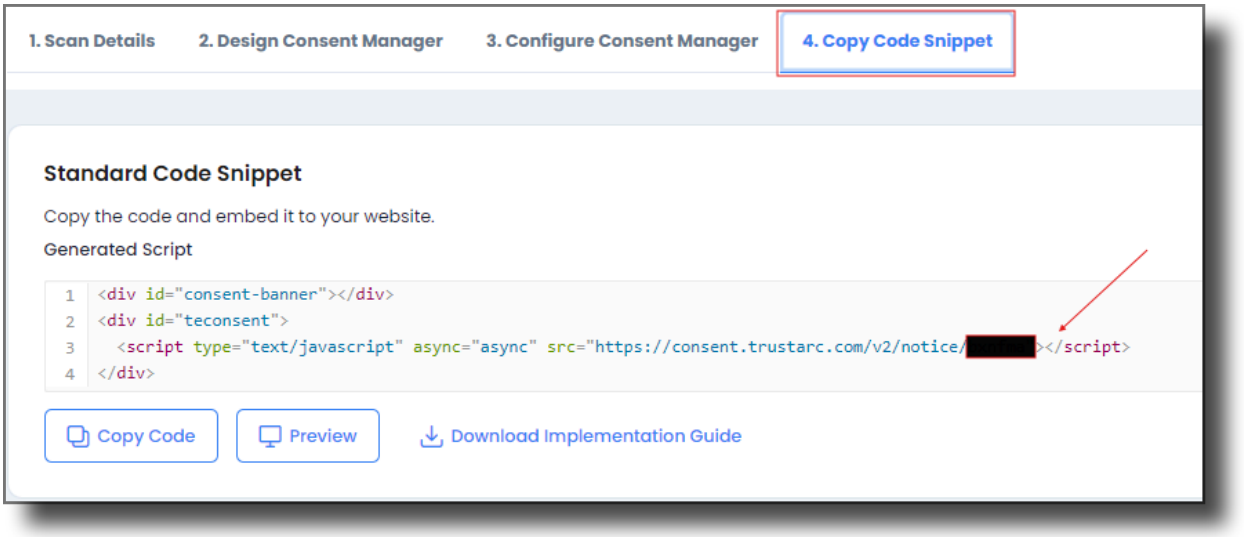
1. From the Wordpress navigation panel, select **Appearance > Theme File Editor**.



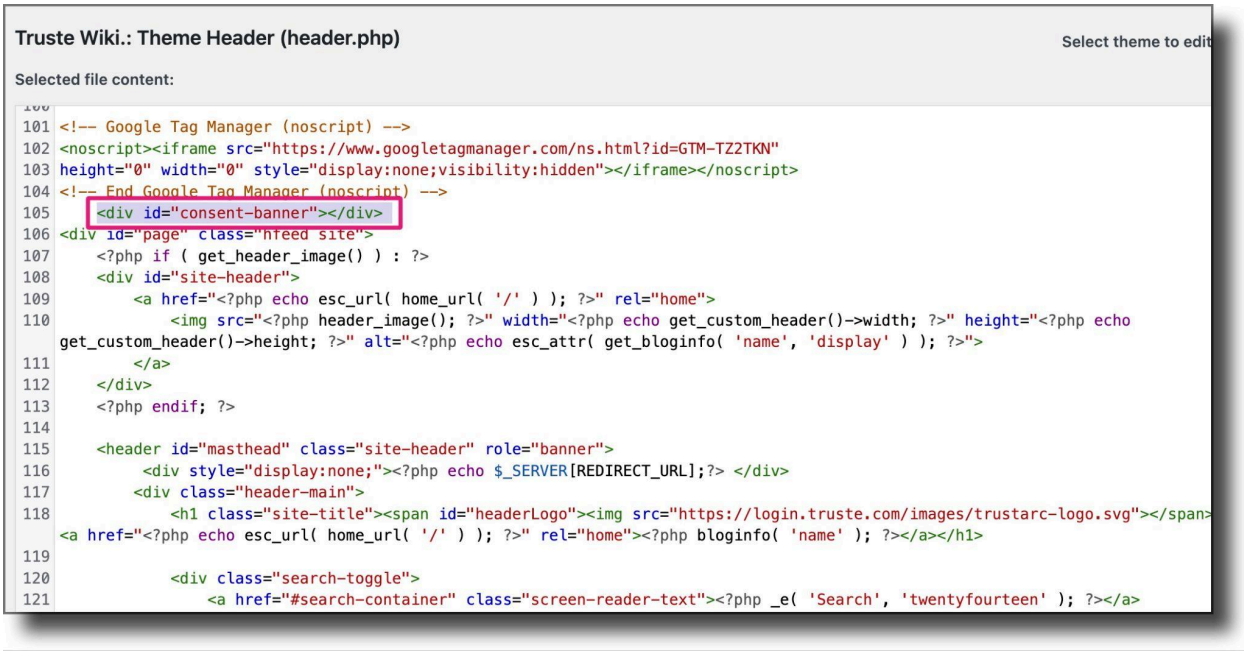
2. This file may have a different name depending on the theme you are using. Make sure to add the script tag in the head tag. Select the **header.php** file to include the script tag in the **head tag**. A parameter needs to be updated to match your Cookie Consent Manager instance. Contact your Technical Account Manager or Support team for the details.



3. Copy your code snippet from **Step 4** under your instance in the CCM portal.



4. Nest the div with ID "**consent-banner**" within the **body tag** to load the cookie consent banner on the website. The script placement must be in the **header tag**. The image below references the placement of these HTML elements.



5. Place the **teconsent** div in the footer tag. Use the tag to align the *Cookie Preferences link* alongside any other hyperlinks present in your footer.

6. Click **Update File** to save your changes.

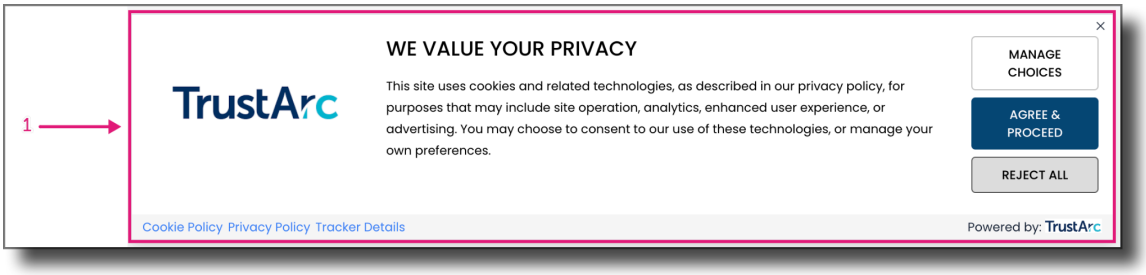
Consent Manager Custom CSS Reference

This section presents the list of classes and ids of the different `<div>` tags in customizing the banner and overlay of a CM instance.

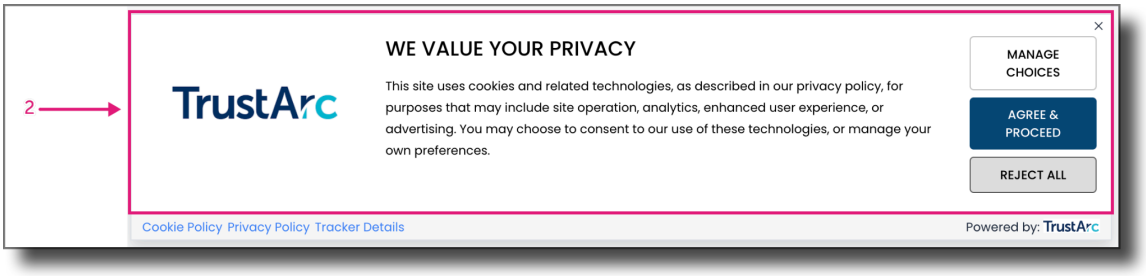
Banner Customization

This section focuses on the different class and id attributes in each `<div>` tags.

1. **#truste-consent-track**
    - This serves as the container for all banner contents and elements.
    - Banner's dimension, position, font properties, and background color can be overridden.



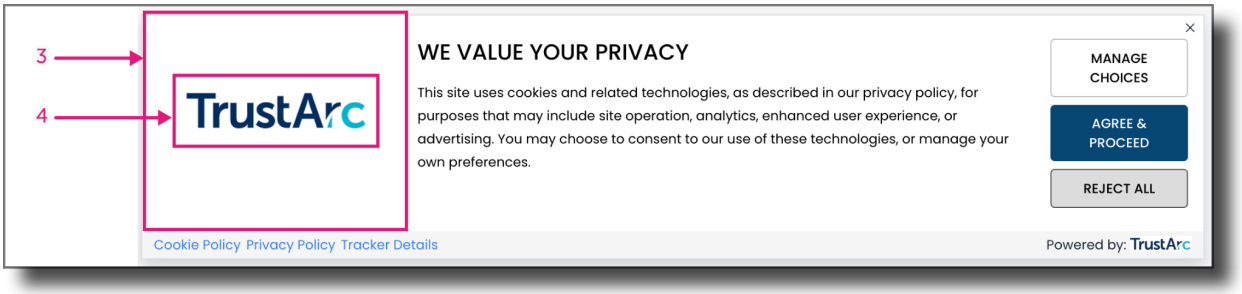
2. **.trustarc-banner-content, #truste-consent-content**
    - This holds the client's logo, header text, body text, consent buttons, and the close button.



3. **.trustarc-client-logo**
    - This holds the logo container for the client's logo.
    - This section can be hidden.
    - Client's logo position can be overridden.

4. **.trustarc-logo-container**
    - This contains the client's logo.

- Logo's width, height, and background color can be overridden.



5. **.trustarc-banner-details**

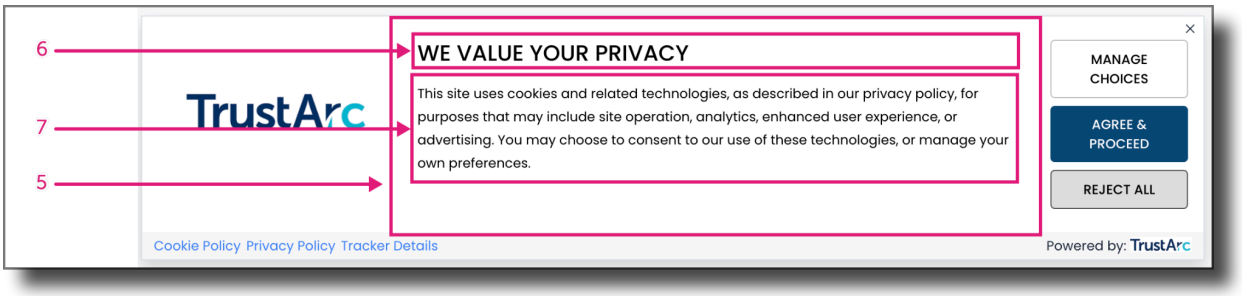
   - This serves as a container for all banner content
   - Text container's width, height, and position can be overridden.

6. **.trustarc-header-text**

   - This holds the banner header text.
   - This section can be hidden.
   - Font properties can be overridden.

7. **.trustarc-body-text**

   - This holds the banner's consent text.
   - Font properties can be overridden.

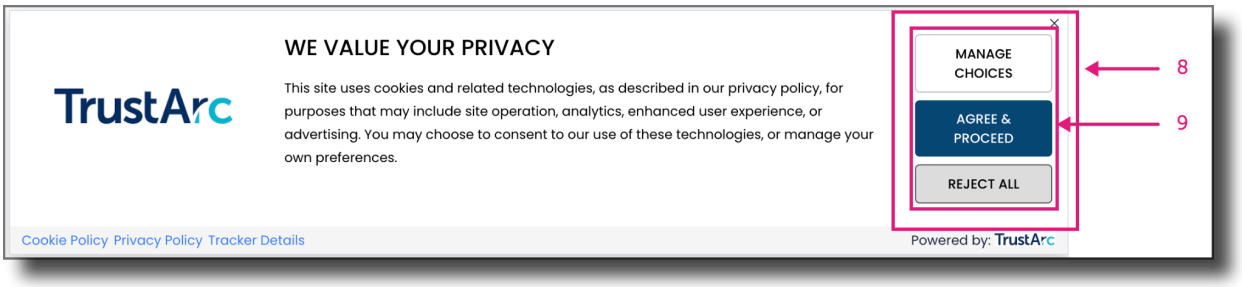


8. **.trustarc-banner-actions**

   - This serves as a container for all consent buttons.
   - Padding can be overridden

9. **#truste-consent-buttons**

   - This encloses the consent buttons.
   - Buttons' width, flex-direction, and order of the buttons can be overridden.



10. **.trustarc-secondary-btn**

- This button launches the consent overlay.
- Font properties can be overridden.

11. **.trustarc-acceptall-btn**

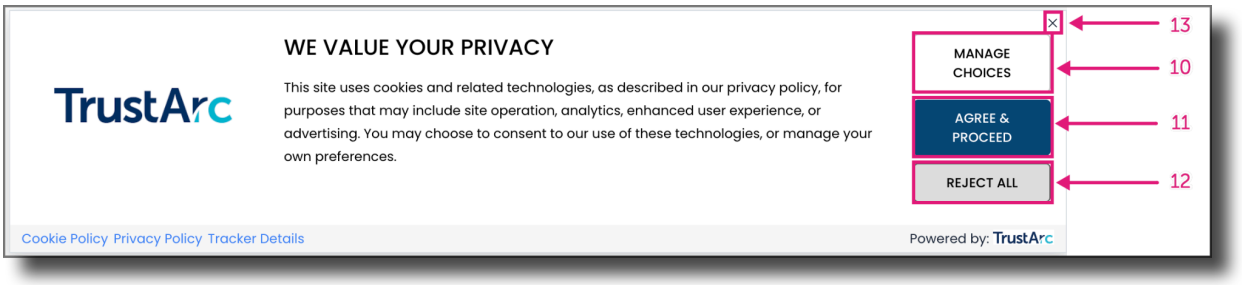
- This button allows users to accept consent to all consent categories.
- Font properties can be overridden.

12. **.trustarc-reject-btn**

- This button allows users to decline consent to all optional consent categories.
- Font properties can be overridden.

13. **.trustarc-banner-close**

- The position of the icon can be modified based on the client's preference.
- This section can be hidden.



14. **.trustarc-banner-footer**

- This encapsulates all banner elements.
- Footer's background color can be overridden.
- This section can be hidden.

15. **.trustarc-banner-left**

- This holds the footer links.
- Font properties and its position can be overridden.

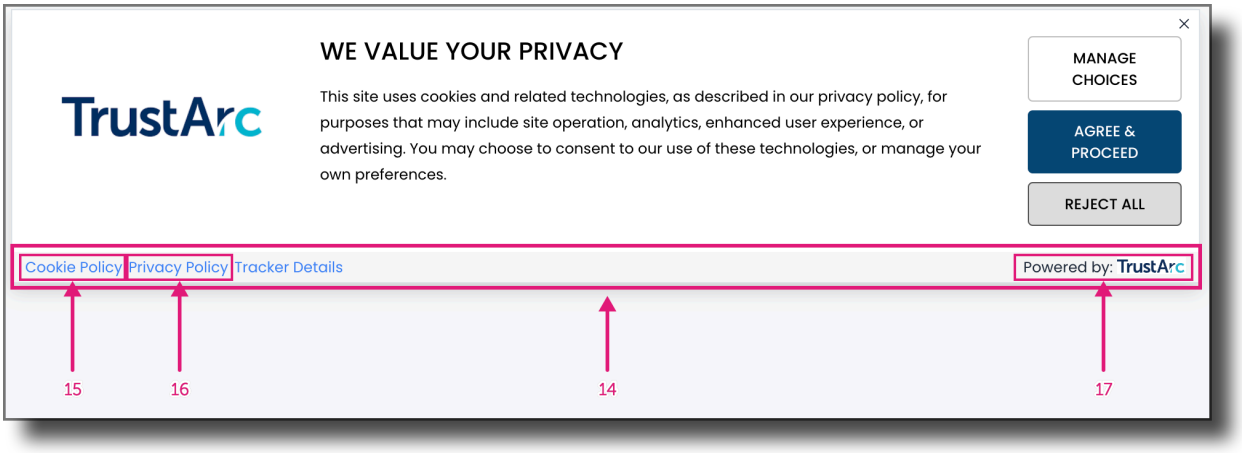
- This section can be hidden.

16. #truste-privacy-button

- This holds the privacy policy link.
- Font properties and its position can be overridden.
- This section can be hidden.

17. .trustarc-banner-right

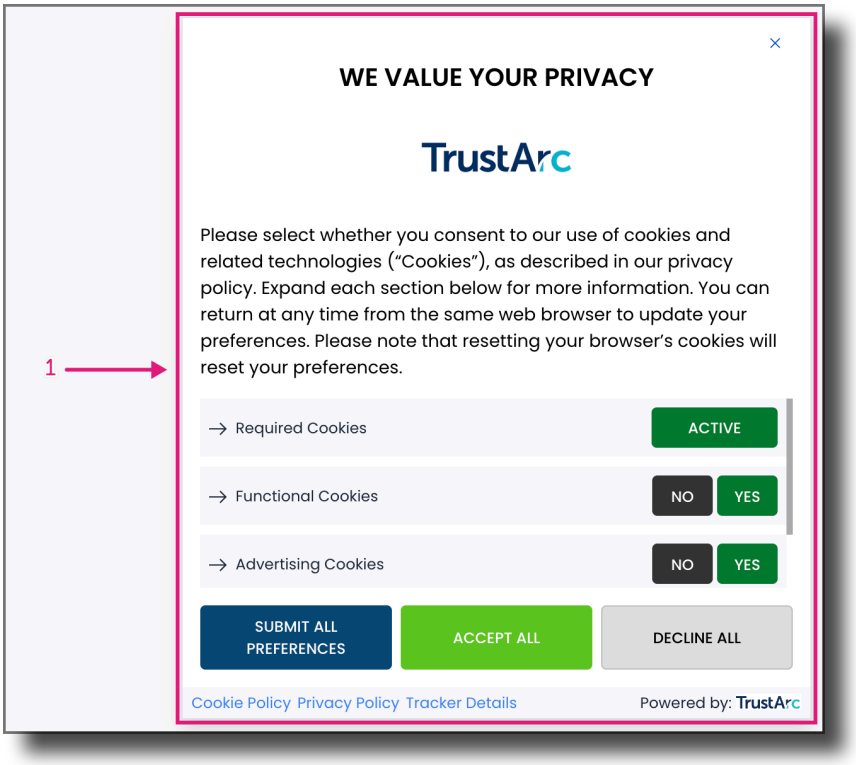
- This holds an image link that redirects the users to the TrustArc website.
- This section can be hidden.



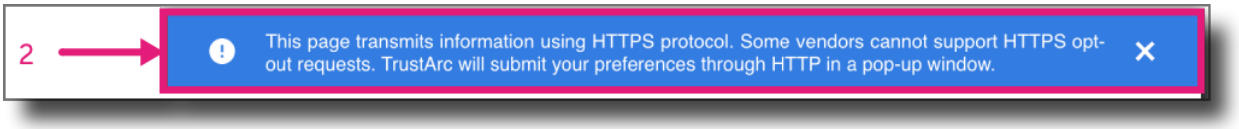
Overlay Customization

This section focuses on the different class and id attributes in each `<div>` tags.

1. **.trustarc-background**
    - This serves as the container for all overlay contents and elements.



2. **.warningMessages**
    - This holds the warning message.
    - Font properties can be overridden.
    - This section can be hidden.



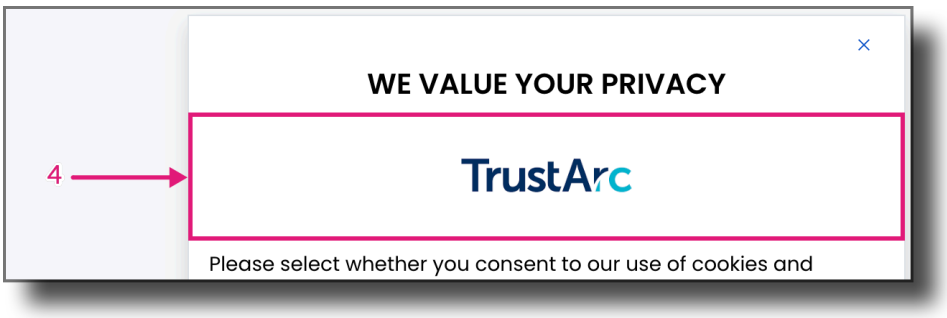
3. **.trustarc-header-text**

   This holds the overlay consent text.

NOTE: If you wish to customize separately the headers of both pages, you may use **.slogan-text.trustarc-header-text** for the overlay's landing page and **span.trustarc-header-text** for the cookie details page.

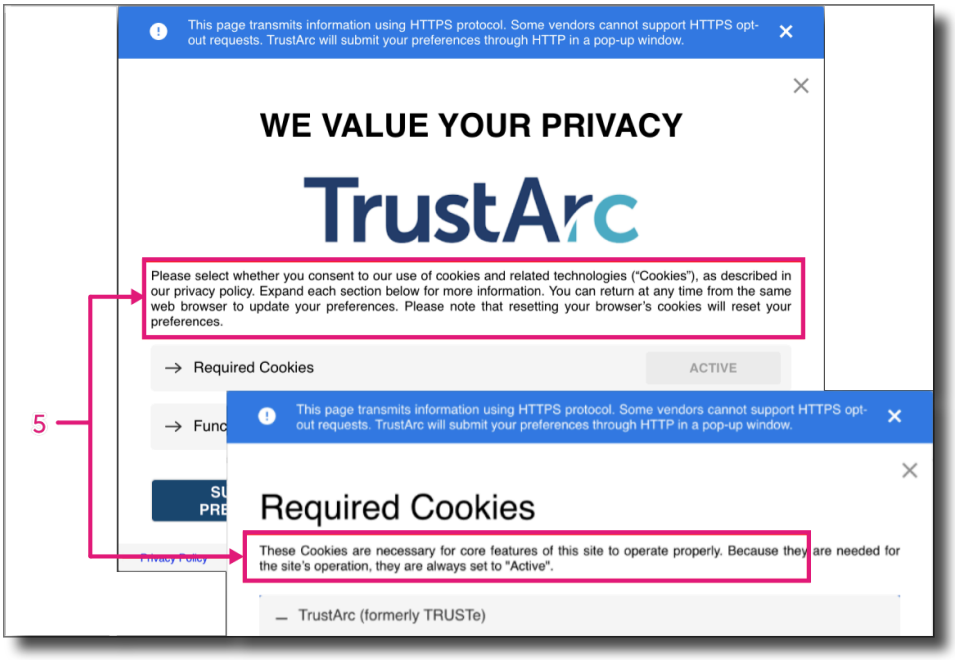
4. **.client-logo**

   - This holds the client's logo.
   - This section can be hidden.
   - Logo's width and margin can be overridden.



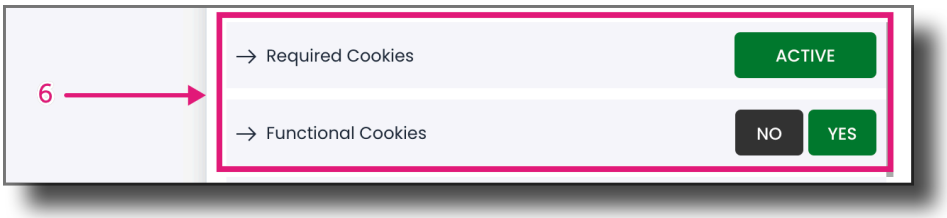
5. **.trustarc-body-text**

- This holds the overlay consent text.
- Font properties can be overridden.



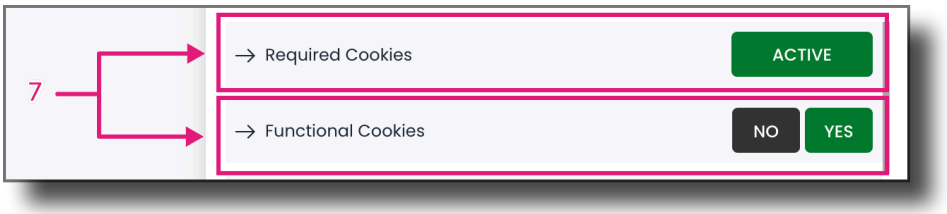
6. **.buckets-list**

- This holds the category bucket listing.
- Panel's background color and font properties can be overridden.



7. **.bucket-list-item**

- This holds the category buckets individually.
- Font properties and panel's background color can be overridden.

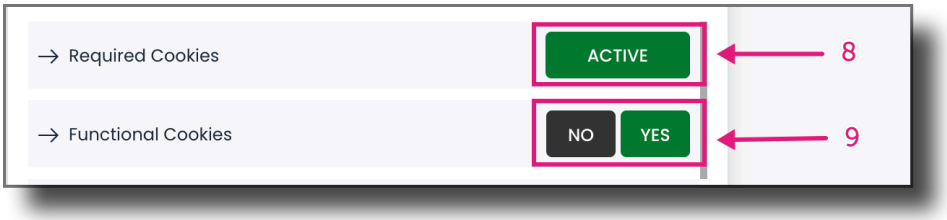


8. **.active-wrapper**

- This holds the ACTIVE button.
- Font properties and button's color can be overridden.

9. **.stv-radio-buttons-wrapper .stv-radio-button+label**

- This encloses the toggle switch.
- Font properties can be overridden.



10. **.trustarc-submit-buttons-container**

- This serves as a container for all consent buttons.
- Flex-direction of buttons can be overridden.

11. **.trustarc-primary-btn**

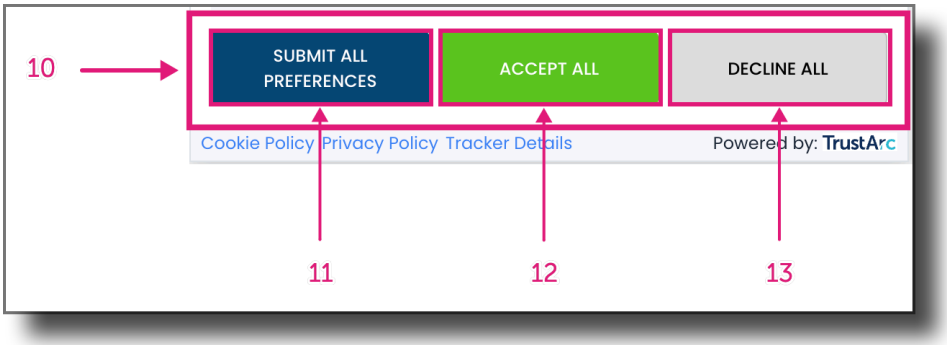
- This enables users to submit their custom cookie preferences.
- Font properties can be overridden.

12. **.trustarc-acceptall-btn**

- This button allows users to accept consent to all consent categories.
- Font properties can be overridden.

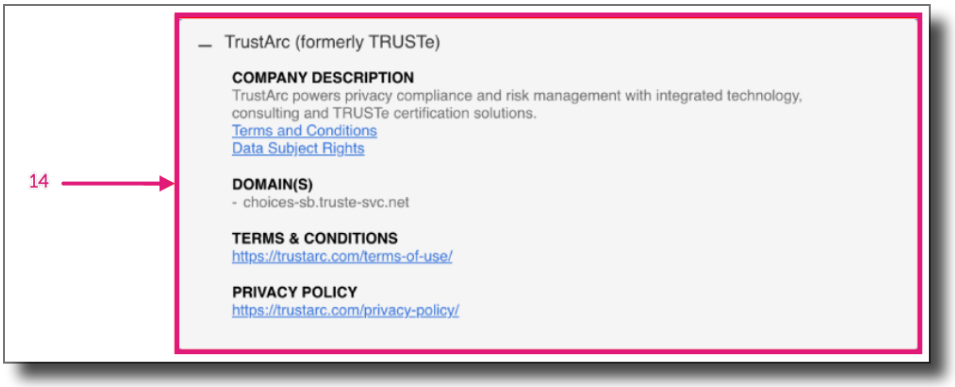
13. **trustarc-declineall-btn**

- This button allows users to decline consent to all optional consent categories.
- Font properties can be overridden.



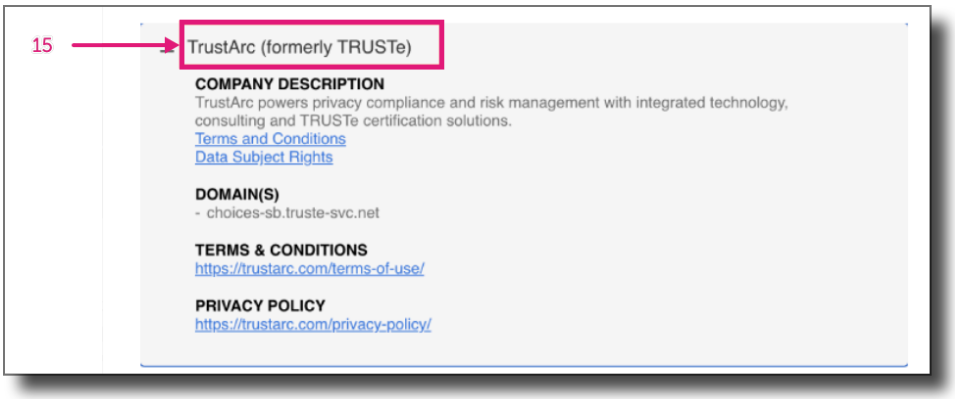
14. **.accordion-item**

- This contains all vendor and tracker details.
- Background color and font properties can be overridden.



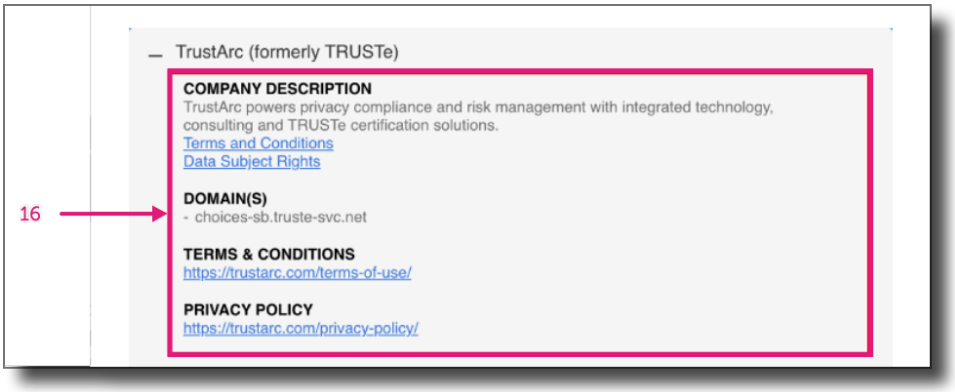
15. **.accordion-header**

- Font properties can be overridden.



16. **.accordion-content**

- This contains all vendor and tracker details.
- Font properties can be overridden.



17. **.trustarc-back-btn**

- This button allows users to navigate the previous modal view.
- Font properties can be overridden

18. **.ta-footer-bottom**

- This encapsulates all overlay elements.
- Background color can be overridden.
- This section can be hidden.

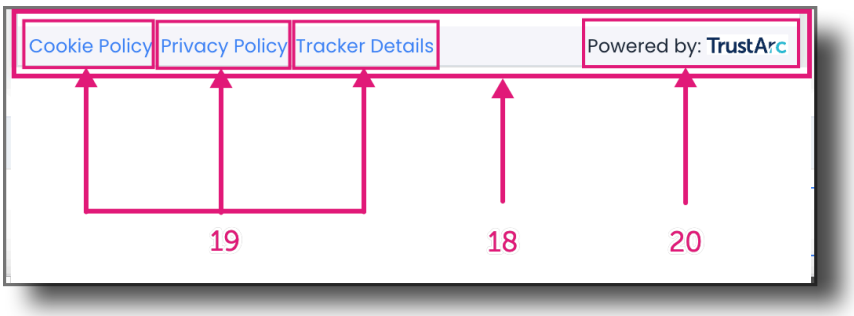
19. **.f-left**

- This holds the cookie policy, privacy policy, and tracker details links.
- Font properties and element's position can be overridden.
- This section can be hidden.

NOTE: To override the text color of the Privacy Policy link, you may use the class **.f-left a** to for the changes to be observed.

20. **.f-right**

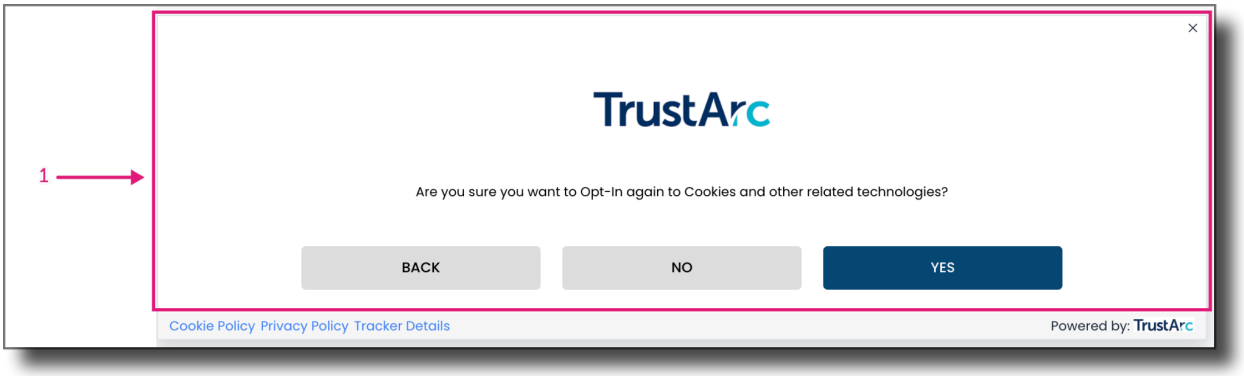
- This holds an image link that redirects the users to the TrustArc website.
- Font properties can be overridden.
- This section can be hidden.



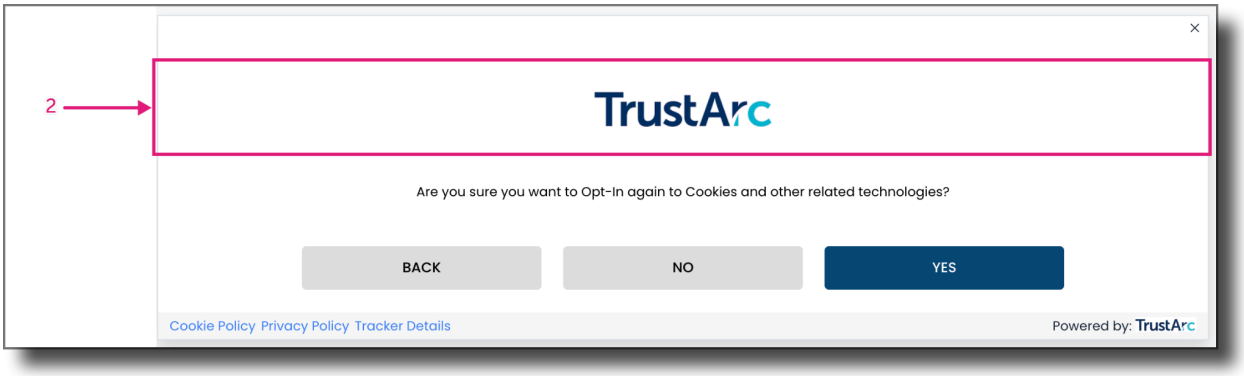
2 Step Opt-In Overlay Customization

This section focuses on the different class and id attributes in each `<div>` tags.

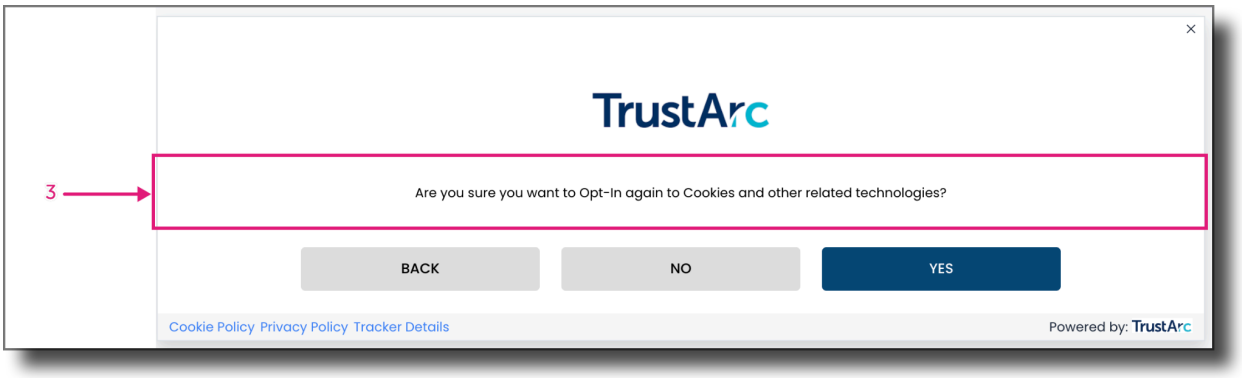
1. **.trustarc-two-step-background**
    - This serves as the container for all overlay contents and elements.



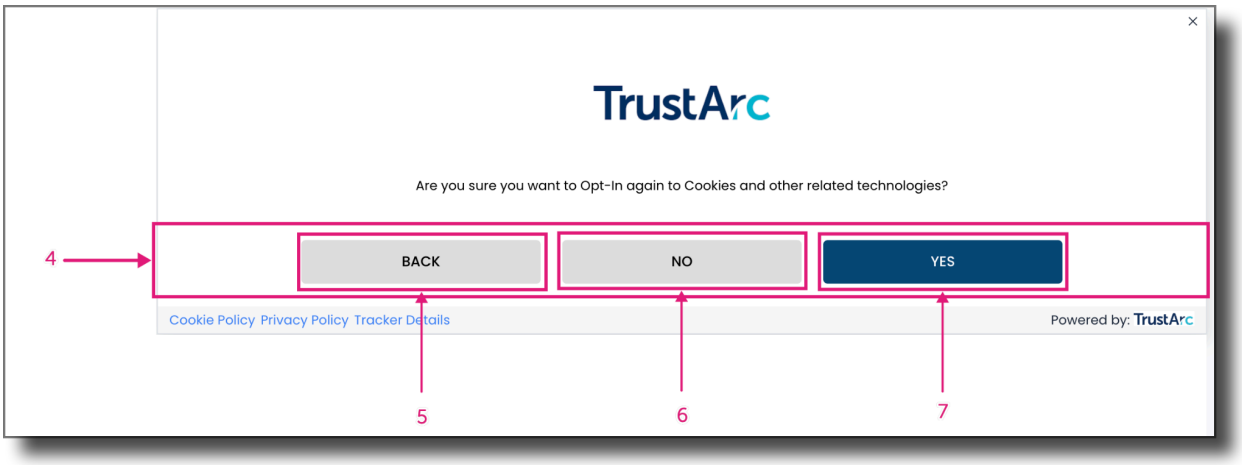
2. **.client-logo**
    - This holds the client's logo.
    - This section can be hidden.
    - Logo's width and margin can be overridden.



3. **.trustarc-two-step-body-text**
    - This holds the overlay consent text.
    - Font properties can be overridden.



4. **.twoStepBottom**
    - This serves as a container for all consent buttons.
5. **.trustarc-two-step-back-btn**
    - This button allows users to navigate the previous modal view.
    - Font properties can be overridden
6. **.trustarc-two-step-opt-out-btn**
    - This button allows users to keep the previous consent.
    - Font properties can be overridden.
7. **.trustarc-two-step-opt-in-btn**
    - This button allows users to update their consent.
    - Font properties can be overridden.



8. **.ta-footer-bottom**
    - This encapsulates all overlay elements.
    - Background color can be overridden.

- This section can be hidden.

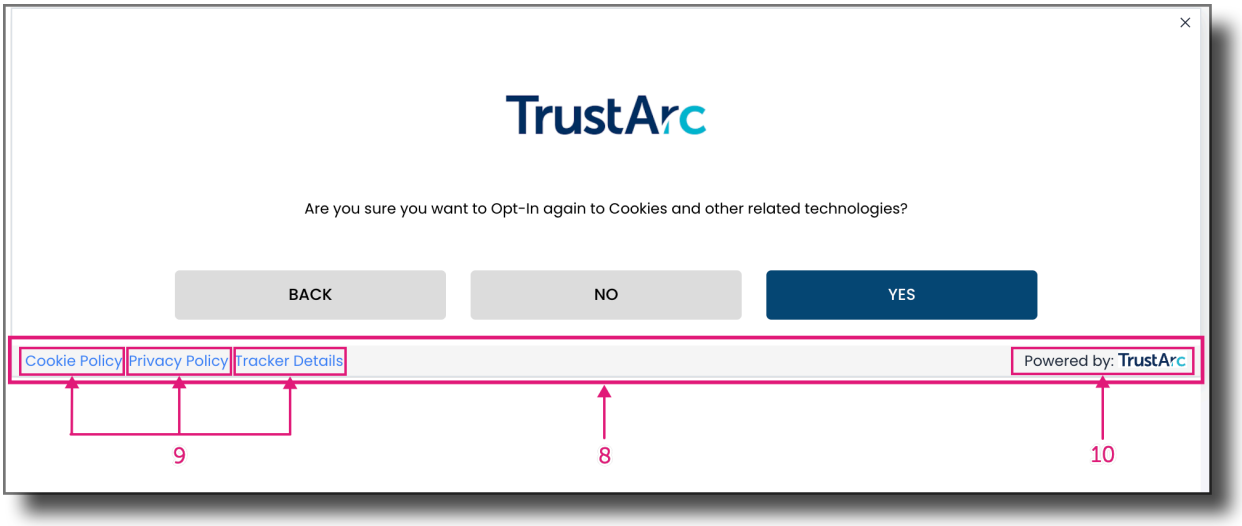
9. **.f-left**

   - This holds the cookie policy, privacy policy, and tracker details links.
   - Font properties and element's position can be overridden.
   - This section can be hidden.

   NOTE: To override the text color of the Privacy Policy link, you may use the class **.f-left a** to for the changes to be observed.

10. **.f-right**

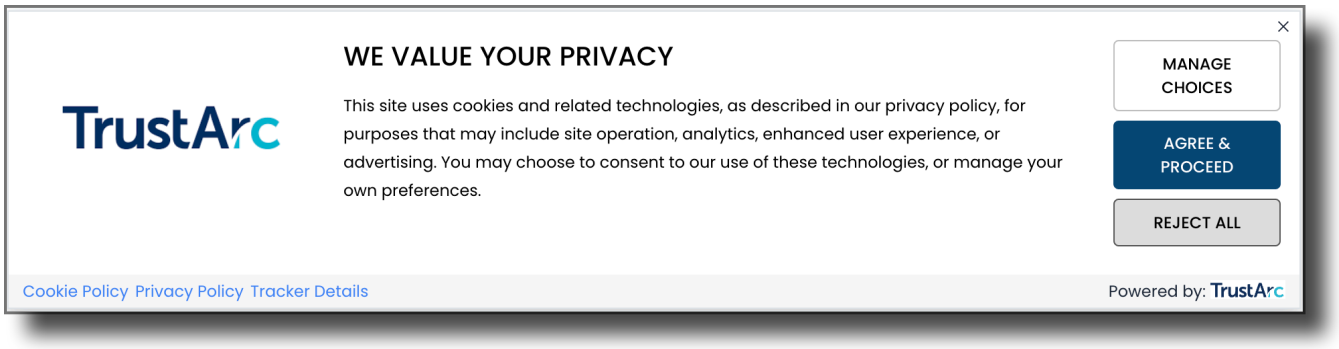
    - This holds an image link that redirects the users to the TrustArc website.
    - Font properties can be overridden.
    - This section can be hidden.



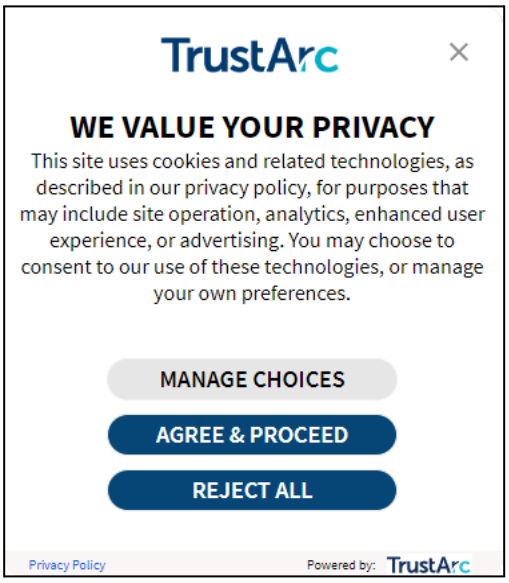
UI Examples

Desktop

Before Adding CSS:



After Adding CSS:



CSS example for above change:

None

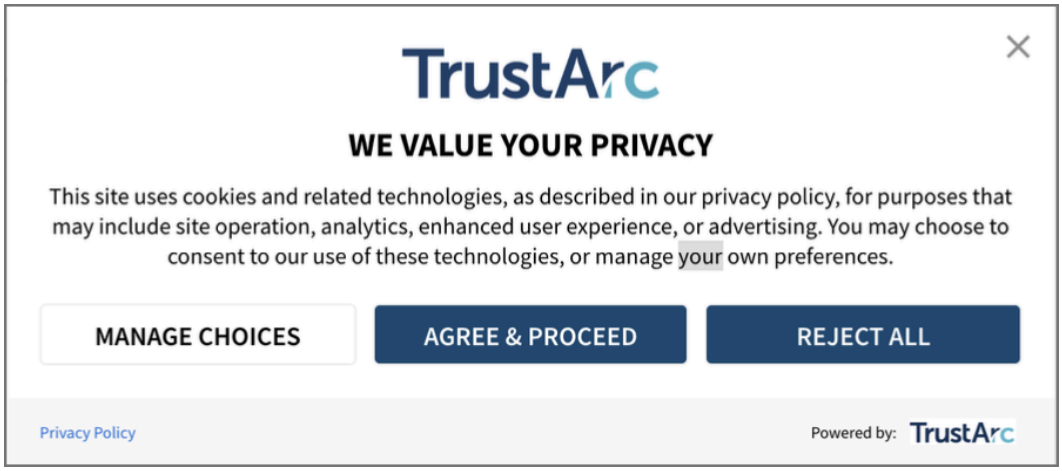
```
#consent-banner .trustarc-banner-wrapper {
    position: fixed;
    bottom: 200px;
    z-index: 1000;
    width: 375px;
    max-width: calc(100% - 20px) !important;
    margin: 20px auto 5% auto !important;
    left: 0;
    right: 0;
    border-radius: 15px;
    box-shadow: 0px 0px 10px 0px rgb(0 0 0 / 20%) !important;
}

.trustarc-background {
    background-color: #faf6f5;
}

#consent-banner .trustarc-banner-actions > div {
    width: 100% !important;
    justify-content: center !important;
    text-align: center !important;
    display: inline-flex;
    flex-wrap: wrap;
}

#consent-banner .trustarc-banner-footer {
    height: 20px;
    background-color: #faf6f5;
}

#consent-banner .trustarc-banner-actions > div > button {
    width: 216px !important;
}

#consent-banner .trustarc-banner-content {
    display: block !important;
}

#consent-banner .trustarc-primary-btn, #consent-banner .trustarc-secondary-btn,
.trustarc-reject-btn {
    font-size: 18px !important;
    line-height: 20px !important;
    border-radius: 25px !important;
    height: 30px !important;
}
```

```
div#truste-header-text {
    font-size: 25px !important;
}

.trustarc-logo-container {
    height: 45px !important;
}

.trustarc-client-logo {
    padding: 15px !important;
}

.trustarc-banner-details {
    text-align: center !important;
    padding: 0px !important;
    margin: 0 10px 0 10px;
}

#consent-banner .trustarc-primary-btn {
    float: right !important;
}

.trustarc-banner-actions {
    margin-right: 0px !important;
}

.trustarc-reject-btn:hover {
    background-color: #01579b;
}

.trustarc-primary-btn:hover {
    background-color: #01579b;
}
```

Tablet

For tablets, this covers screen width from **550px** to **950px**.

After Adding CSS:



CSS example for above change:

None

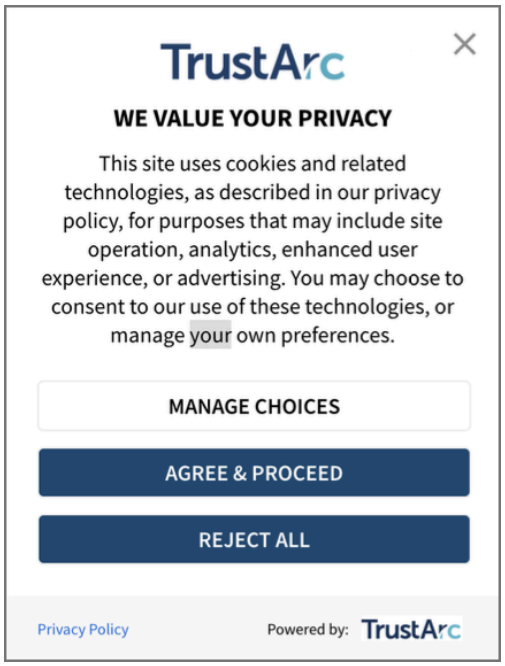
```
@media only screen and (min-width: 550px) and (max-width: 950px) {

    a#truste-consent-close {
        margin: 15px !important;
    }

    .trustarc-client-logo {
        padding: 20px 0px 10px !important;
    }

    .trustarc-logo-container {
        height: 40px;
    }

    .trustarc-banner-header {
        font-size: 20px !important;
    }

    .trustarc-body-text {
        font-size: 14px !important;
    }
```

```
    .trustarc-secondary-btn, .trustarc-primary-btn, .trustarc-acceptall-btn,
.trustarc-declineall-btn {
        font-size: 16px !important;
        height: 35px!important;
    }

    .trustarc-banner-actions > div {
        width: 100% !important;
        display: flex !important;
        flex-direction: row !important;
    }

    .trustarc-agree-btn {
        margin-right: 10px;
        margin-left: 10px;
    }
}
```

Mobile

For mobile, this covers screen width up to **549px**.

After Adding CSS:



CSS example for above change:

```
None
@media only screen and (min-width: 1px) and (max-width: 549px) {

    .trustarc-client-logo {
        padding: 10px 0px 0px !important;
    }

    .trustarc-logo-container {
        height: 50px !important;
    }

    .trustarc-client-logo img {
        height: 30px !important;
    }

    .trustarc-banner-header {
        font-size: 16px !important;
    }

    .trustarc-body-text, .trustarc-warning-text {
        font-size: 14px !important;
    }

    .trustarc-secondary-btn, .trustarc-primary-btn, .trustarc-acceptall-btn,
.trustarc-declineall-btn {
        font-size: 14px !important;
        height: 30px!important;
    }

    a#truste-consent-close {
        margin: 15px !important;
    }

    .trustarc-banner-actions > div {
        width: 100% !important;
    }
}
```

Consent Manager API Functions

Based on the level of end-user consent received via the Cookie Consent API, companies are able to manage tags on their websites.

Below is a list of the API functions that are available in the Consent Manager that may provide the functionality necessary to help customers utilize the end user's consent choices in a customized and unique manner, fitting for their website and needs, to provide users with their consent choices.

getConsent(...)

Description	This method provides the ability to ascertain the opt-out status for a specific vendor domain. **Note**: This is for the use of the website hosting Consent Manager and requesting specific vendor opt outs.		
callApi Parameters	Name	Value	Description
	`getConsent`	(string)	[required]
	`self`	(string)	[required] This represents the API caller (current website).
	`domain`	(string)	[optional] The vendor domain you are requesting opt in status for Default Value, which if left blank, the value is assumed to be the same as *self.location.hostname*
	`authority`	(string)	[optional] The domain name of the source from which the API caller derives permission to make the call

	`type`	(string)	[optional] The bucket type you are requesting opt in status for; values can be: • default types - "required," "functional," or "advertising" • bucket/category name - e.g. "Required Cookies" **NOTE:** This can only be used if there is already a consent. • bucket/category number - index starts at 1, e.g. "1" for Required Cookies
Sample JS Function Request			```var json = truste.cma.callApi("getConsent", "yourdomain.com", "targetVendorDomain.com", "authorizingBodyDomain.com", "functional");``` **NOTE:** If both `targetVendorDomain.com` and `functional` are specified and have conflicting opt-in status, the result will be the consent provided for the specified bucket (`functional`).
Sample Response			```json = {source:"asserted", consent:"denied"};``` Where: • `source` will have the following values: ○ **asserted** – The consent is expressed - i.e. achieved by an explicit user action (such as a mouse click) in an implied or expressed consent notice. ○ **implied** – The consent is implied - i.e. achieved via displaying a corresponding notice to the user where a user has not taken any explicit action, or set as a default preference by the website owner. • `consent` will have the following values:

	○ **approved** – based on the user preference, the party is allowed to set cookies and other related technologies ○ **denied** – based on the user preference, the party is NOT allowed to drop cookies and other related technologies Default Values (prior to consent): ● If the Consent Manager's behavior is expressed: ○ `{source:"implied", consent:"denied"};` ● If the Consent Manager's behavior is implied: ○ `{source:"implied", consent:"approved"};`

getConsentDecision(...)

Description	This method provides the ability to ascertain the consent level of the user that provided consent on your website. This function only returns the highest category index that the user opted in.		
callApi Parameters	Name	Value	Description
	`getConsentDecision`	(string)	[required]
	`self`	(string)	[required] This represents the API caller (current website).
Sample JS Function Request	`var json = truste.cma.callApi("getConsentDecision", "yourdomain.com");`		

Sample Response

```
json = {consentDecision:$integer, source:"asserted"};
```

Where:

- `consentDecision` values:
  - `$integer` = index of the highest category opted in; category index starts with 1, that is, by default:
    - `1` = required
    - `2` = functional
    - `3` = advertising
  - `0` = no preference set

  NOTE: The value returned is the highest category opted in. In the case where `functional` is opted out but `advertising` is opted in, the `consentDecision` value will still be 3.
- `source` will have the following values:
  - **asserted** – The consent is expressed - i.e. achieved by an explicit user action (such as a mouse click) in an implied or expressed consent notice.
  - **implied** – The consent is implied - i.e. achieved via displaying a corresponding notice to the user where a user has not taken any explicit action, or set as a default preference by the website owner.

  NOTE: If you have custom bucketing, you should factor in the number associated with your customized naming convention that should be returned in the `$integer` value.

  Please see the *Appendix* section of the *Cookie Consent Manager API Integration Guide* for the possible use case and resulting values this function can provide.

getGDPRConsentDecision(...)

<table>
<tr><td>Description</td><td colspan="3">This method provides the ability to ascertain the consent level that the user chose to a granular level. This method takes into account the multiple preference values that a user can now choose.</td></tr>
<tr><td rowspan="3">callApi Parameters</td><td>Name</td><td>Value</td><td>Description</td></tr>
<tr><td><code>getGDPRConsentDecision</code></td><td>(string)</td><td>[required]</td></tr>
<tr><td><code>self</code></td><td>(string)</td><td>[required]
This represents the API caller (current website).</td></tr>
<tr><td>Sample JS Function Request</td><td colspan="3"><code>var json = truste.cma.callApi("getGDPRConsentDecision","yourdomain.com");</code></td></tr>
<tr><td>Sample Response</td><td colspan="3"><code>json = {consentDecision: $integer_array, source: "asserted"}</code>

Where:
• <code>consentDecision</code> values:
○ <code>$integer_array</code> = provides the granular consent levels (represented by category indices) that the user selected. For example, if the user opted in to required and advertising, but opted out of functional, the value will be <code>[1,3]</code>
○ <code>[0]</code> = no preference set
• <code>source</code> will have the following values:
○ asserted – The consent is expressed - i.e. achieved by an explicit user action (such as a mouse click) in an implied or expressed consent notice.
○ implied – The consent is implied - i.e. achieved via displaying a corresponding notice to the user where a user has not taken any explicit action, or set as a default preference by the</td></tr>
</table>

	website owner. Please see the *Appendix* section of the *Cookie Consent Manager API Integration Guide* for the possible use case and resulting values this function can provide.

getConsentCategories(...)

Description	This method provides the ability to retrieve all the preference values set by an end user for all vendors.		
callApi Parameters	Name	Value	Description
	`getConsentCategories`	(string)	[required]
	`self`	(string)	[required] This represents the API caller (current website).
Sample JS Function Request	`var json = truste.cma.callApi("getConsentCategories","yourdomain.com");`		

Sample Response	Each domain/vendor will have one of the following values:
	• `0` – means opted-out
	• `1` – means opted-in
	• `2` – means no preference
	
	`json =`
	`{`
	`categories:{`
	`"Required Cookies":{`
	`"value":"0",`
	`"domains":{`
	`"bluekai.com":"2",`
	`"partner.bluekai.com":"2"`
	`}`
	`},`
	`"Functional Cookies":{`
	`"value":"1",`
	`"domains":{`
	`"2o7.net":"1",`
	`"everesttech.net":"1"`		

<table>
<tr><td></td><td>

```
            }
        },
        "Advertising Cookies":{
            "value":"2",
            "domains":{
                "ads.creative-serving.com":"0",
                "yieldoptimizer.com":"0",
                "adaptv.advertising.com":"0",
                "adap.tv":"1"
            }
        }
    }
}
```

NOTE: If there is no consent yet, the object returned will be `{categories: 'no categories'}`
</td></tr>
</table>

getDefaultCategories

<table>
<tr><td>Description</td><td colspan="3">This provides the ability to fetch the default category settings configured for a specific country, even before a user provides consent.</td></tr>
<tr><td rowspan="2">callApi Parameter</td><td>Name</td><td>Value</td><td>Description</td></tr>
<tr><td><code>getDefaultCategories</code></td><td>(string)</td><td>[required]</td></tr>
<tr><td>Sample JS Function Request</td><td colspan="3"><code>truste.cma.callApi('getDefaultCategories','<cminstanceid>');</code></td></tr>
<tr><td>Sample Response</td><td colspan="3">

```
country: "us"
defaultCategory: "Functional Cookies"
state: "az"
```
</td></tr>
</table>

setConsentLevel

Description	This method provides the ability to set a consent programmatically for the end user. When consent is set using this API, the consent manager banner/modal will be dismissed.		
callApi Parameters	Name	Value	Description
	`setConsentLevels`	(string)	[required]
	`yourdomain.com`	(string)	[required] This represents the API caller (current website).
	`consentArray`	[array]	[required] represents an array of consent category levels that will be opted-in.
Sample JS Function Request	`var json = truste.cma.callApi('setConsentLevels', 'truste.com ', [1,2,3]);`		
Sample Response	Each element in the array corresponds to a consent category set in the Consent Manager. For the default category, 1 represents Required Cookies, 2 represents Functional Cookies, and 3 represents Advertising Cookies. **NOTE:** Please check the Appendix for sample use cases.		

Event Handlers to Retrieve Notifications from Consent Manager

TrustArc provides the capability to notify a third-party on the publisher's site, publisher itself, or the tag management system of certain events that have occurred with the Consent Manager tool. If you choose to be notified of certain events such as receiving the end-user's consent level, looking up a specific vendor opt out, and/or trying to get information from within a Third Party IFrame, please see below for the possible methods to use.

Method to Retrieve End User Level Consent

To receive this notification, pass a Javascript callback function in the following way:

```javascript
var apiObject = {PrivacyManagerAPI:
       { action:"getConsentDecision",
         timestamp: new Date().getTime(), self: "domainThatIsAsking.com" }};

var json = JSON.stringify(apiObject); window.top.postMessage(json,"*");
window.addEventListener("message",  yourMessageHandler,  false);
```

Where `yourMessageHandler` is the javascript function that handles PostMessage events. Please see the sample section below regarding implementing this method to see an example of the response.

TrustArc Cookie Consent Manager will dispatch events in the following cases:

1. When an end user clicks on **Agree & Proceed** button.
2. When an end user finishes adjusting his/her preferences by clicking **Submit All Preferences** after going to the *Overlay* screen.
3. When an end user changes the previously set (or implied) settings by clicking **Submit All Preferences** after manually invoking TrustArc Cookie Consent Manager from the publisher's site.
4. When an implied preference is set outside of the Consent Manager modal window, for example, the Banner.

Using TrustArc API from Third-Party iFrame

For security purposes, not all third parties on the sites where publishers integrated TrustArc Cookie Consent Manager will be allowed to query user preferences about domains other than their own from inside third-party iFrames. This is because end user privacy preferences can be considered private information per se. They are intended only for the companies they apply to or for the companies who are capable to act upon them to enforce user preferences such as tag management companies. Therefore, all callers who wish to request consent for domains other than their own (i.e. other than the "origin" attribute of the PostMessage event) are required to obtain prior authorization from an authorizing body such as TrustArc to call this API in an asynchronous way.

Important: TrustArc will log all calls to its API on its server and will blacklist the iFrame domain names of the third parties whom it believes have not obtained the proper authorization to use this API.

Below is a typical example of how to make a previously mentioned `getConsent` call from a third-party iFrame.

```
None
var apiObject ={PrivacyManagerAPI:
       {action:  "getConsent",
       timestamp: new Date().getTime(),
       domain:  "targetDomain.com"
       self:   "domainThatIsAsking.com",
       authority:   "authorizingBody.com",
       type: "type1, type2, type3, ..."}};
var json = JSON.stringify(apiObject); window.top.postMessage(json,"*");
window.addEventListener("message",handleMessage,false);
```

This method accepts exactly the same parameters as described above for the synchronous `getConsent` call. However, when a call is made from inside an iFrame, TrustArc will assume the caller's identity based on the iFrame properties automatically set by the browser (i.e. the "origin" property of the PostMessage). If the request is for the user preferences for the same domain, the `authority` parameter will be ignored.

How to Implement HandleMessage Function

Below is an example of how callers could implement `handleMessage` function:

```
None
handleMessage(e){
       var json = JSON.parse(e.data);
       if(json &&
       json.PrivacyManagerAPI){
              switch(  json.PrivacyManagerAPI.consent){
                       case "denied":
                              break;
                       case "approved":
                              break;
              }
       }
}
```

The returned json will have the following properties:

```
None
{PrivacyManagerAPI:
        {capabilities:["getConsent"],
        source:"asserted",
        consent:"denied",
        action:  "getConsent",
        timestamp: 1234567890,
        domain:  "targetDomain.com"
        self:   "domainThatIsAsking.com",
        authority:   "authorizingBody.com",
        type: "type1, type2, type3, ..."}};
```

Where:

- `source` and `consent` properties will have the same range of values as described above
- `action`, `domain`, `self`, `authority`, and `type` properties will simply duplicate the input parameters in the initial asynchronous method calls
- `capabilities` will be an array of call names TrustArc has authorized the caller to make. (Right now this is simply a list of all calls supported by the API. This field can also be used for versioning purposes. TrustArc may change it in future versions of this API).

NOTE: To prevent any unnecessary delays, TrustArc will send the response immediately as soon as it knows the end user preference - not after it would log the call and verify whether the caller is authorized to make it. If it is found later that the caller did not obtain proper authorization, it will blacklist the caller and not respond to any future calls from this company.

Zero Cookie Load Integration Using Consent Manager API

A Zero Cookie/Tracker load integration with Consent Manager indicates that you wish to hold back all firing of any tags that do not meet or belong in the Required Cookies Bucket within your Consent Manager instance. For example, an end user visiting your site for the first time will only be served content and code that will drop ONLY required or strictly necessary cookies for the website to function properly.

Normally a Tag Management System (a system that can be used to integrate third-party software) is utilized to help achieve a Zero Cookie Load integration, but other technologies can be used as well. We do not recommend or restrict our clients from using any other technologies to achieve this functionality.

The CM API can be used to ascertain the consent level of the end user and based on that consent level, whether it be a bucket level consent or more granular, you can utilize this consent preference and pass that back accordingly to your technology of choice to ensure that Zero Cookie Load is being respected. The API will be the mechanism to grab the end user consent, vendor opt-outs, or specific events you will need to listen for in order to properly signal your technology to respect the choice of the end user and not fire those third-party tags and other code.

Verification and Troubleshooting

This section outlines how to verify a Zero-Tracker integration and some common troubleshooting steps.

Verifying a Zero-Tracker Integration

The process of verifying your CM API Integration will vary depending on the specifics of your deployment, CM configuration, and overall tools at your disposal. However, in general you will be testing that the correct blocking occurs in three situations:

- A new user with Standard Load
- A new user with Zero-Tracker Load
- Changing the existing consent for each category

A new user with Standard Load

When verifying a new user who should see a Standard Load, we are simply ensuring that all tags are fired as expected on the initial load. You should see the same tags fire after the integration as before the integration.

A new user with Zero-Tracker Load

When verifying a new user who should see a Zero-Tracker Load, we are verifying that non-essential tags that have the blocking conditions applied have NOT been fired when coming from a region requiring a Zero-Tracker Load. This should be done before any user interaction with the Consent Manager.

Once consent has been submitted, and if consent has been provided, the appropriate tags should fire on the next page load or dynamically by executing the tags.

Changing the existing consent for each category

When verifying for a user who has changed their consent, we are verifying that the correct tags are blocked or fired based on the change. For example, when verifying the change in consent from Advertising to Functional, we would expect to see any Advertising tags NOT to be fired on the next page load. Conversely, a change from Functional to Advertising should cause any Advertising tags to be fired. A change both to and from each consent category is recommended.

Important: To ensure consent preferences are appropriately honored, once a tag script or code has been executed on a page (i.e., a tag has fired), you will need to configure an automatic page refresh in order for the new consent preference to be reflected. Failure to execute a refresh will result in the prior tracking behavior (e.g., tracking will continue as if an opt-out had not occurred) to persist until a manual refresh is done by a web visitor themselves.

CM API Demonstration

For reference, you may visit this demo website where the CM API is being used to control third party tags so that these trackers will fire as per the consent preference of the user.

Use Case Scenarios

This section provides the common use case scenarios that a user may provide consent through their interaction with the TrustArc Consent Manager. Through these possible use cases, we also detail the possible results returned by the consent manager API calls. When referencing the table result values, please reference the use case scenario below for more details on when or how this might be collected.

Description	Opt-In User Action
1	When an EU user opens the page having TrustArc Consent Manager for the first time, the banner is added to the footer.
2	User closes the banner.[1]
3	User selects the **Required Cookies** level on TrustArc Consent Manager dialog using the *Slider/Radio* button.
4	User selects the **Functional Cookies** level on TrustArc Consent Manager dialog using the *Slider/Radio* button.
5	User selects the default option by clicking **Agree and Proceed** on TrustArc Consent Manager dialog. OR User selects the **Advertising Cookies** level on TrustArc Consent Manager dialog
6	User selects the **Required Cookies** level on TrustArc Consent Manager Advanced Settings or Granular Consent UI.
7	User selects the **Required, and Functional Cookies** level on TrustArc Consent Manager Advanced Settings or Granular Consent UI.
8	User selects the **Required, Functional, and Advertising Cookies** level on TrustArc Consent Manager Advanced Settings or Granular Consent UI.

[1] **NOTE:** For scenario 2, it is considering that the **Close** button is the same as the **Decline All**. Implementations for the **Close** button may change based on the customer's settings.

9	User selects the **Required, and Advertising Cookies** level on TrustArc Consent Manager Advanced Settings or Granular Consent UI.

getConsentDecision Results Table

If you are using the Slider version of the Consent Manager UI, these are the potential results that you will receive from the APIs:

Use Case	Source Value	CMAPI Value	Message from TrustArc	cmapi_cookie_privacy	cmapi_gtm_bl
1	implied	0	{consentDecision:[0], source:"implied"};	<empty>	<empty>
2	asserted	1	{consentDecision:[1], source:"asserted"};	permit 1 required	ga-ms-ua-ta-asp-bzi-sp-awct-cts-csm-img-flc-fls-mpm-mpr-m6d-tc-tdc
3	asserted	1	{consentDecision:[1], source:"asserted"};	permit 1 required	ga-ms-ua-ta-asp-bzi-sp-awct-cts-csm-img-flc-fls-mpm-mpr-m6d-tc-tdc
4	asserted	1,2	{consentDecision:[2], source:"asserted"};	permit 1,2 functional	ta-asp-bzi-sp-awct-cts-csm-img-flc-fls-mpm-mpr-m6d-tc-tdc
5	asserted	1,2,3	{consentDecision:3, source:"asserted"};	permit 1,2,3	<empty>
6	asserted			Same as Use Case #3	
7	asserted			Same as Use Case #4	

8	asserted			Same as Use Case #5
9	asserted			Same as Use Case #5

getGDPRConsentDecision Results Table

If you are using the Granular version of the Consent Manager UI, these are the potential results that you will receive from the APIs:

Use Case	notice_gdpr_prefs cookie value	CMAPI Value	Message from TrustArc	cmapi_cookie_privacy	cmapi_gtm_bl
1	<empty>	0	{consentDecision:[0], source:"implied"};	<empty>	<empty>
2	-1	0	{consentDecision:[0], source:"asserted"};	<empty>	<empty>
3	0	1	{consentDecision:[1], source:"asserted"};	permit 1 required	ga-ms-ua-ta-asp-bzi-sp-awct-cts-csm-img-flc-fls-mpm-mpr-m6d-tc-tdc
4	0,1	1,2	{consentDecision:[1,2], source:"asserted"};	permit 1,2 functional	ta-asp-bzi-sp-awct-cts-csm-img-flc-fls-mpm-mpr-m6d-tc-tdc
5	0,1,2	1,2,3	{consentDecision:[1,2,3], source:"asserted"};	permit 1,2,3	<empty>
6	0	1	{consentDecision:[1], source:"asserted"};	Same as Use Case #3	
7	0,1	1,2	{consentDecision:[1,2], source:"asserted"};	Same as Use Case #4	

8	0,1,2	1,2,3	{consentDecision:[1,2,3], source:"asserted"};	Same as Use Case #5
9	0,2	1,3	{consentDecision:[1,3], source:"asserted"};	Same as Use Case #5

setConsentLevels Results Table

If you are using this API to programmatically set consent for the user, below are the results in different use cases.

Sample Value	Parameters	Expected Results (notice_gdpr_prefs)	Expected Results (cmapi_cookie_privacy)
truste.cma.callApi('setConsentLevels', 'truste.com ', [1,2,3]);	&gtm=1	0,1,2	permit 1,2,3
truste.cma.callApi('setConsentLevels', 'truste.com ', [1]);	&gtm=1	0	permit 1,required
truste.cma.callApi('setConsentLevels', 'truste.com ', [1,2]);	&gtm=1	0,1	permit 1,2 functional
truste.cma.callApi('setConsentLevels', 'truste.com ', [1,2,3,4,5]);	&gtm=1	0,1,2,3,4	permit 1,2,3,4,5
truste.cma.callApi('setConsentLevels', 'truste.com ', [1,2,3]);	&gtm=1	0,1,2	permit 1,2,3
truste.cma.callApi('setConsentLevels', 'truste.com ', [1,2]);	&gtm=1	0,1	permit 1,2 functional
truste.cma.callApi('setConsentLevels', 'truste.com ', [1]);	&gtm=1	0	permit 1,required
truste.cma.callApi('setConsentLevels', 'truste.com ', [1,3]);	&gtm=1	0,2	permit 1,3

truste.cma.callApi('setConsentLevels', 'truste.com ', [1,2,3]);	autoblock enabled	0,1,2	permit 1,2,3
truste.cma.callApi('setConsentLevels', 'truste.com ', [3);	autoblock enabled	0,2	permit 1,3
truste.cma.callApi('setConsentLevels', 'truste.com ', [1);	autoblock enabled	0	permit 1,required
truste.cma.callApi('setConsentLevels', 'truste.com ', [1,3);	autoblock enabled	0,2	permit 1,3

Google Tag Manager

The following sections provide instructions to implement and utilize TrustArc's Cookie Consent Manager with Google Tag Manager's (GTM) triggers and rules. This guide is designed to work with GTM to correctly apply the triggers to your 3rd party and 1st party tags to provide support for the ePrivacy Directive and GDPR in the EU. You may use this guide to build a Zero Cookie/Tracker load (i.e., no non-required 3rd party tags can fire until the user provides consent) integration with Consent Manager or a deployment that submits opt-out preferences for the user's choices.

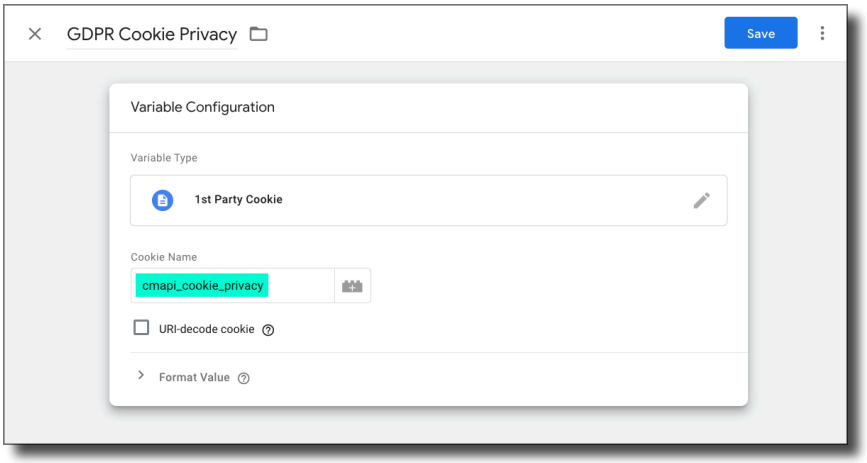
Before utilizing this integration guide, please make sure the following items are completed:

- At this point, you have migrated all your scripts to GTM.
- You need to know how many tags are on your site, and which category these tags belong to.
- You have also added the GTM tag to all your site pages.
- If you have questions regarding the use, purpose, or other configurations of GTM, please contact your Google Account Manager and/or visit their *online documentation* for websites.

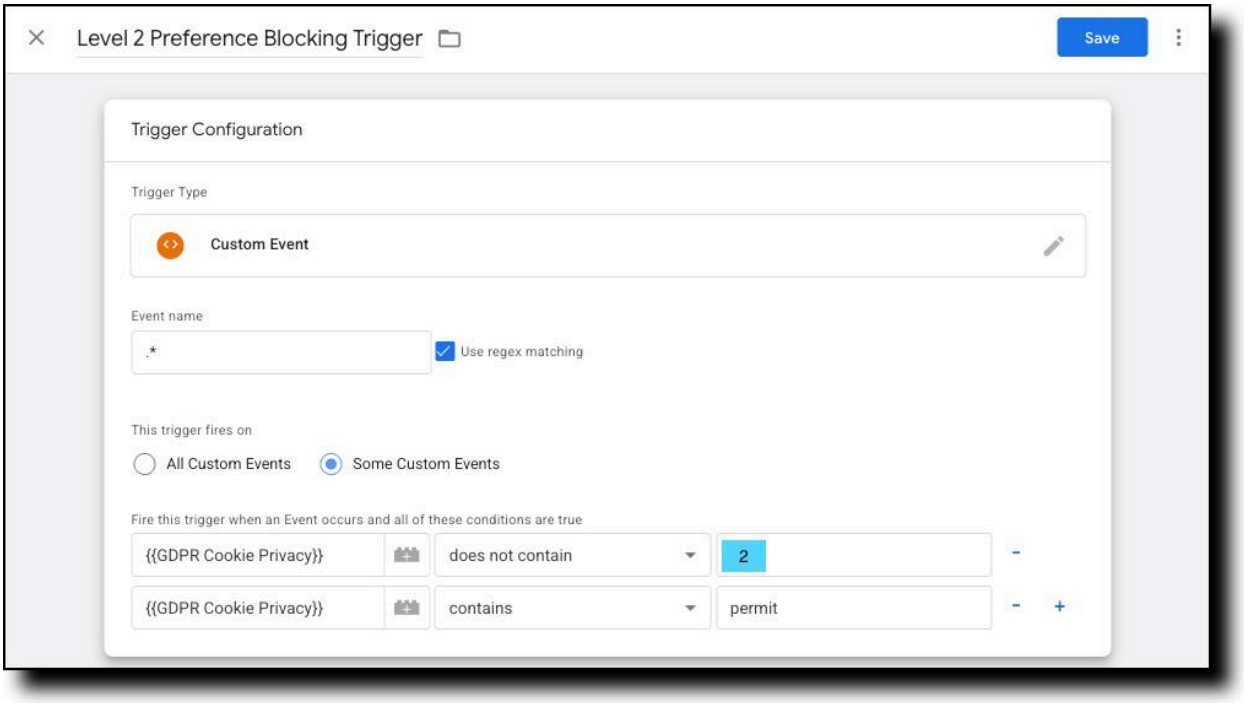
Creating Custom Variable and Triggers in GTM

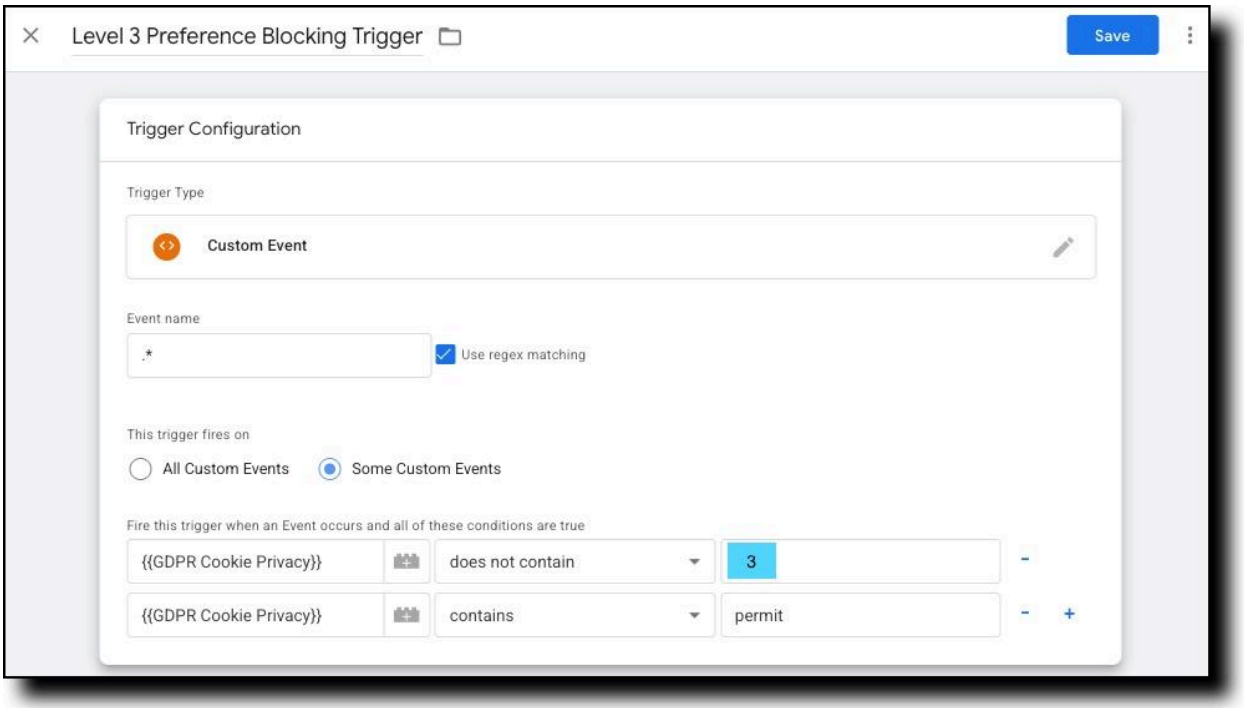
This section outlines the steps in creating custom variables and triggers in the GTM System. Follow the steps accordingly.

1. Add a GTM Variable named **GDPR Cookie Privacy**. Set the cookie name as **cmapi_cookie_privacy**.



2. Create individual Preference Level blocking triggers. Preference levels are by default: **1-Required**, **2-Functional**, and **3-Advertising**. Each should be triggered on "Custom Event", and the filter condition set as seen below, where the third field is the Preference Level (in this case, 3). You will need to create Level 2 and Level 3 Blocking Trigger in the system, please see below screenshot for example:



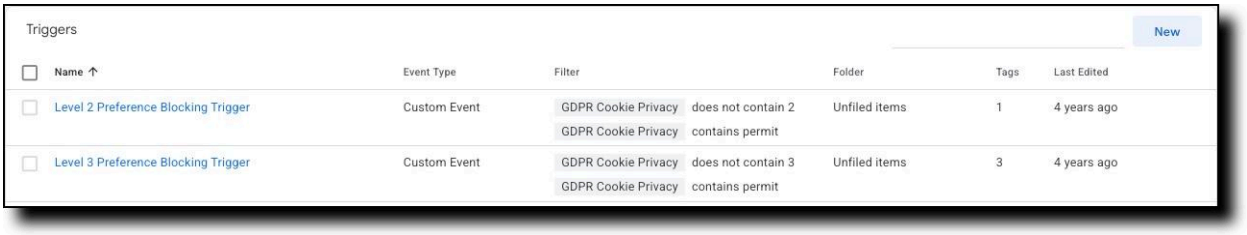


NOTE: **Uncategorized bucket will always take the highest value of 4 in your CM, if it exists**. If

you wish to account for the Uncategorized bucket vs going into the Self-Service CM UI to place the trackers in the right category level, you will have to account for the 4th value of Uncategorized cookies.

Level 1 Preference Blocking Trigger (for required trackers) is not necessary to set up within GTM since required cookies can fire, please note for due diligence, you must always categorize all non-required cookies.

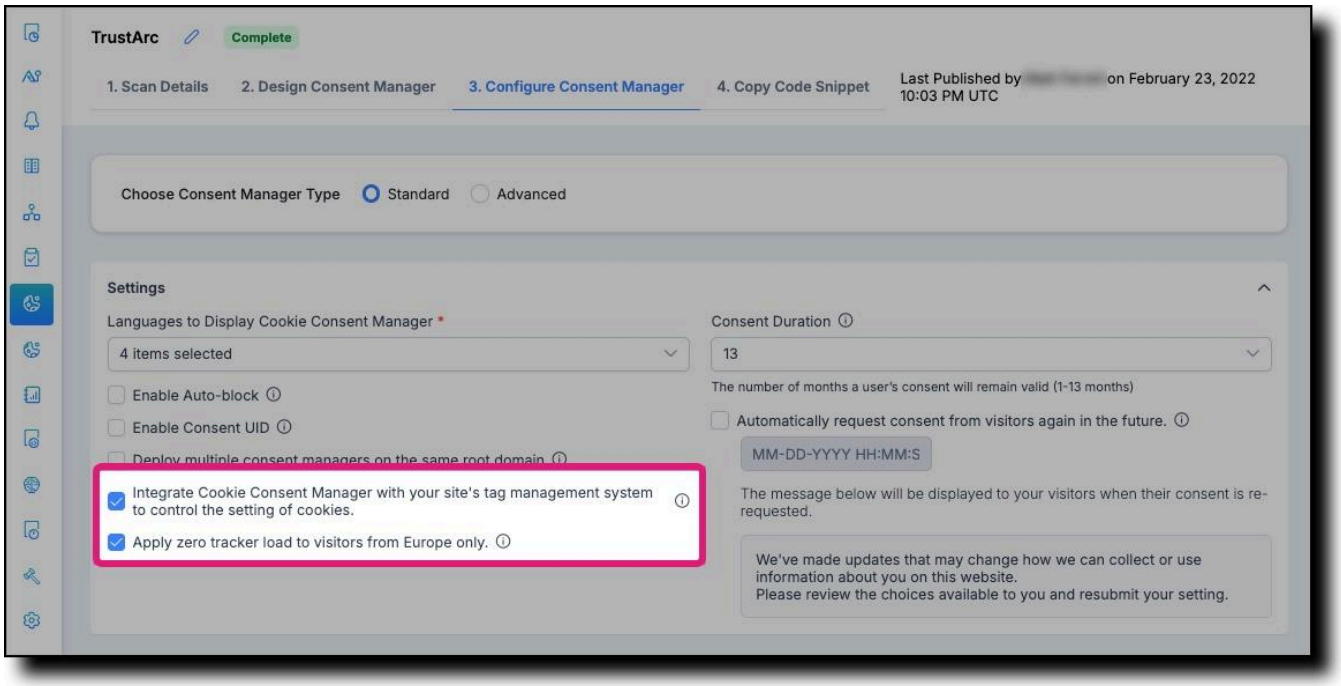
At the end of step one, your list of GTM Triggers should be as follows:



Deploying a Zero Cookie/Tracker Load Integration

A Zero Cookie/Tracker load integration with Consent Manager indicates that you wish to hold back all firing of any tags that do not meet or belong in the Required Cookies Bucket within your Consent Manager instance. An end user visiting your site for the first time will only be served content and code that will drop ONLY required or strictly necessary cookies for the website to function properly.

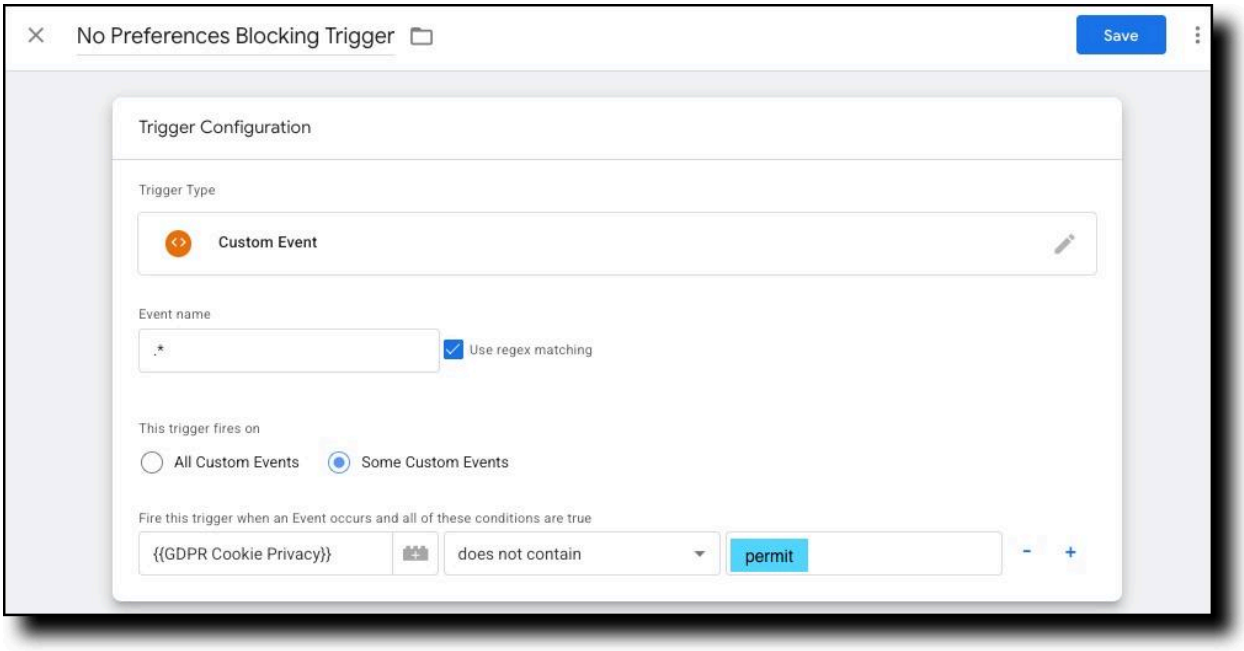
Important: To support this behavior, you will need to ensure that the tag management option in the *Setup* page is **enabled**.



NOTE: If you do NOT want to provide a Zero Tracker Load experience, you can skip this step.

Zero Cookie Load with Expressed Consent Configuration

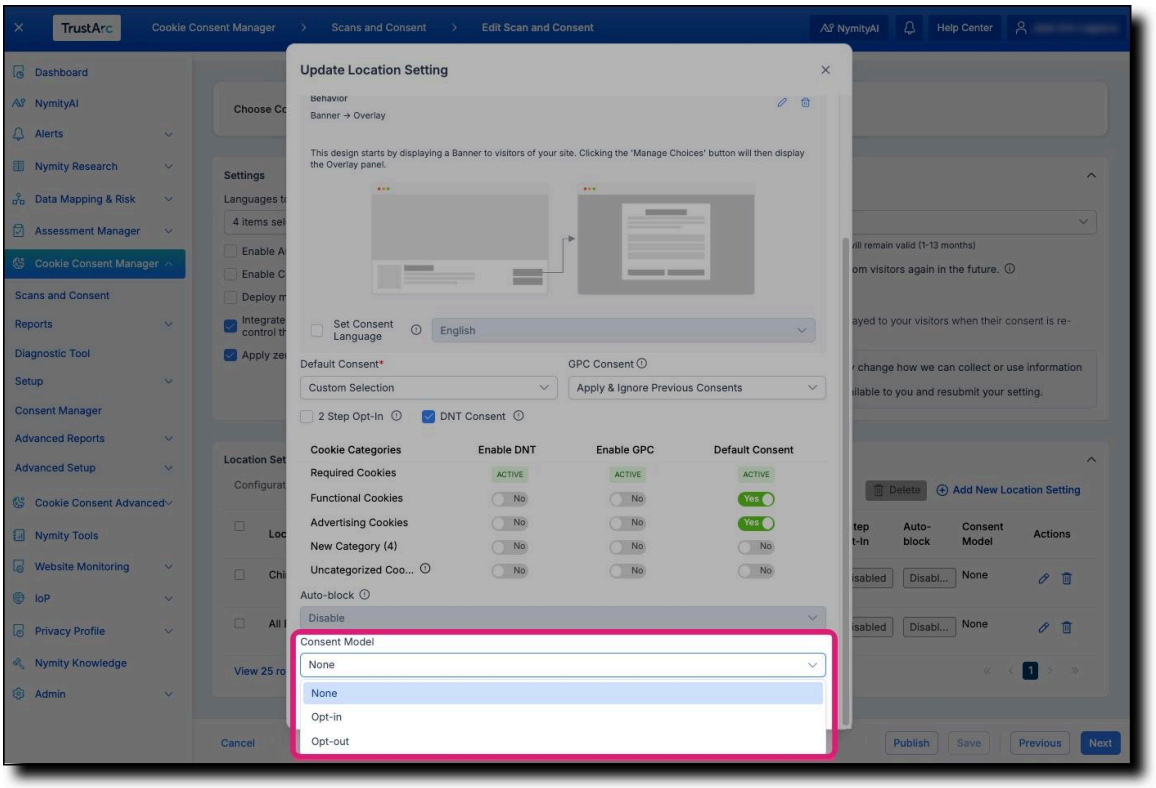
1. Apply a **No Preference Trigger** as a blocking rule (exception) to all tags.



2. If you wish to provide users from certain countries or states with Zero cookie load experience,

follow the steps below:

a. Make sure to properly configure the *Consent Model* settings for each group of countries or states under the *Location Settings* in the Step 3 tab of the Self-Service Portal.

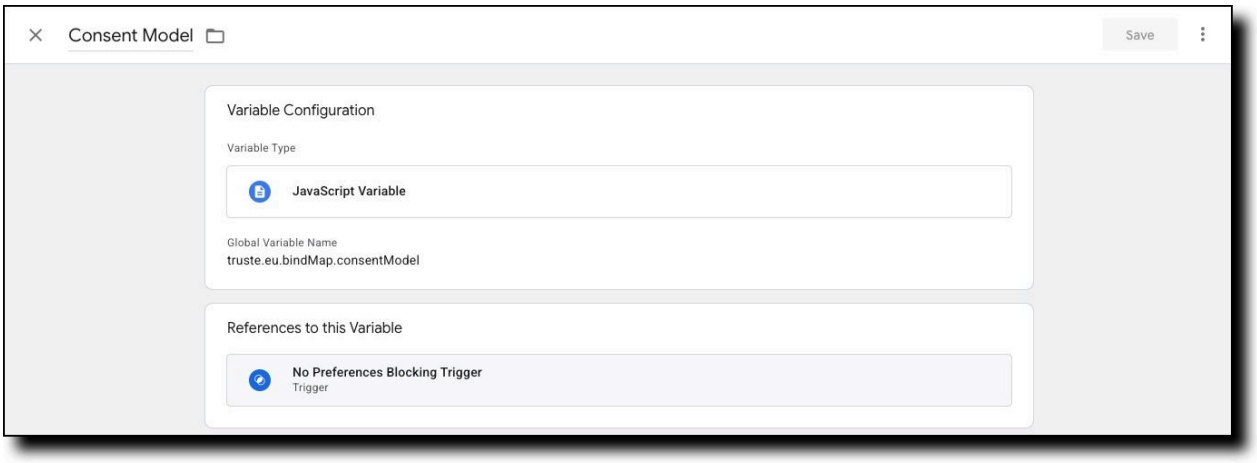


b. Choose the Opt-in value for countries or states that you want to apply the Zero-Tracker Load. While Opt-out for the countries or states that you want to apply the Standard Load.

c. Inside your GTM container, create a new variable named as Consent Model with the following values:

   i. **GTM Variable Type:** *JavaScript Variable*
   ii. **Variable Name**: *truste.eu.bindMap.consentModel*
   iii. **Variable Purpose:** *This cookie indicates the user's consent model, based on their location*
   iv. **Possible Values:**

- **opt-in** = Consent Model is opt-in (zero-tracker load)
- **opt-out** = Consent model is opt-out (standard load)



3. Add it to the *No Preference Trigger* above (step 1), using the 'contains' condition where the value is opt-in.

   NOTE: The `consentModel` API key is populated once Consent Manager is loaded. To ensure the `consentModel` API key is populated prior to tags being loaded, two general approaches can be made:

   I. Set the firing event for influenced trackers to be either **DOM Ready** or **Window Loaded**.

   II. Load Consent Manager outside of GTM

      A. Include the Consent Manager script directly on the page (HEAD element suggested).

      B. Create a custom event to be triggered once Consent Manager has loaded.

      C. The action of the custom event is to load GTM elements onto the page.

Adding TrustArc Integration Assets

To add TrustArc Integration Assets, follow these steps:

1. Add TRUSTARC CM script to your GTM container. Apply firing triggers, which will place this tag on all the pages you wish covered by the Cookie Consent Manager. This is to load the Cookie Consent Manager.

   Here is a sample JS code:

```javascript
<!-- Replace {cmId} with the six (6) character unique ID of the consent manager -->
<div id="consent-banner"></div>
<div id="teconsent">
<script type="text/javascript" async="async"
src="//consent.trustarc.com/v2/notice/{cmId}"></script>
```

```
</div>
```

2. Add the TrustArc DIV to your HTML
    - Option 1: Directly into your site. **Recommended Deployment**

```
None
<!-- This is where the cookie icon will show up -->
<div id="teconsent"></div>
<!-- This is where the banner will show up -->
<div id="consent-banner"></div>
```

NOTE: You may use other HTML elements besides a `<div>` such as `<span>`

    - Option 2: Implementation via GTM.

      You will include the script below into GTM as a custom HTML tag.

```
None
var d = document.getElementById("{{YOUR_HTML_ELEMENT_ID}}");
var e = document.createElement("div");
var f = document.createElement("div");
e.id = "teconsent";
f.id = "consent-banner";
d.appendChild(e);
d.appendChild(f);
</script>
```

Important: The `YOUR_HTML_ELEMENT_ID` is a macro and does not exist. This needs to be replaced with a custom value ID of the parent element that you wish to add the CM script to.

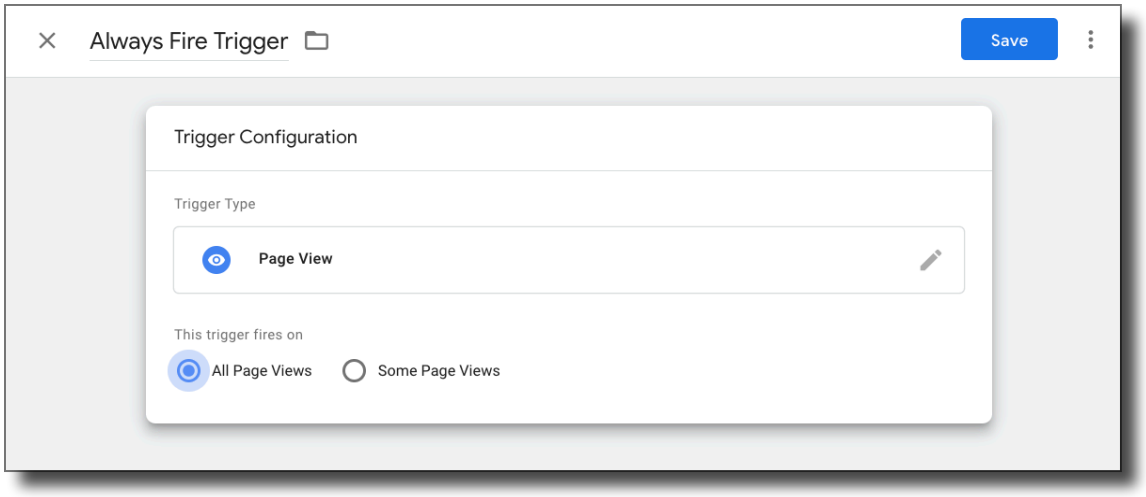
NOTE:

- Deploying Cookie Consent Manager via GTM has been known to cause issues with tags being fired prior to consent being provided by the user. To minimize this issue when deploying via GTM please use the Window Loaded Page View trigger for your tags.
- You can combine step 1 and step 2 - Option 2 in a single Custom HTML Tag.
- The TrustArc Cookie Consent script by default drops opt-out cookies when the user submits preferences. In order to leverage the GTM integration to honor user preferences, the script must have the Tag Management System feature enabled in Step 1 of the

Self-Service UI Configuration Wizard.

Firing Rules

You will likely have your own rules regarding when to load certain tags. These are called firing rules, and you should apply them to all your tags as appropriate. If your tags do not have a specific firing rule and you wish to load the tag on all pages, then you can use the following trigger:



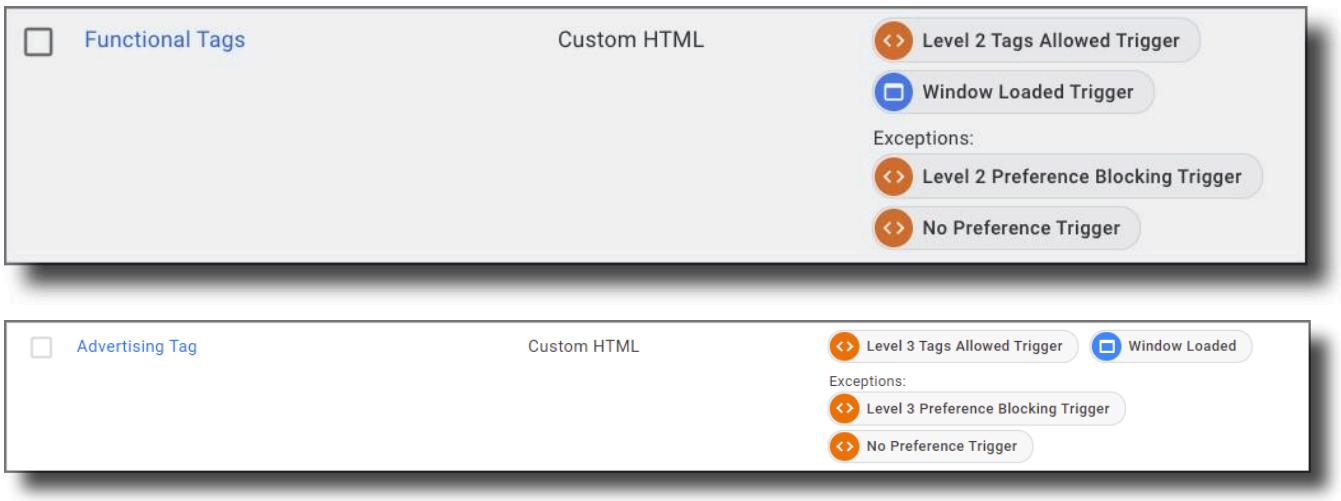
Preference Blocking Triggers

You can apply the Preference Blocking Triggers to all the tags you wish to be affected by the TrustArc CM. To do this, follow the steps outlined below.

1. Add Level 2 Preference Blocking Trigger as a blocking rule (exception) to all Functional (or any term you use for your level 2) tags.
2. Add Level 3 Preference Blocking Trigger as a blocking rule (exception) to all Advertising (or any term you use for your level 3) tags.
3. If you have more levels (4, 5, 6 etc), you need only increment the value and follow the same steps.

   NOTE: Uncategorized always takes the greatest level value, if it exists.

Example setup:



Event-based Firing Rules

The above integration is cookie-based, and thus rules can fire immediately upon page load, loading the relevant tags instantly. Another approach is event-based firing rules, which load tags once an event occurs on the page. This is a much more direct and precise control, allowing tags to be fired at any time. For example, using event-based rules, tags can be loaded immediately when a user sets or changes their preference, instead of waiting for the next page load.

However, events are not dispatched until the TrustArc javascript tags have loaded on the page. If your goal is easy integration, then you can use only event-based firing rules, but if load time is paramount, you can use event-based rules in conjunction with the above cookie-based rules, to get the best of both.

Create GTM Custom Events

First, create the following Custom Event Triggers. If you have more than the default Required, Functional, Advertising tag buckets, you can continue to add more Custom Event Triggers as appropriate.

Apply Custom Events to Your Tags

After you've created the custom event triggers, you need only apply the trigger to its corresponding tag. Following the naming scheme depicted above, the "Functional Tags Allowed" Trigger would be applied as a firing rule to all of the tags bucketed as Functional, and the "Ad Tags Allowed" Trigger would be applied as a firing rule to all of the tags bucketed as Advertising.

Add Custom Javascript tag to Trigger Events

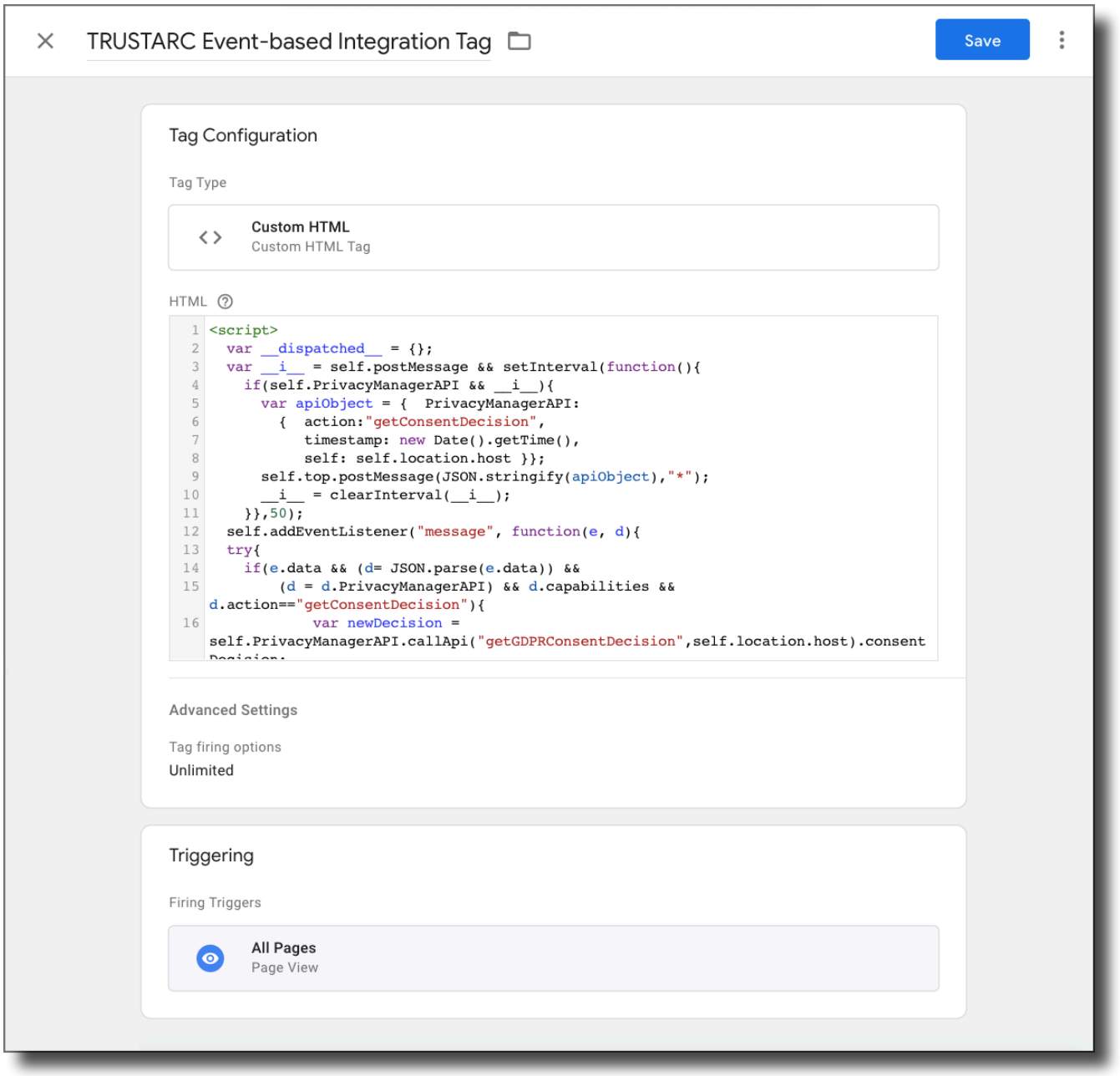
By default, the TrustArc code does not have the ability to generate the required GTM events listed in this section. You must add, either the page HTML as a static script or as another GTM javascript tag, the following code.

None

```
var __dispatched__ = {}; //Map of previously dispatched preference levels

  /* First step is to register with the CM API to receive callbacks when a preference
     update occurs. You must wait for the CM API (PrivacyManagerAPI object) to exist
on
     the page before registering.
```

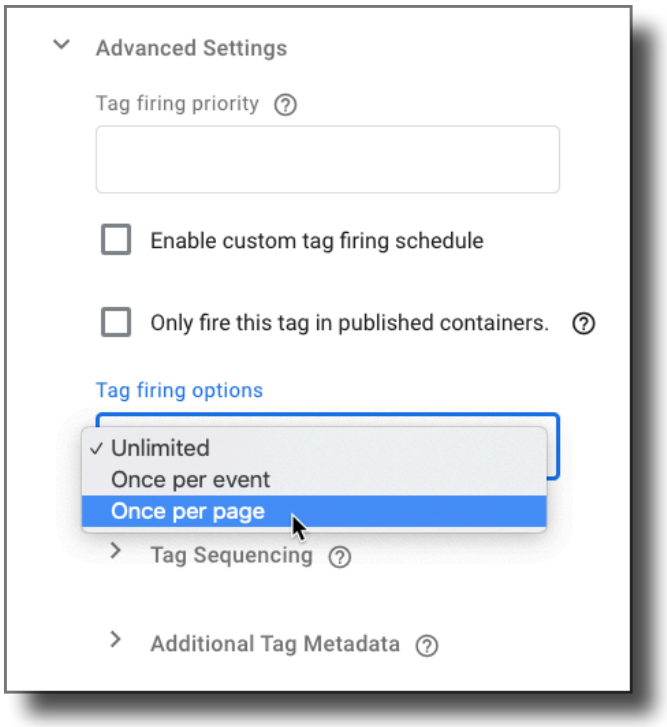
```
  */
  var __i__ = self.postMessage && setInterval(function(){
    if(self.PrivacyManagerAPI && __i__){
      var apiObject = {  PrivacyManagerAPI:
           {  action:"getConsentDecision",
              timestamp: new Date().getTime(),
           self: self.location.host }};
      self.top.postMessage(JSON.stringify(apiObject),"*");
      __i__ = clearInterval(__i__);
    }},50);

  /*
    Callbacks will occur in the form of a PostMessage event.
    This code listens for the appropriately formatted PostMessage event,
    gets the new consent decision, and then pushes the events into the GTM framework.
    Once the event is submitted, that consent decision is marked in the 'dispatched'
map
    so it does not occur more than once.
   */
  self.addEventListener("message", function(e, d){
  try{
    if(e.data && (d= JSON.parse(e.data)) &&
        (d = d.PrivacyManagerAPI) && d.capabilities &&
d.action=="getConsentDecision"){
            var newDecision =
self.PrivacyManagerAPI.callApi("getGDPRConsentDecision",self.location.host).consentDec
ision;
             newDecision && newDecision.forEach(function(label){
             if(!__dispatched__[label]){
                  self.dataLayer && self.dataLayer.push({"event":"GDPR Pref Allows
"+label});
                  __dispatched__[label] = 1;
               }
            });
      }
    }catch(xx){/** not a cm api message **/}
  });
```

If you wish to add the above code to GTM to be loaded as a tag, then it would need to be added to all pages which the TRUSTARC CM tag is added, that is, all pages which you want to be covered by these preferences in this guide. To do so, you would create a new Custom HTML Tag, like the following:


TRUSTARC Event-based Integration Tag
Save
Tag Configuration
Tag Type
Custom HTML
Custom HTML Tag
HTML
<script>
var __dispatched__ = {};
var __i__ = self.postMessage && setInterval(function(){
if(self.PrivacyManagerAPI && __i__){
var apiObject = { PrivacyManagerAPI:
{ action:"getConsentDecision",
timestamp: new Date().getTime(),
self: self.location.host }};
self.top.postMessage(JSON.stringify(apiObject),"*");
__i__ = clearInterval(__i__);
}},50);
self.addEventListener("message", function(e, d){
try{
if(e.data && (d= JSON.parse(e.data)) &&
(d = d.PrivacyManagerAPI) && d.capabilities &&
d.action=="getConsentDecision"){
var newDecision =
self.PrivacyManagerAPI.callApi("getGDPRConsentDecision",self.location.host).consent
Advanced Settings
Tag firing options
Unlimited
Triggering
Firing Triggers
All Pages
Page View

Using Event and Cookie-based Triggers Together

It's possible to use the Event-based and the Cookie-based Triggers together. This setup would provide for instant tag loading when preferences are already set, and allow for instant tag loading when preferences are set or changed, without a page load. You would simply follow all the steps above in this guide, but then there is one additional step. To prevent tags from firing more than once per page, for ALL tags you integrate, you need to open the tag's *Advanced Settings* and select **Once per page**.

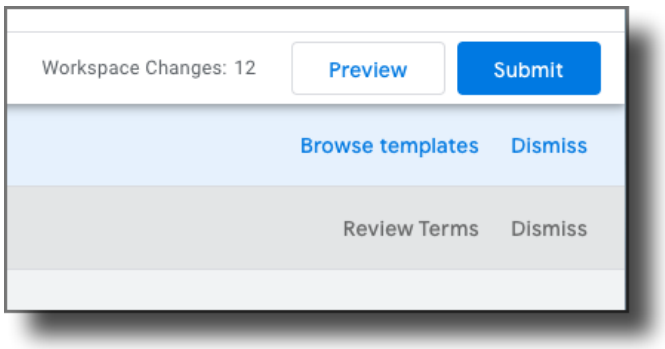


Verification and Troubleshooting

This section outlines how to verify a Zero-Tracker GTM integration and some common troubleshooting steps.

GTM Preview Mode

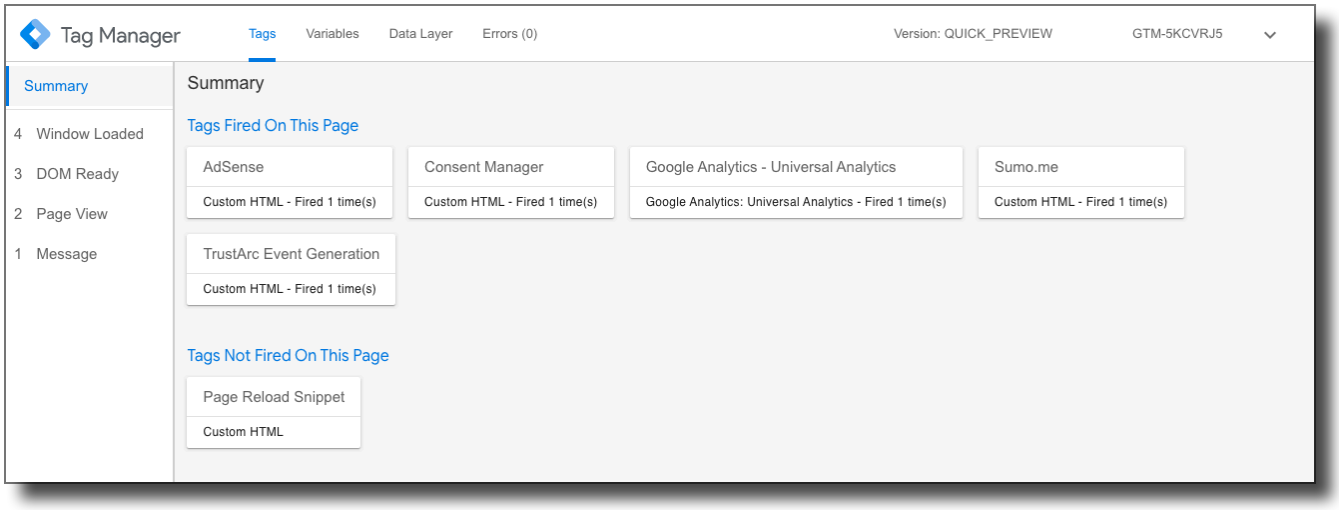
The **Preview** mode in GTM can be a vital tool when verifying and troubleshooting a Zero-Tracker integration as it allows you to examine the state of each variable and trigger condition for every tag at each GTM Event. If you have the appropriate permissions for the GTM container, you can launch the Preview Mode by clicking the Preview button next to the Submit button within GTM.



When viewing your site with your browser in **Preview** mode you can examine information for each tag fired on each of the GTM events.

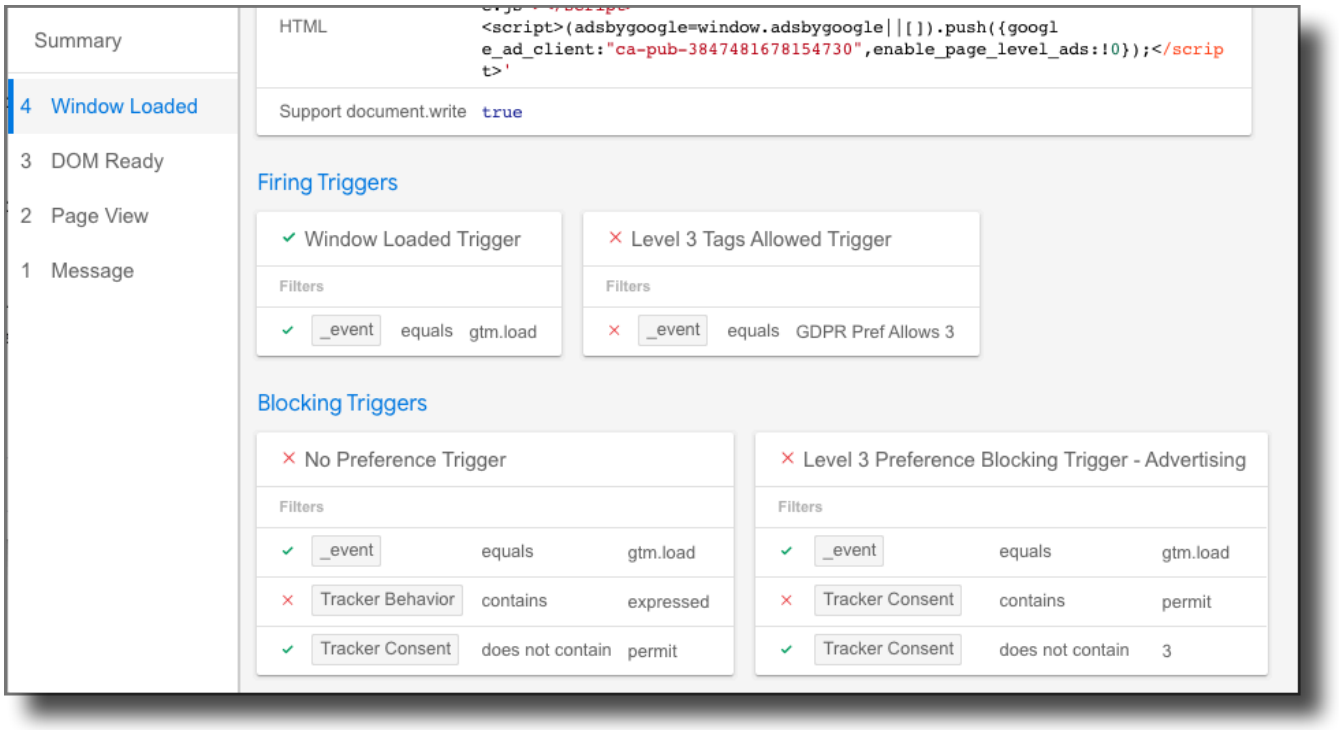
Summary

The **Summary** information displays the individual tags that were fired and blocked on each GTM Event which occurred.



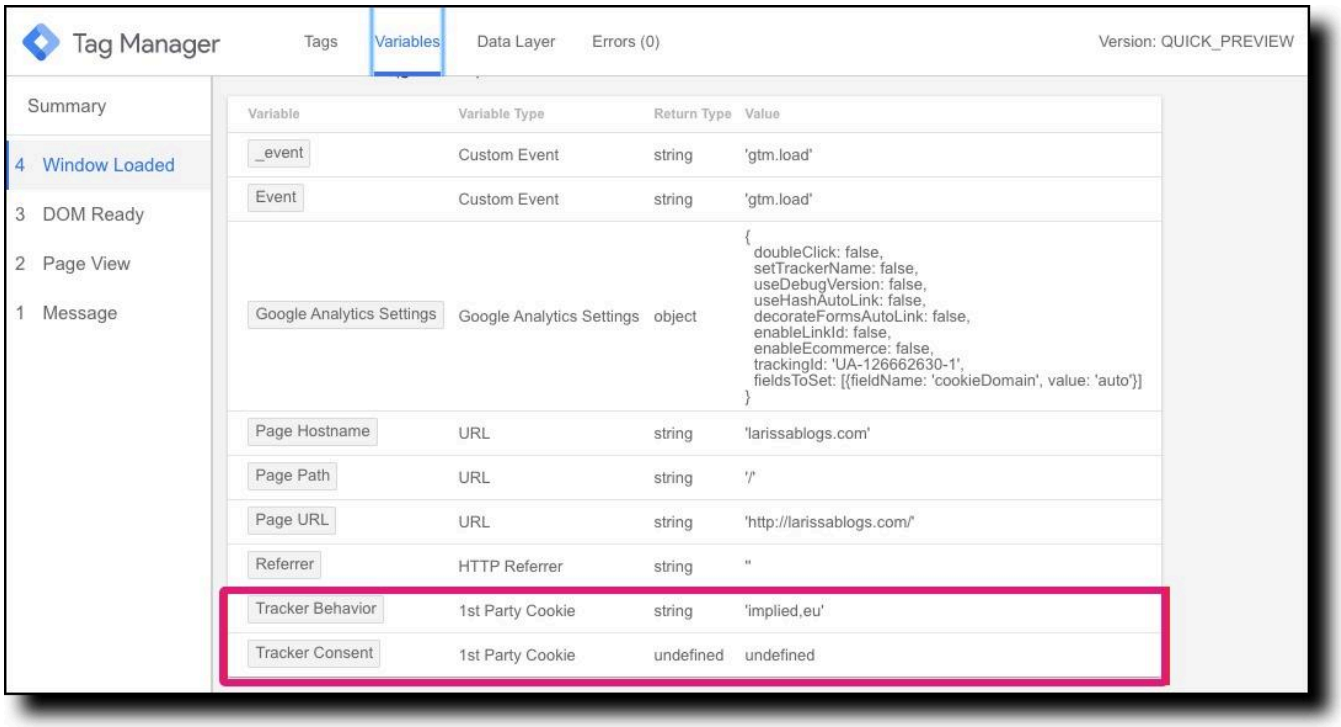
Tags

Clicking on any of the tags when a GTM Event is selected will display a snapshot of the pertinent information for that tag at that event. Included in this information are the states of the individual trigger conditions of any applied trigger.



Variables

Selecting the Variables tab when a GTM Event is selected will display a snapshot of the information contained within each variable.



Verifying a GTM Zero-Tracker Integration

The process of verifying your GTM Zero-Tracker Integration will vary depending on the specifics of your deployment, CM configuration, and overall tools at your disposal. However, in general you will be testing that the correct blocking occurs in three situations:

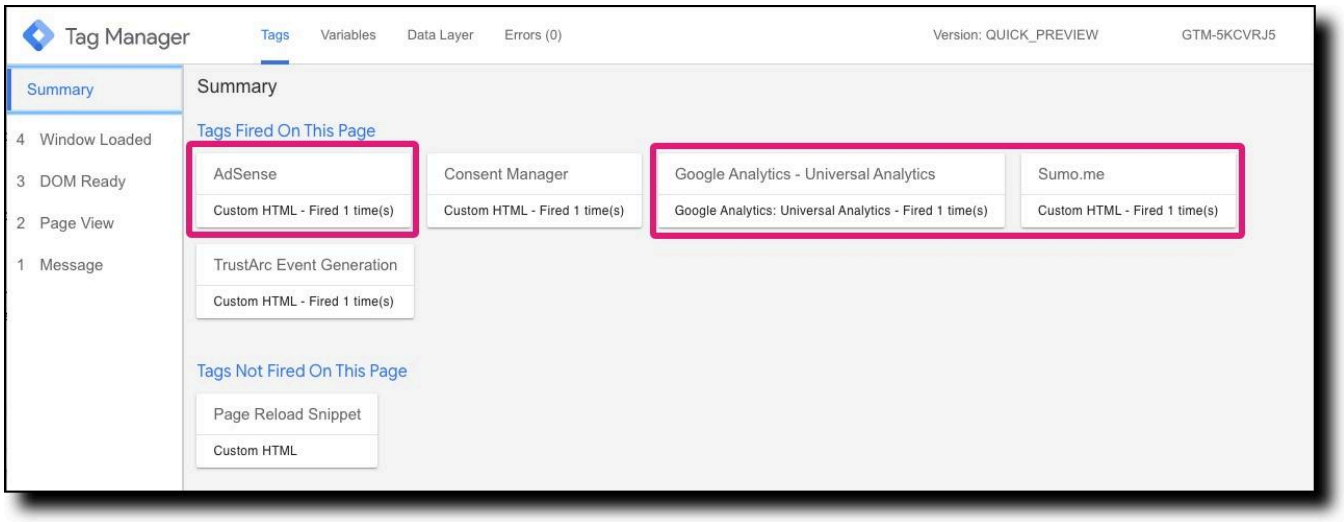
- A new user with Standard Load
- A new user with Zero-Tracker Load
- Changing the existing consent for each category

It is recommended to verify the mechanics and functionality of the GTM Integration with a single tag of each consent category prior to applying the integration to your GTM implementation as a whole.

In the following examples the AdSense tag is a Level 3 tag and the Google Analytics and Sumo.me tags are Level 2 tags.

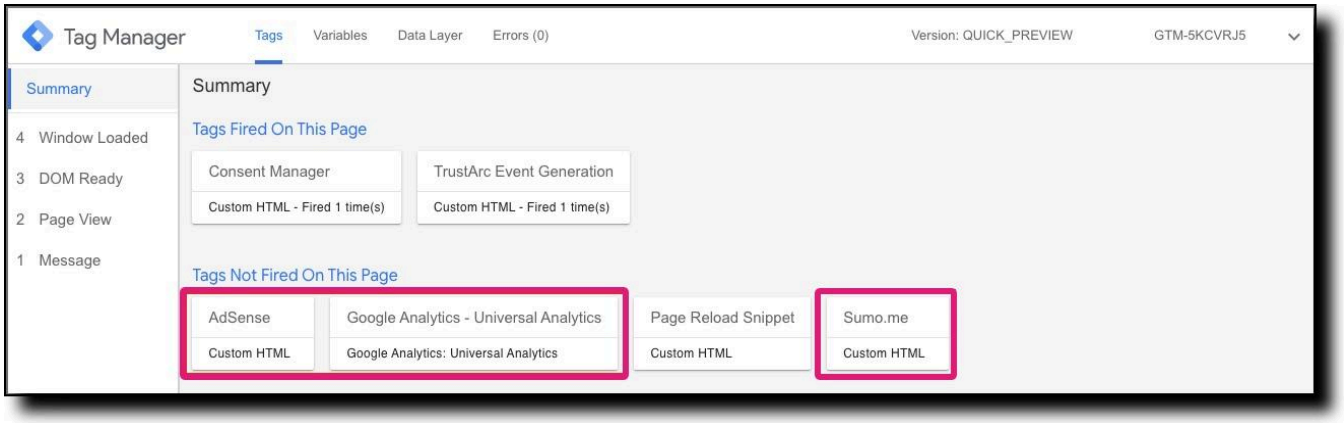
A new user with Standard Load

When verifying a new user who should see a Standard Load, we are simply ensuring that all tags are fired as expected on the initial load. You should see the same tags fire after the integration as before at each GTM Event. If you are using GTM to deploy any of the CM assets you should also see those tags fired on each page.



A new user with Zero-Tracker Load

When verifying a new user who should see a Zero-Tracker Load, we are verifying that all of the tags that have the Blocking Triggers applied have NOT been fired. This should be done before any user interaction with the Consent Manager.



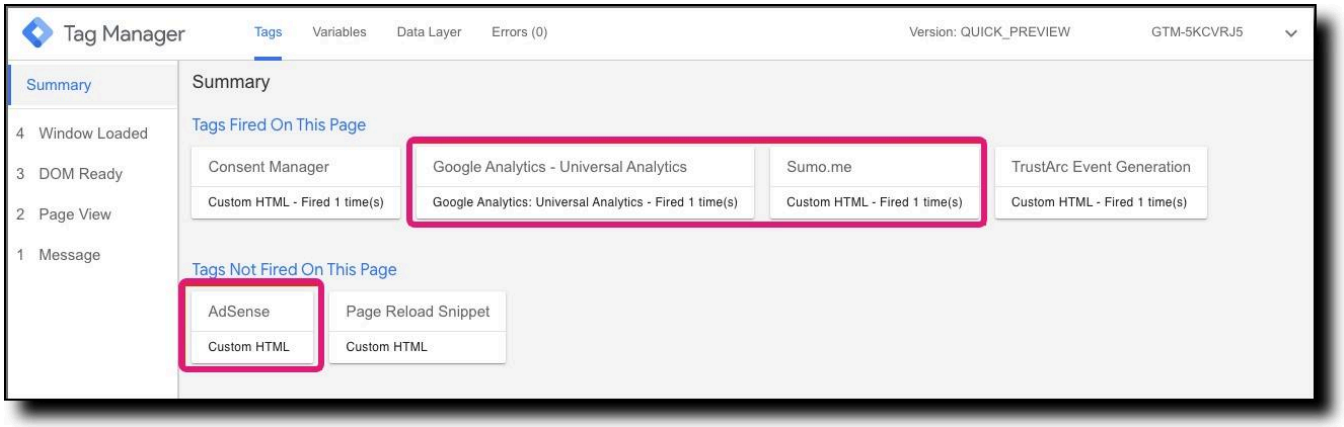
Once consent has been submitted, and if consent has been provided, the appropriate tags should fire immediately.

Changing the existing consent for each category

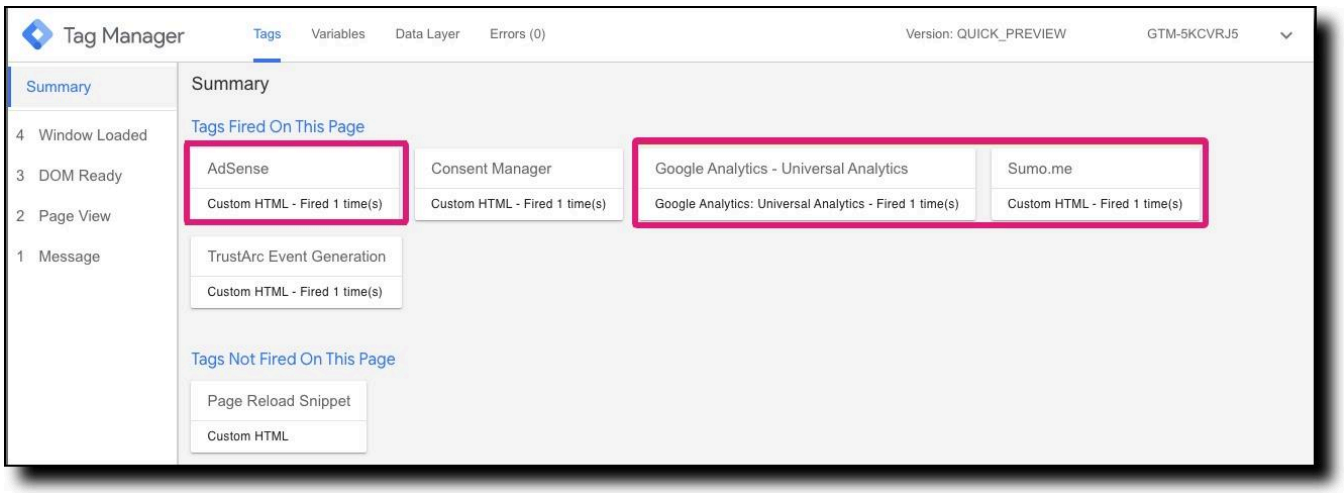
When verifying for a user who has changed their consent we are verifying that the correct tags are blocked or fired based on the change. For example, when verifying the change in consent from *Advertising* to *Functional* we would expect to see any tags with the *Level 3 Preference Blocking Trigger* applied NOT to be fired on the next page load. Conversely, a change from *Functional* to *Advertising* should cause any tags with the *Level 3 Preference Blocking Trigger* applied to be fired. A change both to and from each consent category is recommended.

Important: To ensure consent preferences are appropriately honored, once a tag script or code has been executed on a page (i.e., a tag has fired), you will need to configure an automatic page refresh in order for the new consent preference to be reflected. Failure to execute a refresh will result in the prior tracking behavior (e.g., tracking will continue as if an opt-out had not occurred) to persist until a manual refresh is done by a web visitor themselves.

With Level 2 enabled / Level 3 disabled



With Level 2 and Level 3 enabled



Troubleshooting

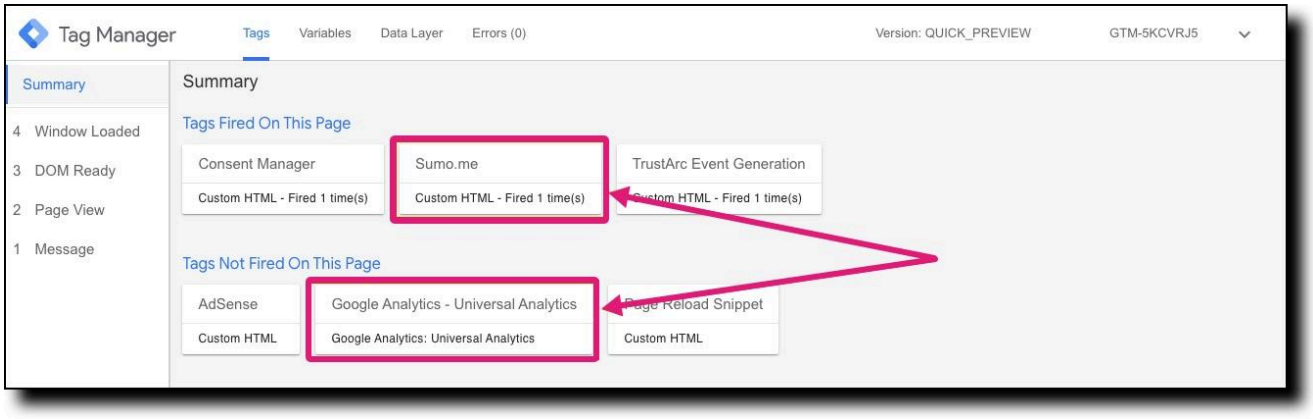
I'm not seeing the expected Zero-Tracker Load occur

The most common cause of a Zero-Tracker Load not occurring when expected is that the **Tracker Behavior** GTM Variable does not have data when the tags firing trigger conditions are met, preventing the corresponding blocking trigger conditions from being met as well. This frequently occurs when GTM is used to deploy the CM script to the page and in these cases the CM script should be moved to fire as early in the header of the page as possible. If this is not possible, or if this does not resolve the issue, you will need to change all of the tags trigger conditions to fire on a later event when the **Tracker Behavior** GTM Variable does have data.

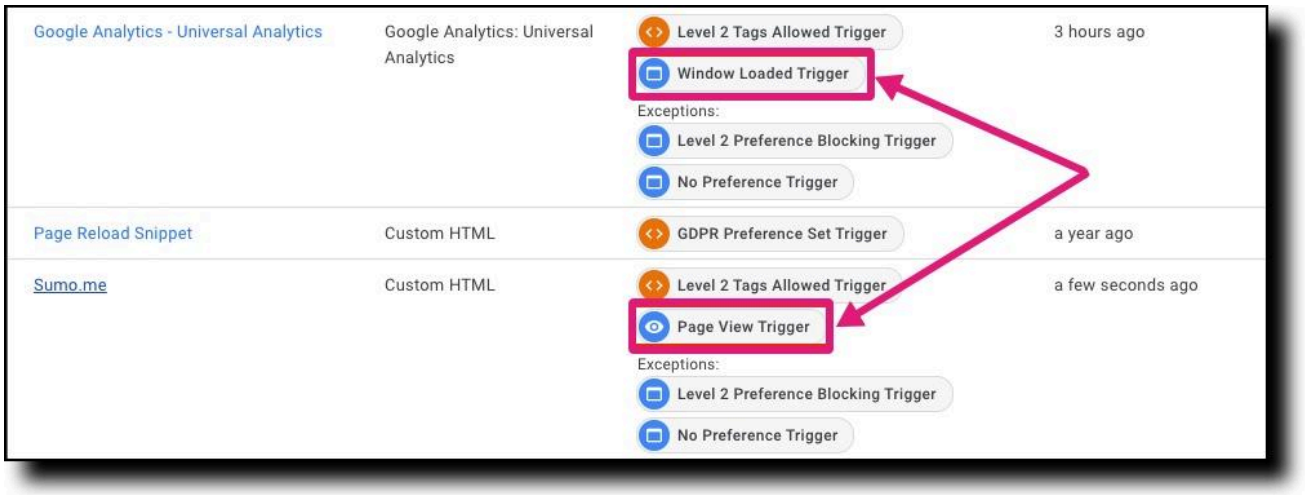
In the above snapshot the **Tracker Behavior** variable is 'undefined' and does not contain data at the *Page View* GTM Event.

A tag is not being blocked along with others containing the same blocking rule

The most common cause of this situation is that the tag uses a different GTM Event for its trigger condition (ie – *Page View* vs. *Window Loaded*). The Firing and Blocking Triggers are GTM Event specific and must match – a tag which fires on the Window Loaded GTM Event would need a corresponding Blocking Trigger configured for the Window Loaded GTM Event as well. To resolve this issue, you would need to either update the Firing Trigger of the tag not being blocked to use the GTM Event of the existing Blocking Triggers, or create a set of Blocking Triggers configured to the GTM Event of the tag not being blocked.



In the above snapshot both the Google Analytics and Sumo.me tags are Level 2 tags and contain the same blocking rule configured for the *Window Loaded* GTM Event.



In the above snapshot we see that the firing triggers are different, with the Sumo.me tag firing on the *Page View* GTM Event and subsequently not blocked by the blocking rule configured for the *Window Loaded* GTM Event.

Adobe Dynamic Tag Manager

The Cookie Consent Manager integrates with Adobe Dynamic Tag Management (DTM) in a three-step process.

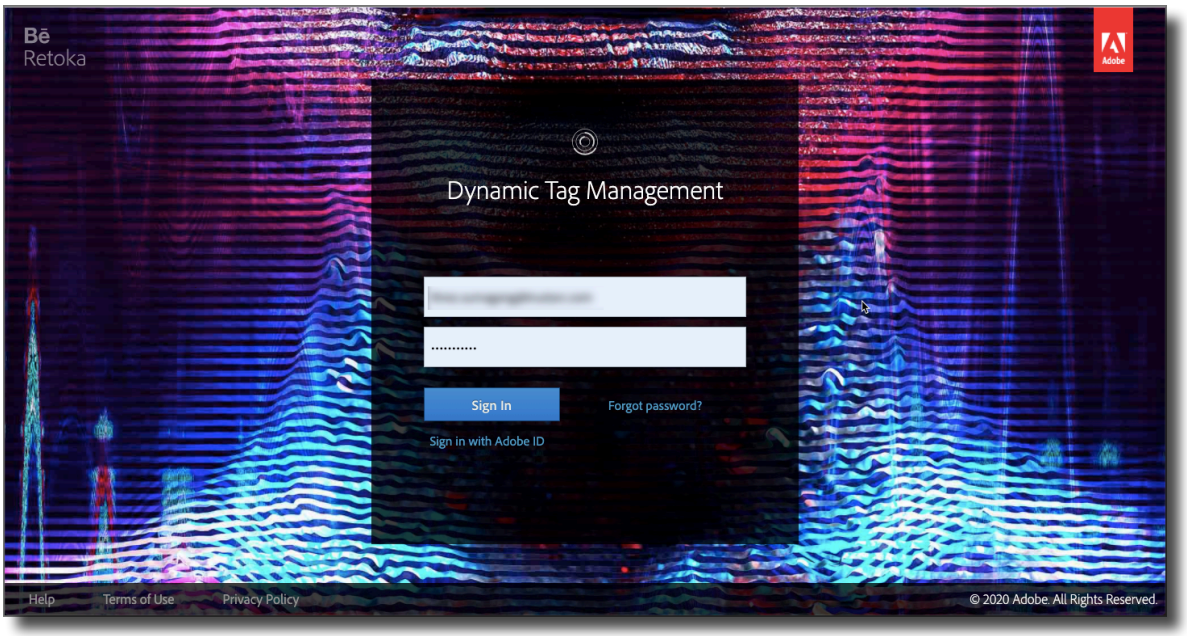
- The first step is just to add the Cookie Consent script like you would any other Third Party tag in DTM.
- The second step is optional and is commonly used for “zero-cookie load” implementations. This requires applying a special tag, which will reload the page when a user has changed their preference, thereby loading any newly allowed tags/rules.
- The third step is applying a special condition to any Rule you wish covered by the Cookie Consent.

The next sections of this document will provide details on how to accomplish these steps.

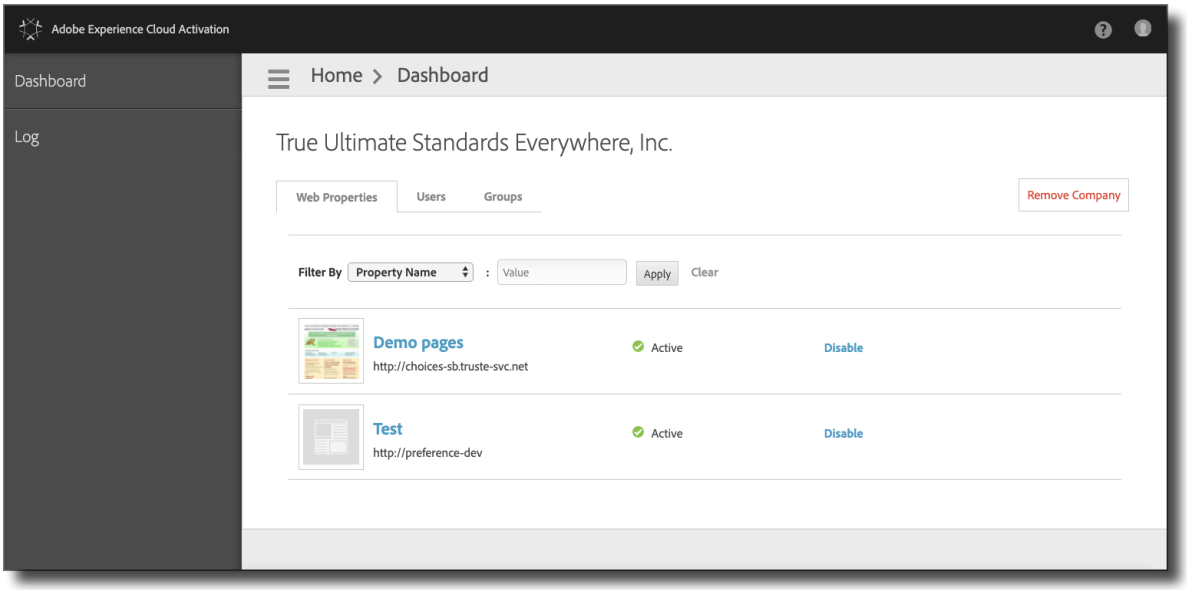
Adding the Cookie Consent Script

To add the Cookie Consent script, follow the steps accordingly.

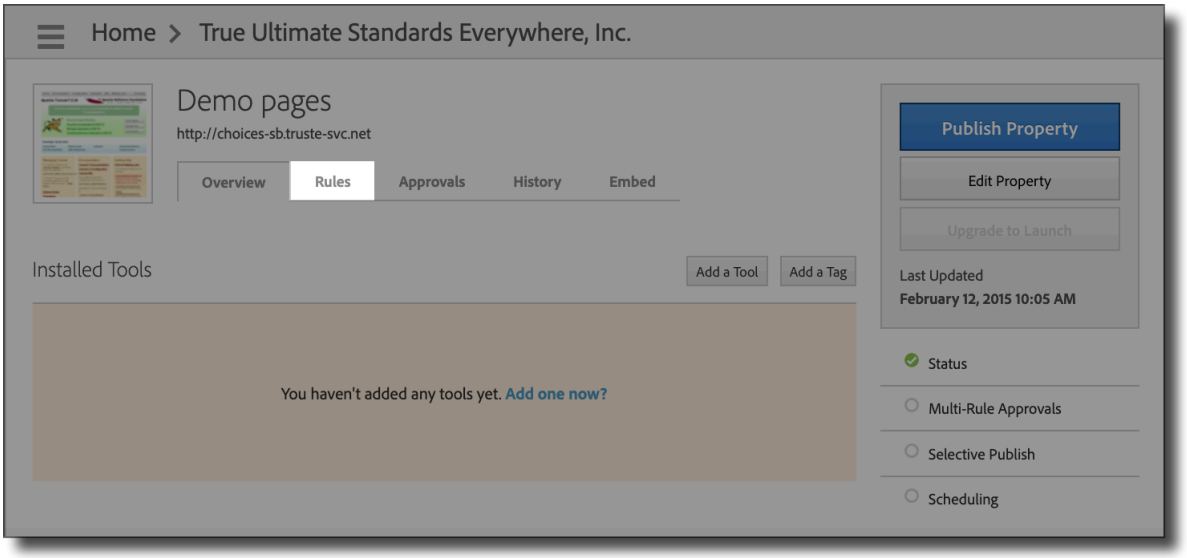
1. Log in to Adobe DTM dashboard.



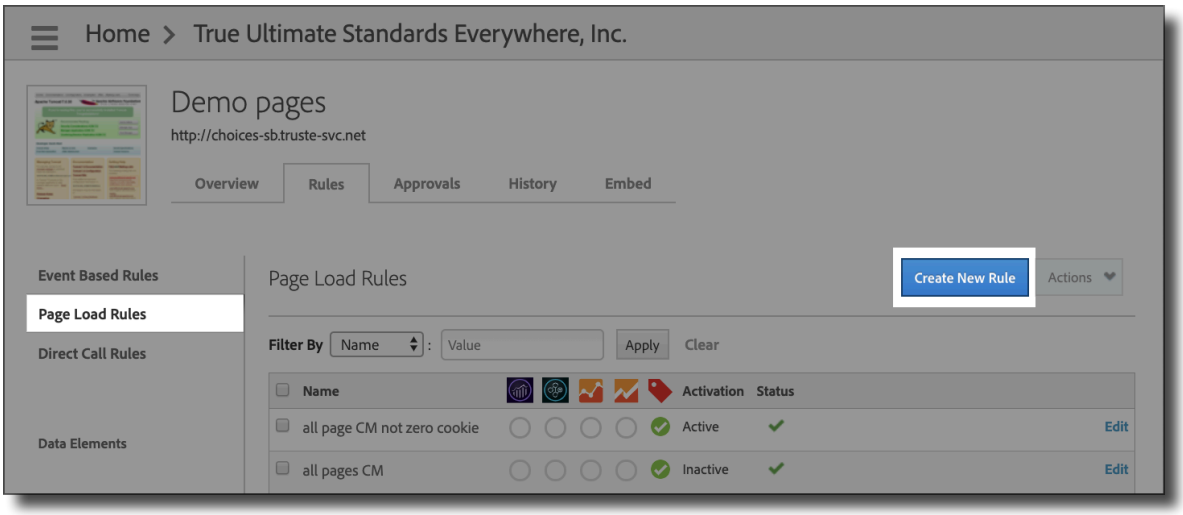
2. Under the *Web Properties* tab, select the domain you wish to add the script.



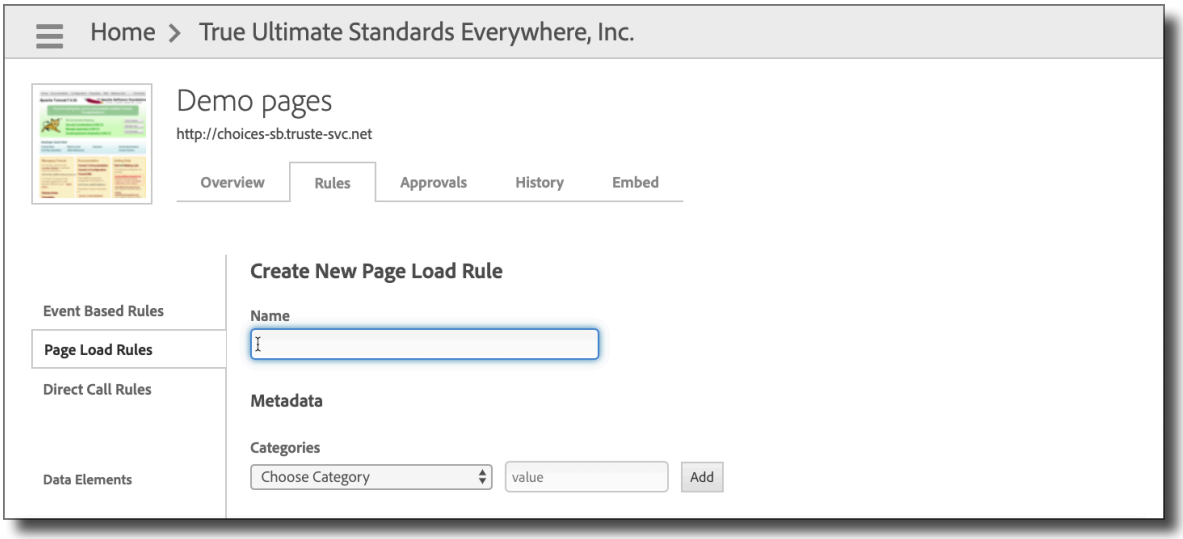
3. Click the **Rules** tab.



4. Under *Page Load Rules*, click **Create New Rule** or select an appropriate existing Rule to add the Cookie Consent tag.



5. Enter a **Name** for the *Page Load Rule* and select the desired category.

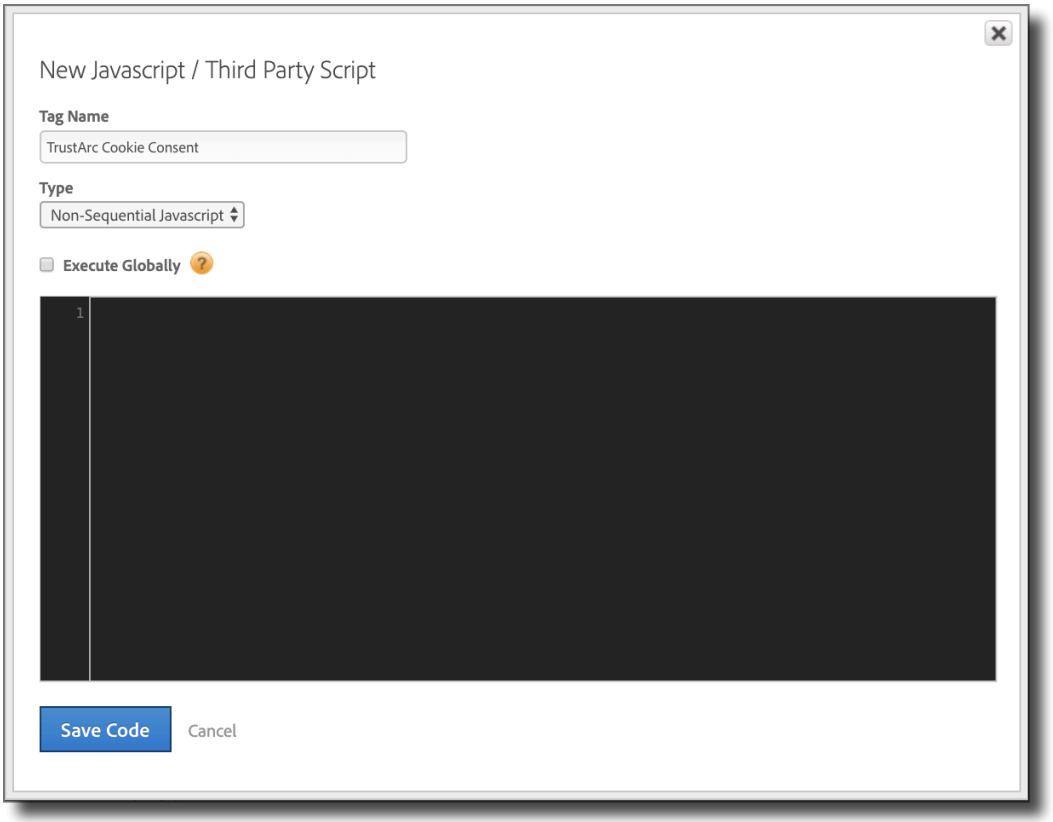


6. Under *Conditions* section, select **Top of Page** in the *Trigger rule* drop-down menu.

NOTE: This Rule can be applied at any time, but **Top of Page** is recommended since the TrustArc tag is already asynchronous.

7. Click **Javascript / Third Party Tags**.
8. With the *Non-Sequential* tab selected, click **Add New Script**.

9. Name the script **TrustArc Cookie Consent**.



10. Paste your TrustArc Cookie Consent tag in the script area via Javascript.

    For example, the code you would embed is:

    var script = document.createElement('script');
    script.src = 'https://consent.trustarc.com/v2/notice/`{cmId}`';
    document.head.appendChild(script);

    Where `{cmId}` is the value of your custom Consent Manager script.

    NOTE: Go to *Adding Page Load Script* below if you wish to use its functionality.

11. Click **Save Code**.

NOTES:

- The TrustArc Cookie Consent script by default drops opt-out cookies when the user submits preferences. In order to leverage the Adobe DTM Rules below to honor user preferences, the script must have the Tag Management System feature enabled in Step 4 of the Self-Service UI Configuration Wizard.
- Do not place `<div id="consent_banner"></div>` and `<div id="teconsent"></div>` in non-sequential Javascript configuration. These elements need to be placed directly on the page templates where you'd like them to appear on the page.

Disclaimer: Plugins and browser extensions can block your tag managers and then, block Consent Manager. Please check if your tag manager works with the popular Adblockers. If your tag manager is blocked, you may consider firing the TrustArc scripts outside of the tag manager in the HEAD tag.

Adding Page Load Script *(Optional)*

The **Page Load Script** is a special script that reloads the page when a user sets their preferences. By reloading the page, any additional cookies the user consented to are then executed.

To add the page load script, follow these steps:

1. Go step 10 of *Adding the Cookie Consent Script* section where you pasted your TrustArc Cookie Consent tag. After the TrustArc Cookie Consent tag, paste the following code:

None

```
<script>
window.addEventListener("message", function(event) {
    var eventDataJson = null;

    // We only care about TrustArc Events at this point. And TrustArc's even it
encoded in JSON
    try {
        eventDataJson = JSON.parse(event.data);
    } catch (e) {
        // Some other event that is not JSON.
        // TrustArc encodes the data as JSON
        // console.log(event.data);
    }

    // Safeguard to make sure we are only getting events from TrustArc
    if (eventDataJson && eventDataJson.source === "preference_manager") {
        // Means that the user has submited their preferences
        if (eventDataJson.message === "submit_preferences") {
            setTimeout(function() {
                window.location.reload();
            }, 20);
        }
    }

}, false);
</script>
```

2. Make sure to add `<script>` and `</script>` to the start and end of the snippet above if using HTML instead of Javascript.

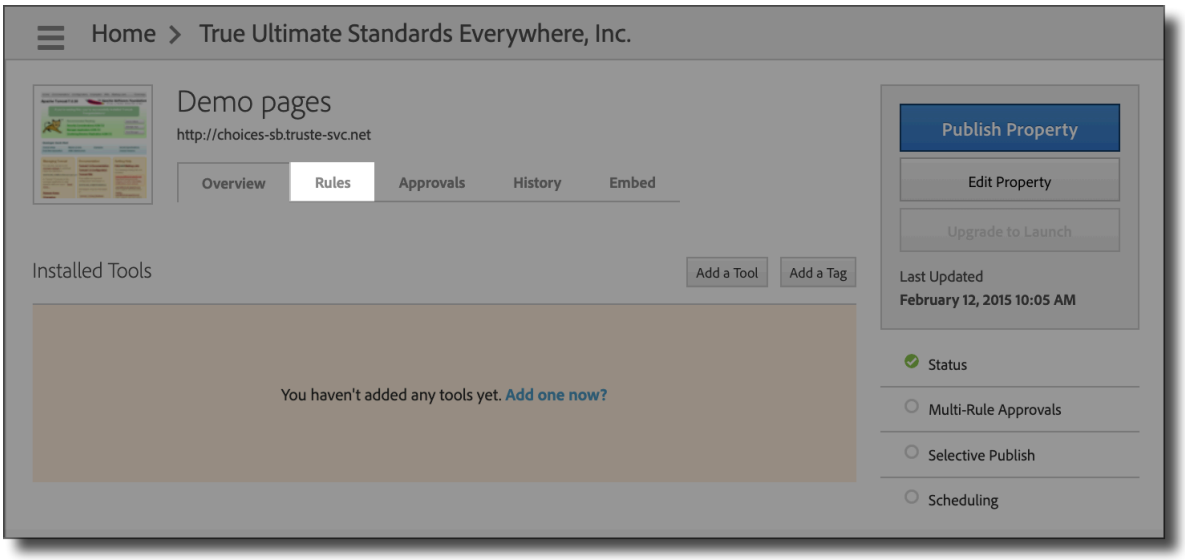
Applying New Condition to All Rules/Tags

When applying these conditions to rules/tags, note that you can only use either *Implied Consent* or *Expressed Consent,* not both. Follow the procedure outlined in this section to properly apply the conditions.

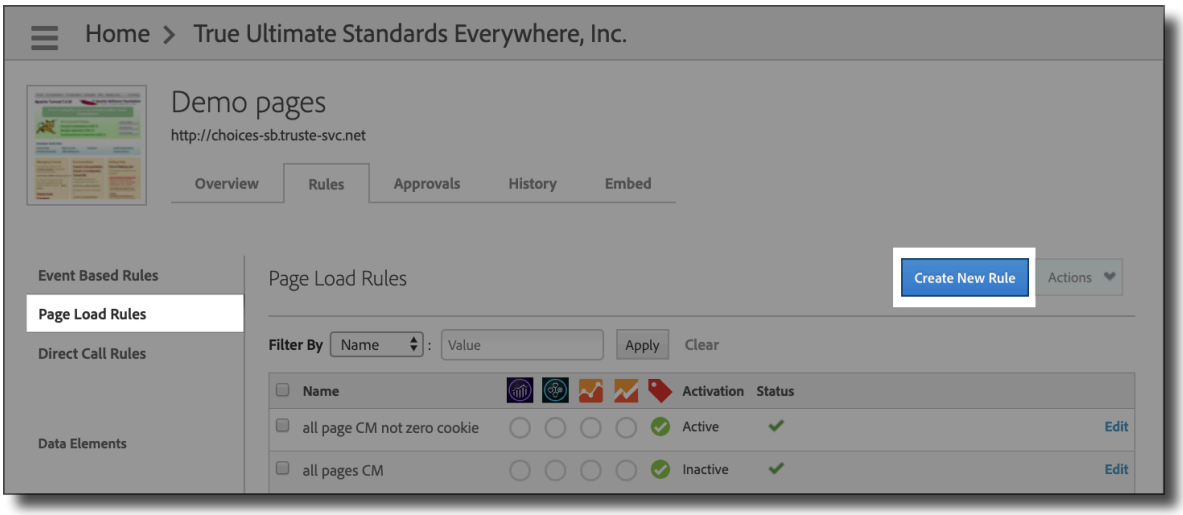
Zero Cookie Load (Expressed Consent)

For each rule/tag you wish to be blocked/allowed based upon user consent, apply the following steps.

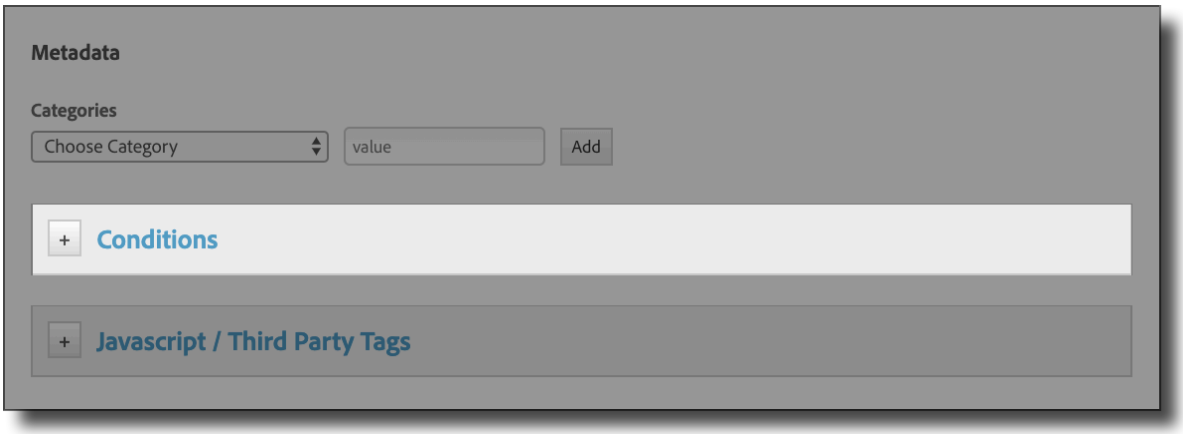
1. Click the **Rules** tab.



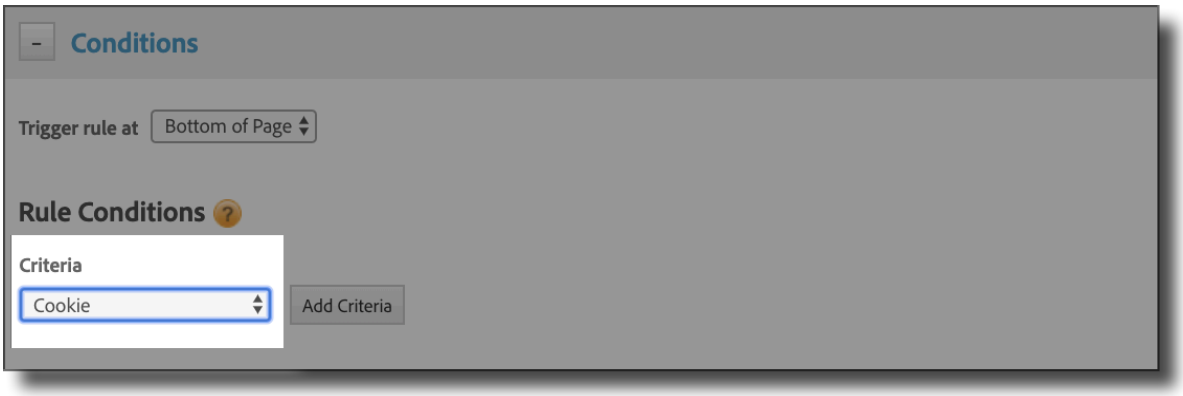
2. Under *Page Load Rules*, click **Create New Rule** or select an appropriate existing Rule.



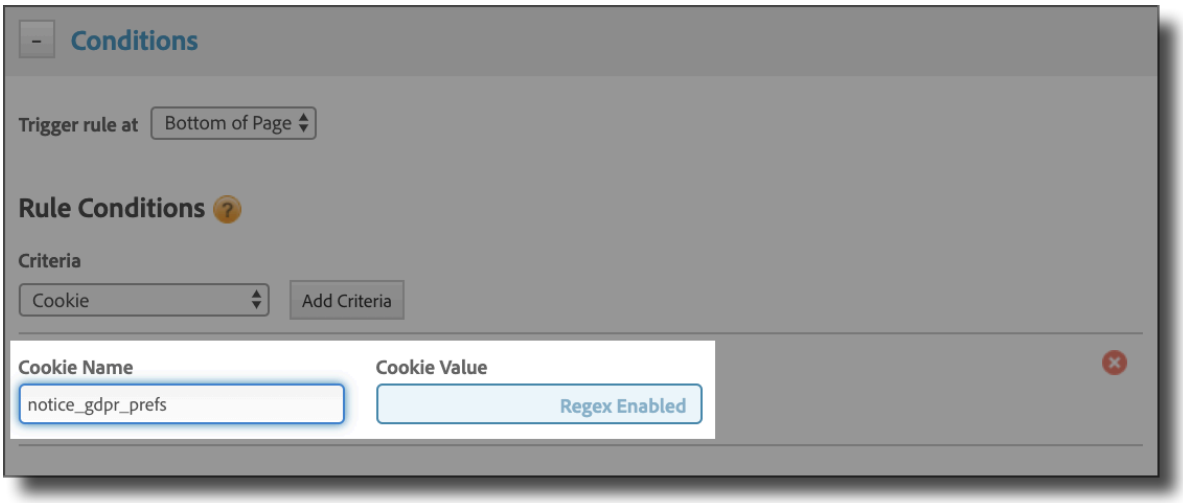
3. Under *Metadata,* click the **+** button next to add conditions.



4. In the *Criteria* list, select **Cookie**. (Trigger rule value does not matter)



5. Click **Add Criteria**.
6. In the *Cookie Name* field, enter **notice_gdpr_prefs** and enable **Regex** in the *Cookie Value* field.

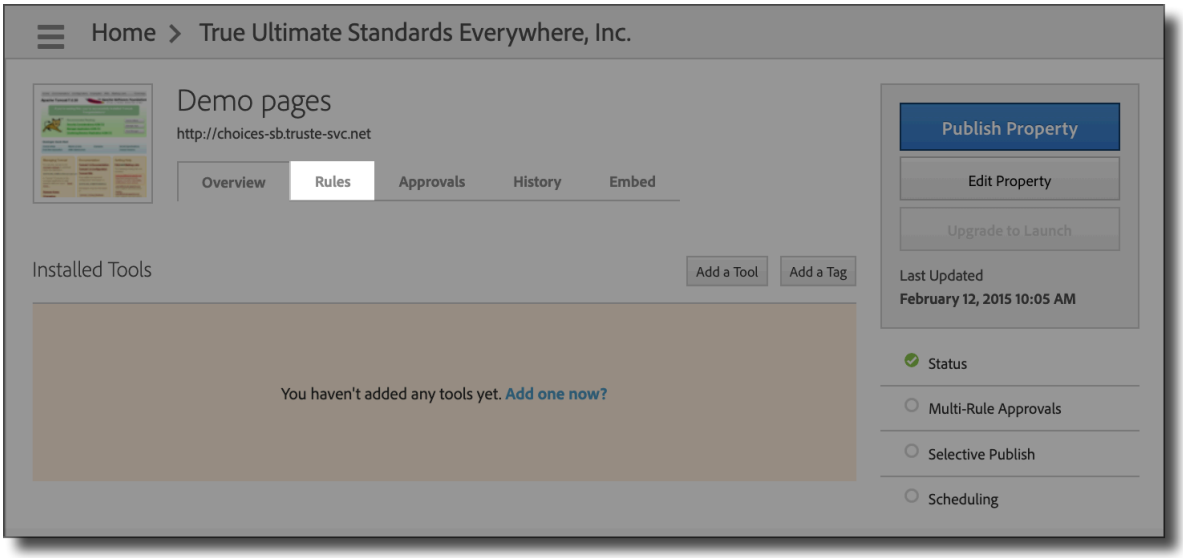


7. For the *Cookie Value*, choose one of the values listed:
    a. For Functional Tags/Rules: **^[^:]*1[^:]*:**
    b. For Advertising Tags/Rules: **^[^:]*2[^:]*:**
8. Click **Save Rule** to save the changes.

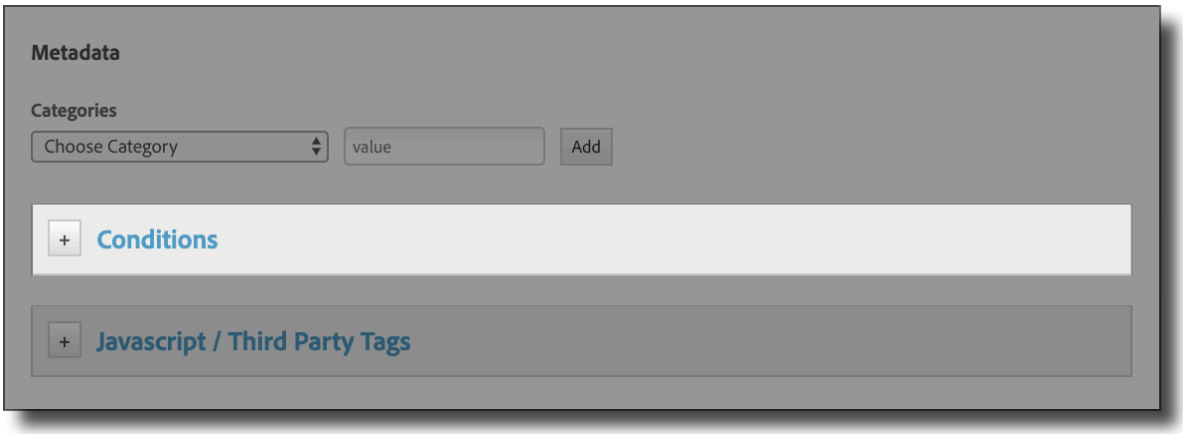
Regular Load (Implied Consent)

For each rule/tag you wish to be blocked/allowed based upon user consent, apply the following steps.

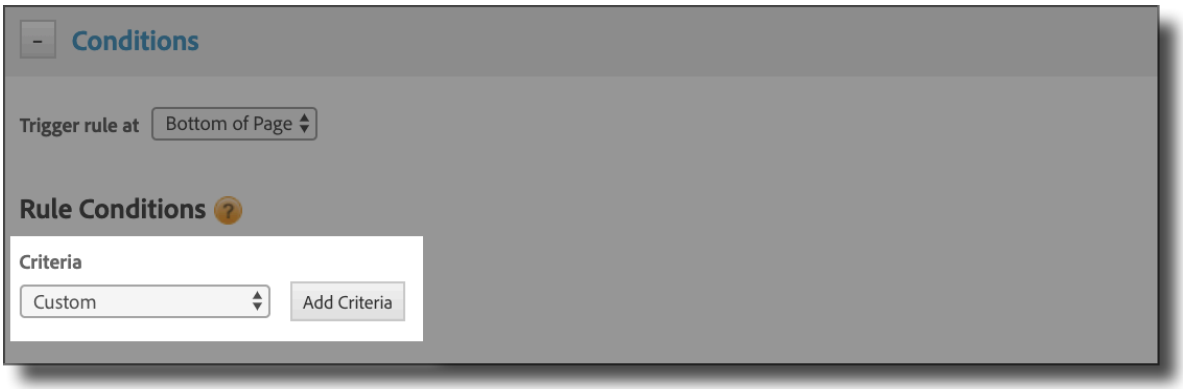
1. Click the **Rules** tab.



2. Under *Page Load Rules*, click **Create New Rule** or select an appropriate existing Rule.
3. Under *Metadata,* click the **+** button next to add conditions.



4. In the *Criteria* list, select **Custom**. (Trigger rule value does not matter)



5. Click **Add Criteria**.
6. In the custom script box, insert the following:
    a. For Functional Tags/Rules

```
return !document.cookie.match(/\bnotice_gdpr_prefs=/)
&& !document.cookie.match(/\bnotice_behavior=\w+,eu\b/)
|| document.cookie.match(/\bnotice_gdpr_prefs=[^:]*1[^:]*:/)
```

    b. For Advertising Tags/Rules

```
return !document.cookie.match(/\bnotice_gdpr_prefs=/)
&& !document.cookie.match(/\bnotice_behavior=\w+,eu\b/)
|| document.cookie.match(/\bnotice_gdpr_prefs=[^:]*2[^:]*:/)
```

Advanced: These instructions assume the common three-level consent of required/functional/advertising. If your Cookie Consent has more than these three levels, the Cookie Values listed can be extended to cover those additional steps.

Advanced additional rules: must append the next integer into rules above.

Example: "**^[^:]*3[^:]*:**" would be a 'next' level above Advertising.

This Rule will only be activated on a web page when a user has set a preference which allows this type of tag.

7. Click **Save Rule** to save the changes.

Adobe Experience Platform

The Cookie Consent Manager can integrate with Adobe Experience Platform's[2] data collection technologies (formerly Launch by Adobe[3]). This next-generation tag management capability within the Adobe Experience Cloud Platform enables clients to:

- Deploy client-side web products using integrations called extensions.
- Consistently capture, define, manage, and share data between marketing and advertising products from both Adobe and other vendors.

The extensibility and open architecture of Adobe Experience Platform's data collection technologies allow TrustArc and Adobe customers to deploy and manage Cookie Consent Manager via an extension, based on their specific needs.

Adding the Cookie Consent Manager Extension

An extension is a packaged set of code that extends the AEP interface and the library functionality. AEP is the platform and extensions are like apps that run on the platform.

NOTE: These instructions describe deploying the TrustArc Cookie Consent Manager (CCM) Script tag using an AEP extension and configuring the extension using a rule. If a more customized approach is needed such as deploying the CCM script tag directly in the page template, this can be done, but you will then not need to follow these steps as only one script tag is necessary.

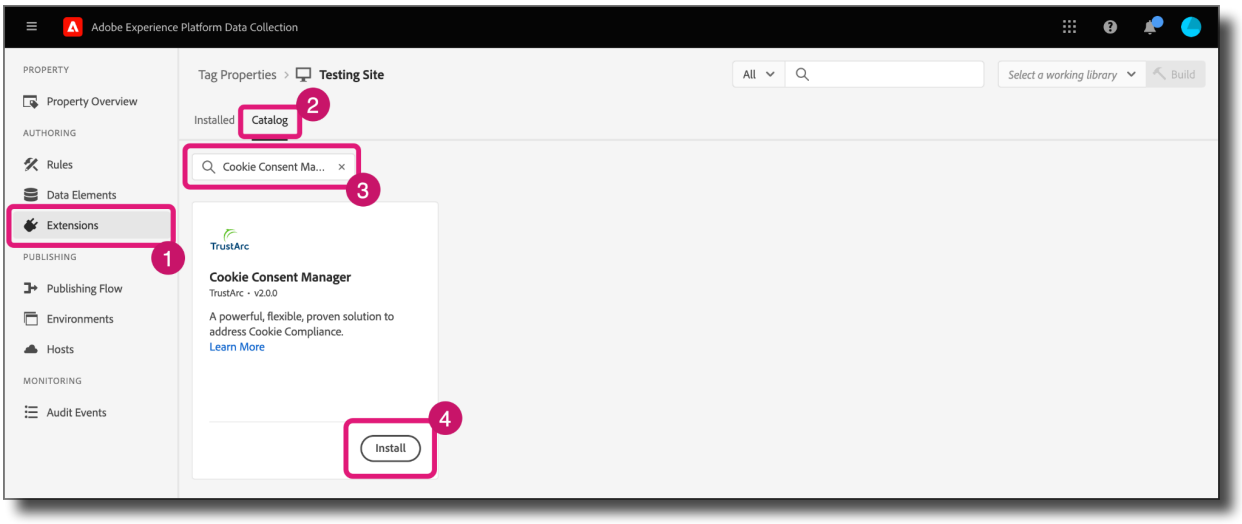
To add the Cookie Consent Manager extension, follow the steps below.

1. Log in to the *Adobe Experience Platform* dashboard.
2. Select a property from the list. A property is a container that you fill with extensions, rules, data elements, and libraries as you deploy tags to your site. A property can be any grouping of one or more domains and subdomains. If a property is not created, you can create one by clicking **New Property**.

[2] https://experienceleague.adobe.com/en/docs/experience-platform/tags/home

[3] https://experienceleague.adobe.com/en/docs/experience-platform/tags/term-updates

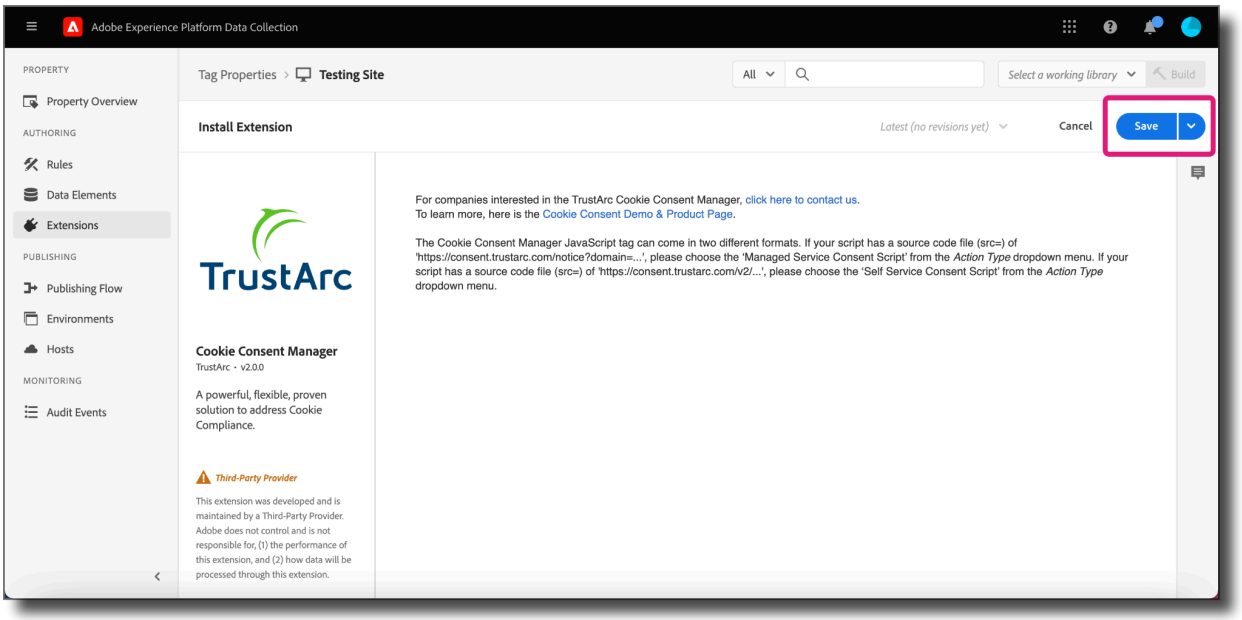
3. From the property's overview page, select the **Extensions** (1) tab.



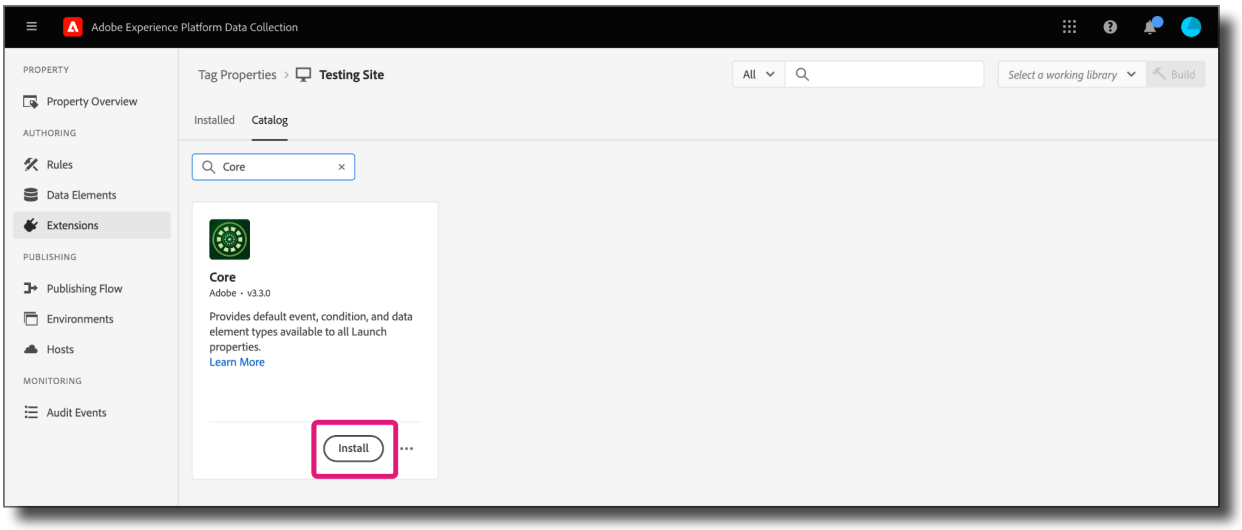
4. Select **Catalog** (2) and type **Cookie Consent Manager** (3) in the *Search* field.

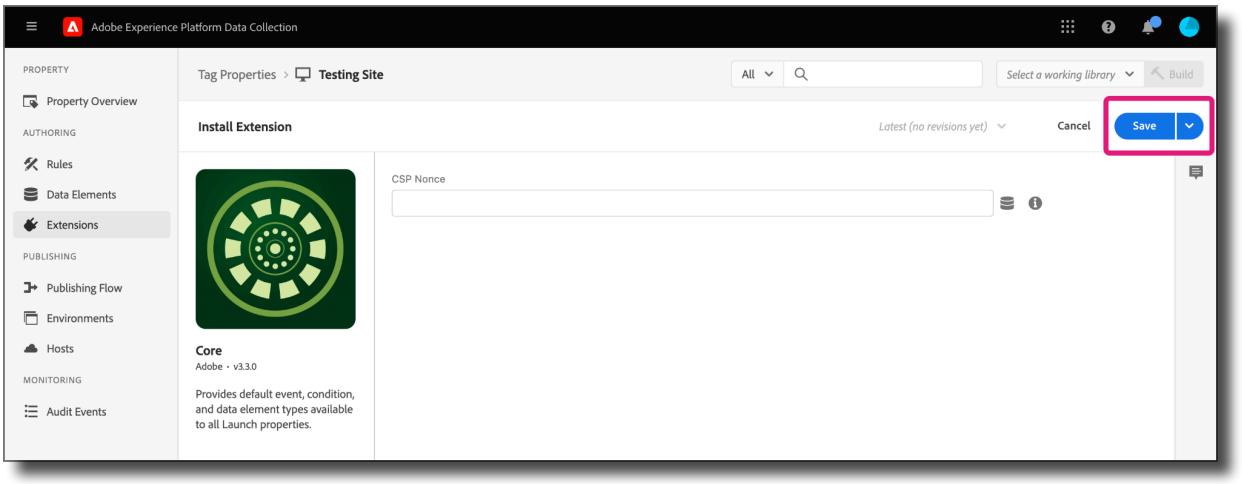
5. The search results display the Cookie Consent Manager widget. Click **Install** (4).

6. In the *Install Extension* window, click **Save**. Once saved, the Cookie Consent Manager is listed in the *Extensions* tab.



7. Install the **Core** extension, if not already installed.



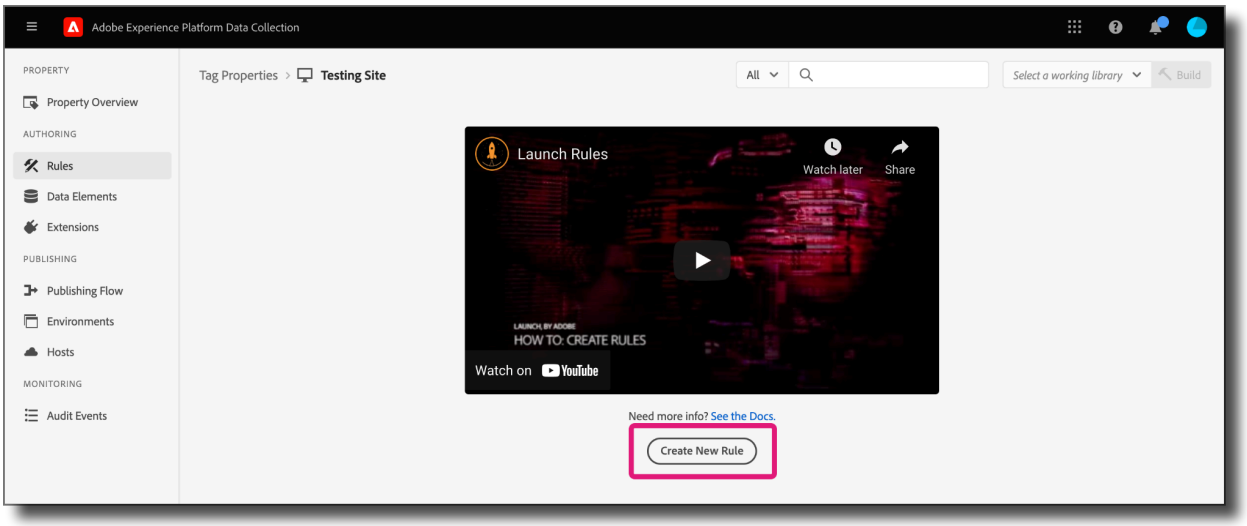


Applying Rules to the TrustArc Cookie Consent Manager

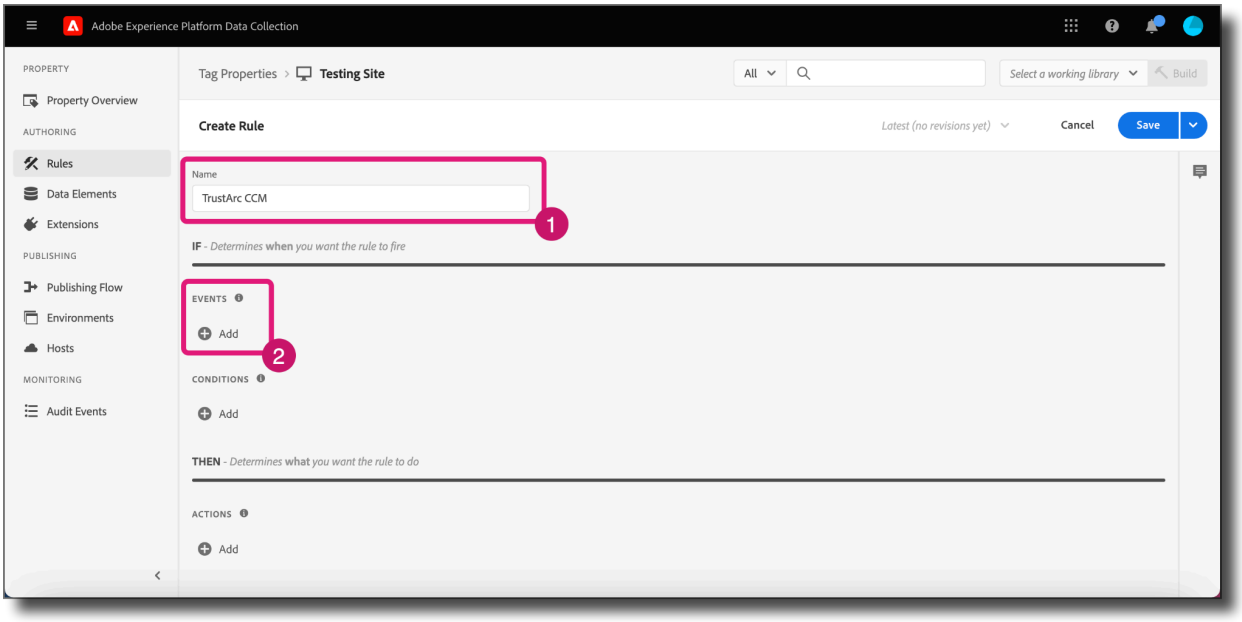
Rules are used to set up the conditions and parameters for your extensions. When the criteria outlined in your rules are met, the rule triggers the CCM extension. The following rules must be set up in Adobe Experience Platform for your Cookie Consent Manager.

IMPORTANT: The TrustArc tag should be loaded at an early event to ensure that the `notice_behavior` cookie can drop before the influenced trackers attempt to fire. This cookie is used to determine firing rules based on whether the end-user is from a location configured as zero tracker load or not. To add the TrustArc tag and apply the rules, follow the steps below.

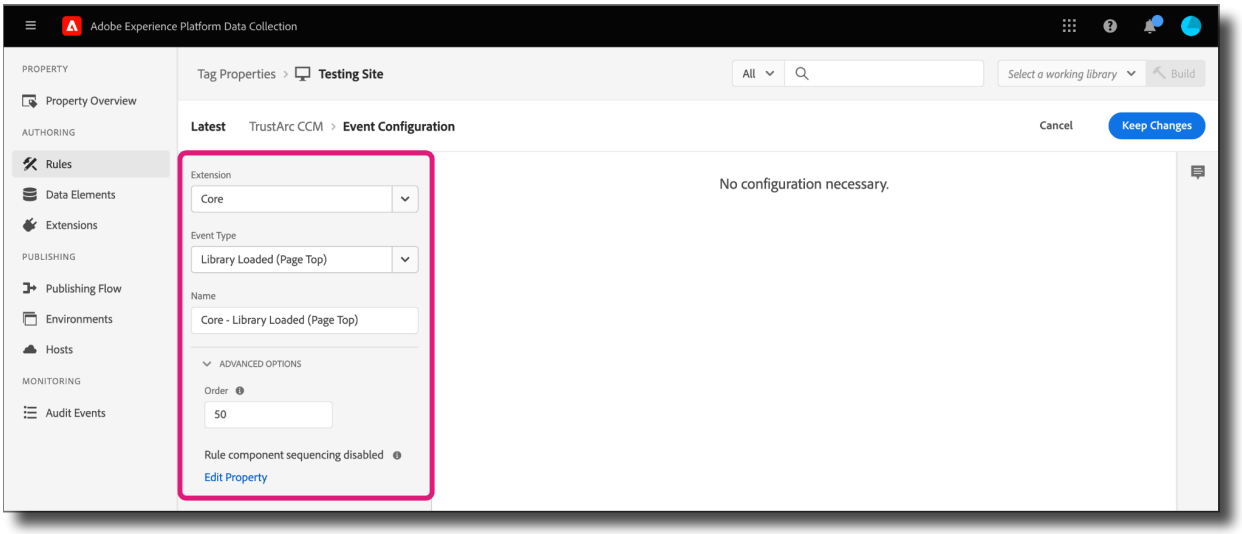
1. From the list of properties, select a **Property** where you want the rule to be applied.
2. From the *Rules* tab, click **Add Rule** or **Create New Rule** if this is your first time creating a rule.



3. Enter a **Name** (2) for the rule, for example, *TrustArc CCM*.
4. Click the **Add** (3) icon under *Events*.



5. Enter the following values in each of the given fields:

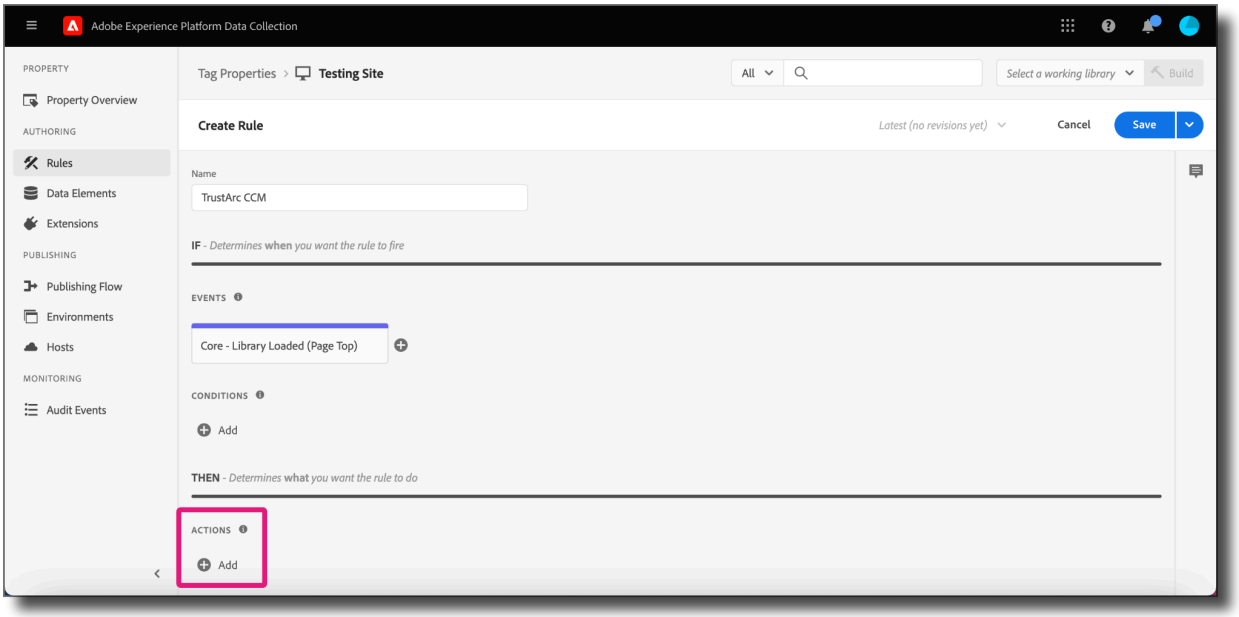


- **Extension**: *Core*
- **Event type**: *Library Loaded (Page Top)*

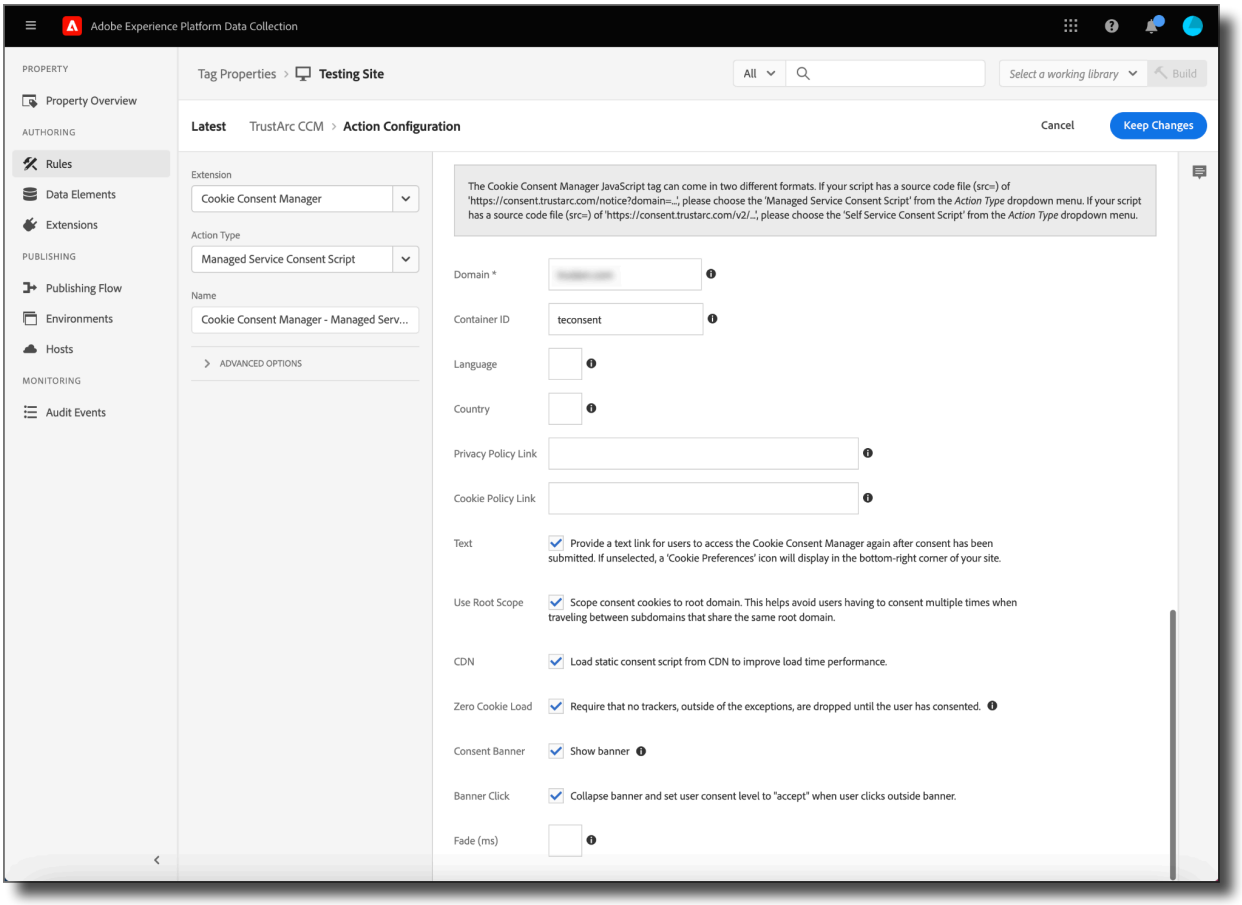
  This ensures that the TrustArc tag fires first before anything else.
- **Name**: Provide a new event **Name** or leave it as the default.
- **Order**: Set a new **Order** value or leave it as the default.

  NOTE: The default order for all rule components is **50**. If two rules have the same event type, the one with the lowest number runs first.

6. Click **Keep Changes**. This sets the event you want the rule to look for before firing.
7. Click the **Add** icon under *Actions*.



8. Enter the following values in each of the given fields:



- **Extension**: *Cookie Consent Manager*
- **Action Type**: *Managed Service Consent Script*
- **Name**: Provide a new **Name** for this action or leave it as the default.
- **Domain**: Enter the **Domain** in the field provided. This is a required identifier provided by your TrustArc Technical Account Manager.
- **Container**: The **Container ID** should match the element used on the page.

  NOTE: It is recommended to use the default ID `teconsent`.

9. Leave the **Language**, **Country**, **Privacy Policy Link**, and **Cookie Policy Link** fields as is.
10. It is recommended to select from the following checkboxes by default:

- Text
- Use Root Scope
- Zero Cookie Load
- Consent Banner

NOTE: Due to a feature change, we now always use CDN. We suggest that you do not enable this option because it is now a legacy preference.

11. Click **Keep Changes**. The actions are performed once an event is triggered and all conditions and exceptions are evaluated.

12. Click **Save** to create the rule.

Adding a Page Reload Script

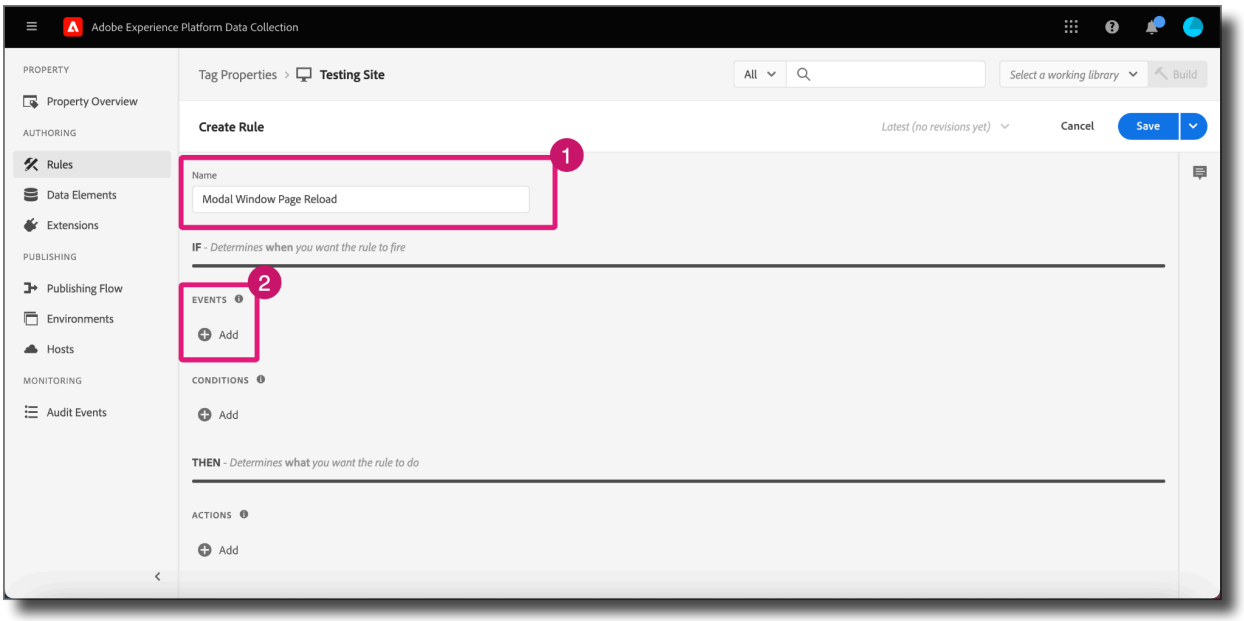
The Page Reload Script ensures that when a user updates their consent preferences, the page automatically reloads. This refresh is essential for immediately removing any cookies or trackers that the user has declined, ensuring that their privacy choices are fully applied and respected.To add the page load script, follow these steps:

1. From the list of properties, select a **Property** where you want the rule to be applied.

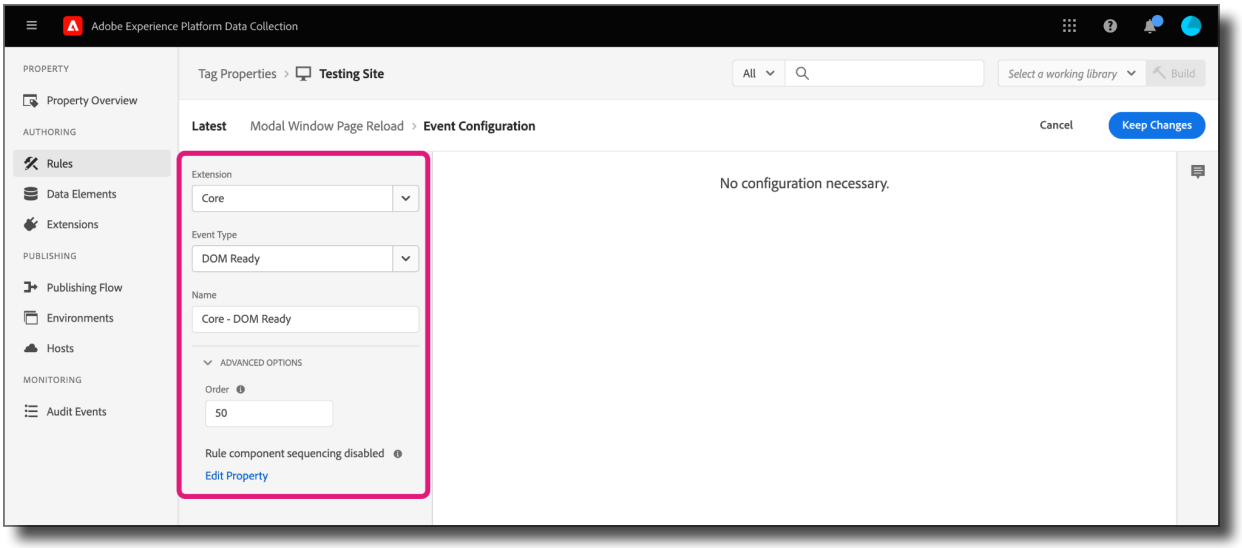
2. From the *Rules* tab, click **Add Rule**.

3. Enter a **Name** (1) for the load script, such as *Modal Window Page Reload*.

4. Click the **Add** (2) icon under *Events*.



5. Enter the following values on each of the given fields:

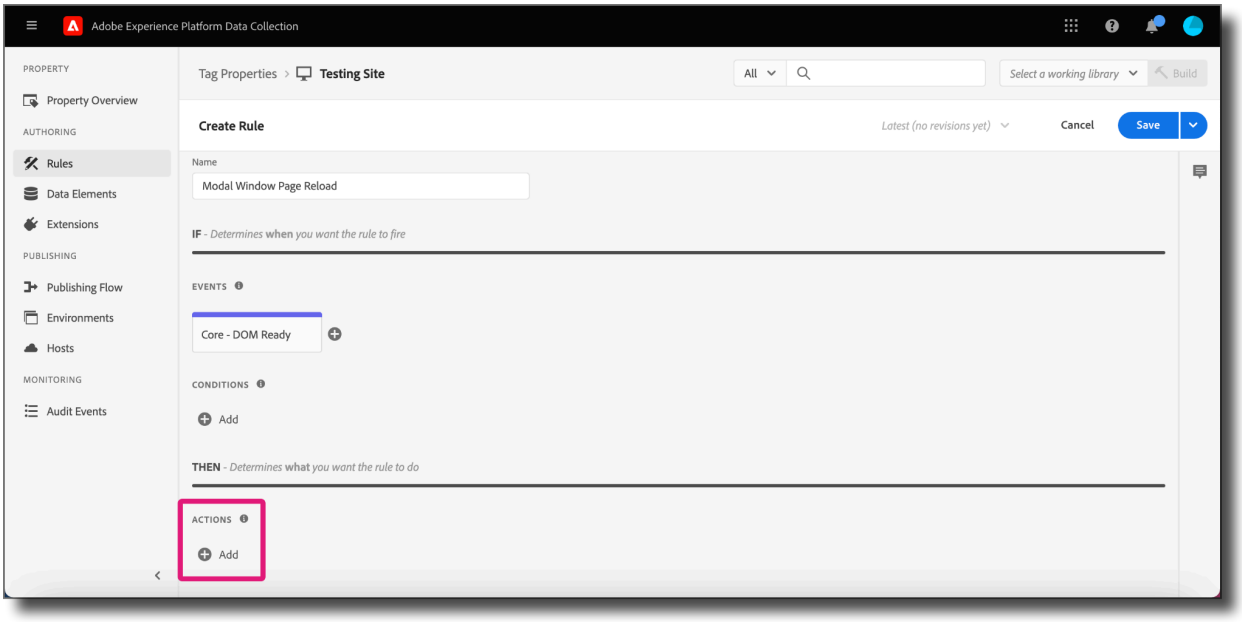


- **Extension**: *Core*
- **Event Type**: *DOM Ready*[4]
- **Name**: Provide a new event **Name** or leave it as the default.
- **Order**: Set a new **Order** value or leave it as the default.

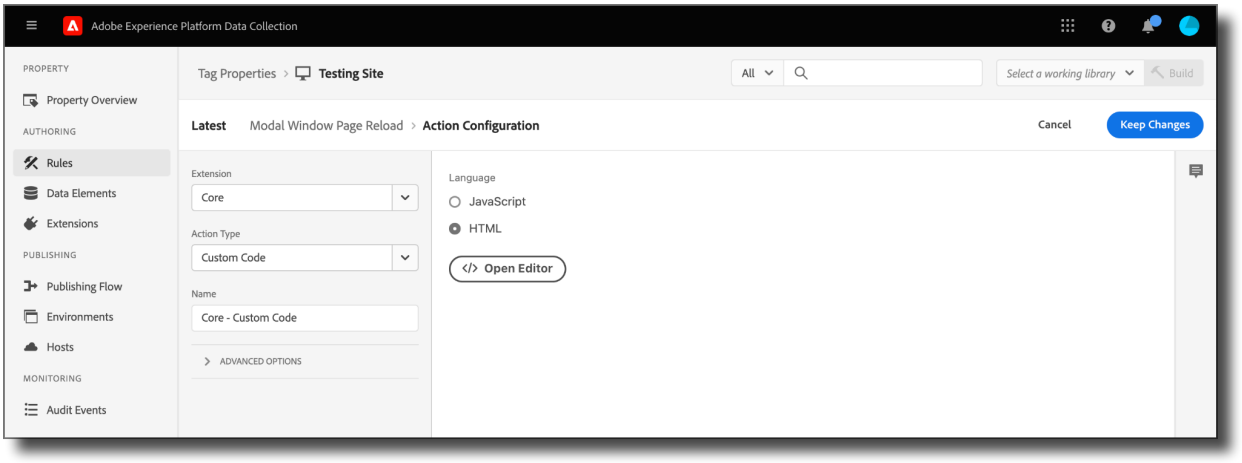
  NOTE: The default order for all rule components is **50**. If two rules have the same event type, the one with the lowest number runs first.

6. Click **Keep Changes**.
7. Click the **Add** icon under *Actions*.

[4] A trigger type that fires after the browser has finished constructing the full page in HTML and the Document Object Model (DOM) is ready to be parsed.



8. Enter the following values on each of the given fields:



- **Extension**: *Core*
- **Action Type**: *Custom Code*
- **Name**: *Enter a new name for this action or leave it as the default.*
- **Language**: Select *HTML*

9. Click **</> Open Editor** and enter the following code:

JavaScript

```
<script>
window.addEventListener("message", function(event) {
    var eventDataJson = null;

    // We only care about TrustArc Events at this point. And TrustArc's even it
encoded in JSON
    try {
        eventDataJson = JSON.parse(event.data);
    } catch (e) {
        // Some other event that is not JSON.
        // TrustArc encodes the data as JSON
        // console.log(event.data);
    }

    // Safeguard to make sure we are only getting events from TrustArc
    if (eventDataJson && eventDataJson.source === "preference_manager") {
        // Means that the user has submited their preferences
        if (eventDataJson.message === "submit_preferences") {
            setTimeout(function() {
                window.location.reload();
            }, 20);
        }
    }

}, false);
</script>
```

NOTE: You can also copy the code from:
https://gist.github.com/trustarctam/0da8eef5ffa95c49de677e16e0f6ec68

10. Click **Keep Changes**. The actions are performed once an event is triggered and all conditions and exceptions are evaluated.

11. Click **Save** to save the Page Reload script.

TrustArc Events and Conditions

This section explains how to set up your Data Elements, Events, and Conditions to properly manage rules in Adobe Experience Manager.

Data Elements

The following Data Elements are used to determine whether a rule should be injected into the page. You will need to create each Data Element within your AEM Property.

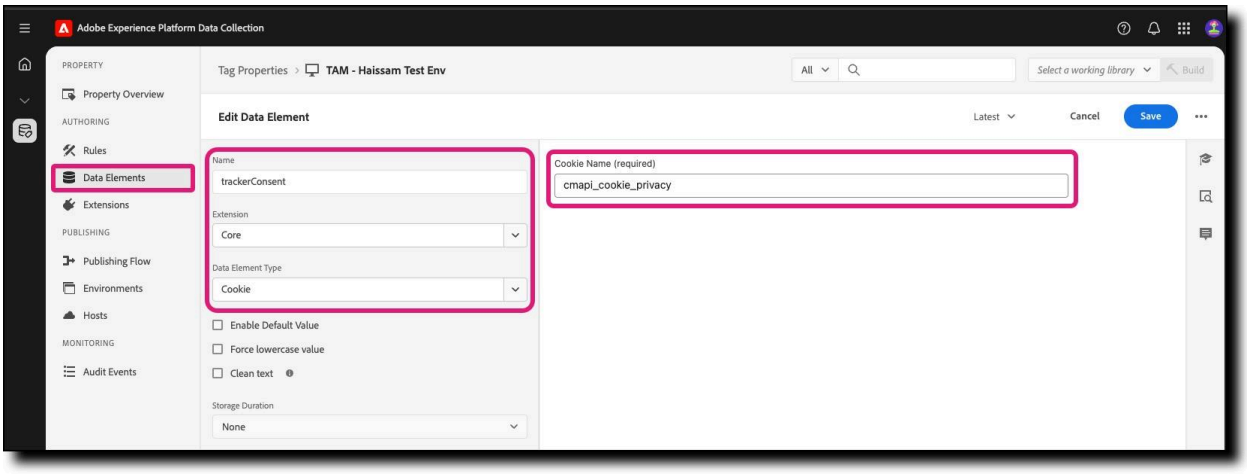
trackerConsent

This Data Element is used to identify if the user's consent is present or to identify the level of consent the user provided. This Data Element uses the value of the `cmapi_cookie_privacy` cookie, which has the following values:

- **1** – refers to preference selections for *Required* (always present as the user cannot opt out of Required)
- **2** – refers to preference selections for *Functional*
- **3** – refers to preference selections for *Advertising*
- **4+** – refers to preference selections for *custom buckets*

To configure this Data Element, follow the steps below:

1. In the *Data Elements* tab, click **Add Data Element**.
2. Enter the following information in each given fields:



- **Name**: *trackerConsent*
- **Extension**: *Core*
- **Data Element Type**: *Cookie*
- **Cookie Name**: *cmapi_cookie_privacy*

3. Click **Save**.

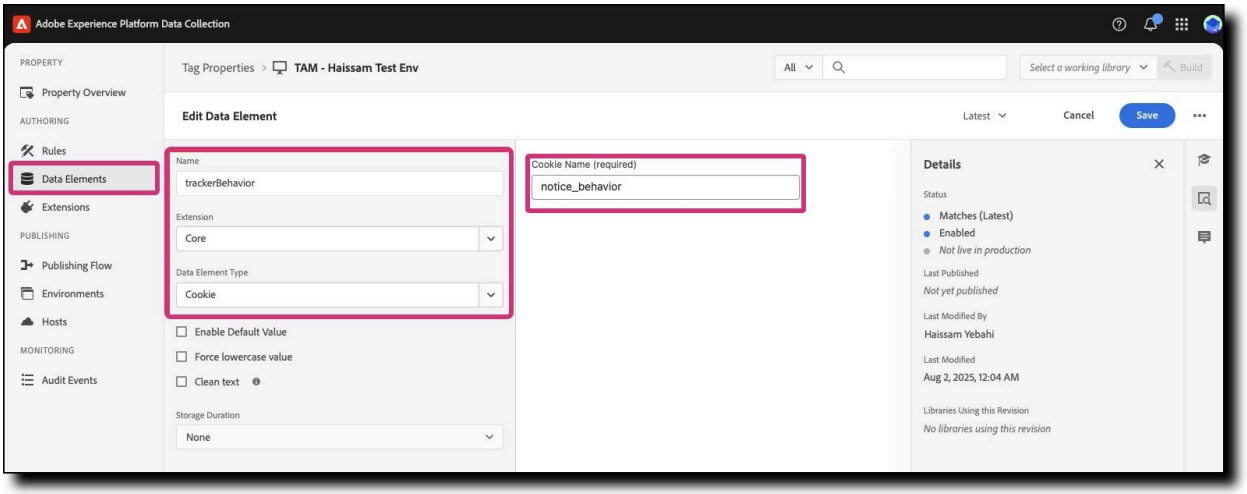
trackerBehavior

This Data Element uses the value of the `notice_behavior` cookie to identify the behavior (**Expressed** or **Implied**) and which consent manager is being used (**EU** or **US**) in the Consent Manager configuration, based on the country or state from which the user is connecting. Below are the possible values for this cookie:

- expressed,eu
- expressed,us
- implied,eu
- implied,us
- none

To configure this Data Element, you may follow the steps below:

1. In the *Data Elements* tab, click **Add Data Element**.
2. Enter the following information in each given fields:



- **Name:** *trackerBehavior*

- **Extension:** *Core*
- **Data Element Type:** *Cookie*
- **Cookie Name:** *notice_behavior*

3. Click **Save**.

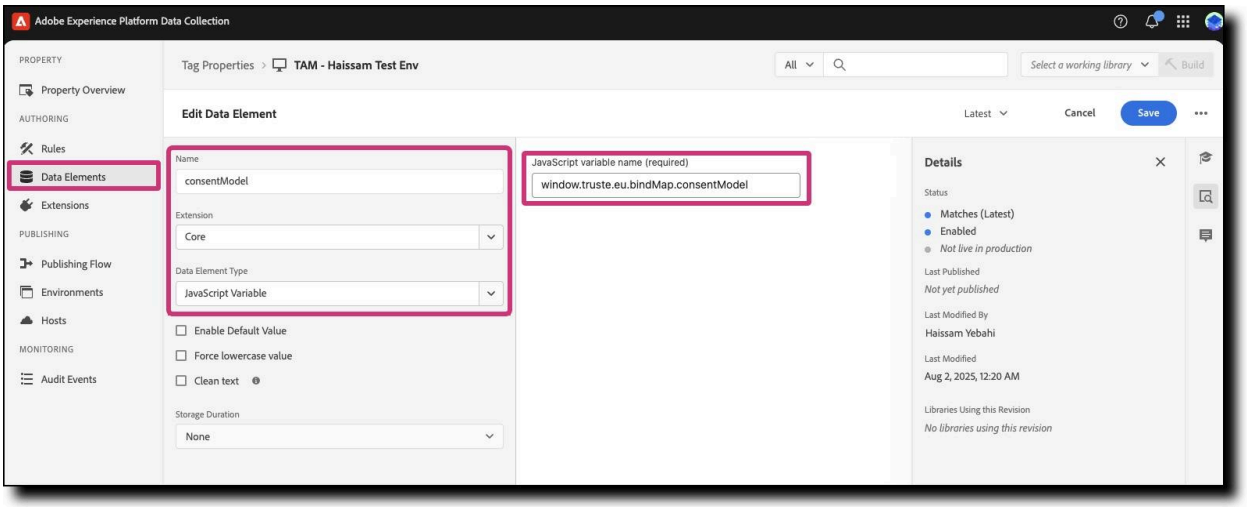
consentModel

This Data Element is using the value of one of our API keys, which is the *Consent Model*. This key has the value **'opt-in'** or **'opt-out,'** which you can use to identify whether the experience should be an opt-in or opt-out experience based on the Consent Manager configuration per country or state. You can use this Data Element instead of the trackerBehavior.

NOTE: Contact your Technical Account Manager to assist you in reviewing these settings.

To configure this Data Element, you may follow the steps below:

1. In the *Data Elements* tab, click **Add Data Element**.
2. Enter the following information in each given fields:



- **Name**: *consentModel*
- **Extension**: *Core*
- **Data Element Type**: *JavaScript Variable*
- **JavaScript variable name**: *window.truste.eu.bindMap.consentModel*

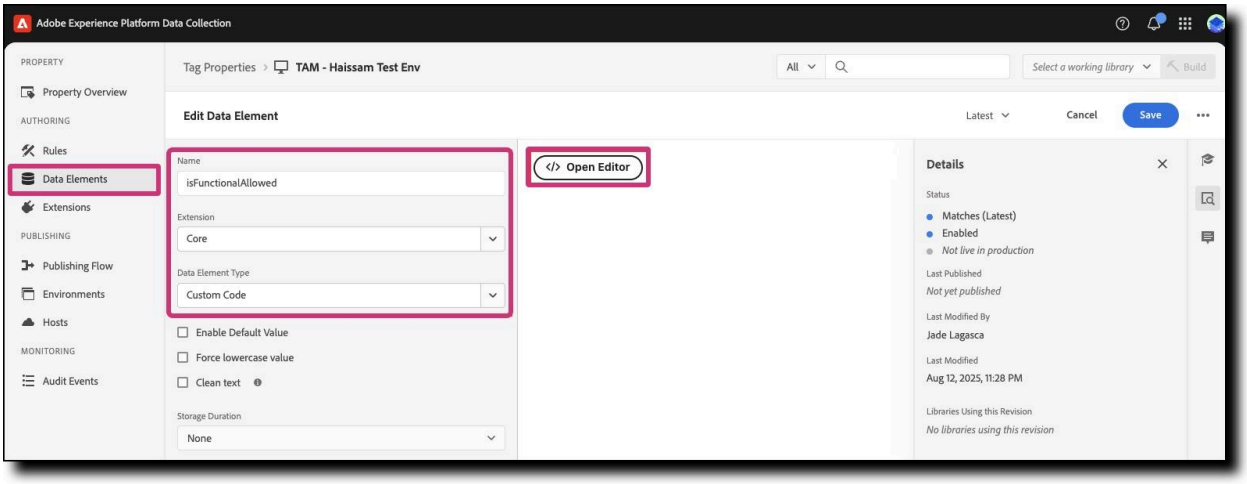
3. Click **Save**.

isFunctionalAllowed

In this Data Element, you add a custom code that will be used in a rule condition to check whether Functional Trackers are allowed according to the user's consent. The function that will return the boolean values **'true'** or **'false.'**

To configure this Data Element, you may follow the steps below:

1. In the *Data Elements* tab, click **Add Data Element**.
2. Enter the following information in each given fields:



   - **Name**: *isFunctionalAllowed*
   - **Extension**: *Core*
   - **Data Element Type**: *Custom Code*
3. Click **Open Editor**.
4. Enter the code below based on the setup that you want to follow:
   - **Option 1**: Using **trackerBehavior** and **trackerConsent**

IMPORTANT: The code snippet needs to be adjusted according to your **region-based** Consent Manager Experience. The snippet below provides several options, and we strongly recommend testing to ensure that the website visitor's preferences are being honored correctly. This includes verifying that tags and cookies are properly blocked or removed when consent is not given. Please feel free to modify the code as needed to meet your specific requirements.

JavaScript

```
var pref = _satellite.getVar('trackerConsent');
var experience = _satellite.getVar('trackerBehavior');
var isfunctionalallowed = false;


/*This is the code you can use if you are using notice_behavior for Zero-Tracker Load
(expressed behavior)*/
isfunctionalallowed = (pref == undefined &&
experience.match(/\b[^;]*\bexpressed\b[^;]*/)) || (pref != undefined &&
pref.match(/^permit\s+(?:\d+,)*2(?:,\d+)*$/));


/*This is the code you can use if you are using notice_behavior for Zero-Tracker Load
(using EU Manager only)*/
//isfunctionalallowed =  (pref == undefined && experience.match(/\b[^;]*\beu\b/)) ||
(pref != undefined && pref.match(/^permit\s+(?:\d+,)*2(?:,\d+)*$/));


/*This is the code you can use if the CM configuration is Implied or Non-ZTL (All
Countries)*/
//isfunctionalallowed =  ((pref == undefined) || (pref != undefined &&
pref.match(/^permit\s+(?:\d+,)*2(?:,\d+)*$/)));


/*This is the code you can use if you are using notice_behavior to identify if it's
Zero-Tracker or Standard Load using (EU and US value)*/
//isfunctionalallowed =  (pref == undefined && experience.match(/\b[^;]*\beu\b/)) ||
(pref != undefined && pref.match(/^permit\s+(?:\d+,)*2(?:,\d+)*$/)) || (pref ==
undefined && experience.match(/\b[^;]*\bus\b/));


return !!isfunctionalallowed;
```

- **Option 2**: Using **consentModel** and **trackerConsent**

IMPORTANT: This option allows for more granular configuration by **country, US state, or Canadian province**. Adjust the code snippet according to your Consent Manager Experience. The snippet below provides several options, and we strongly recommend testing to ensure that website visitor preferences are being honored correctly. This includes confirming that tags and cookies are properly blocked or removed when consent is not granted. Please feel free to modify the code as needed to meet your specific requirements.

```javascript
var pref = _satellite.getVar('trackerConsent');
var experience = _satellite.getVar('consentModel');

var isfunctionalallowed = false;

/*This is the code you can use if you are using consentModel for Zero-Tracker Load (All Countries)*/
isfunctionalallowed =  (pref == undefined && experience == 'opt-out') || (pref != undefined && pref.match(/^permit\s+(?:\d+,)*2(?:,\d+)*$/));

return !!isfunctionalallowed;
```

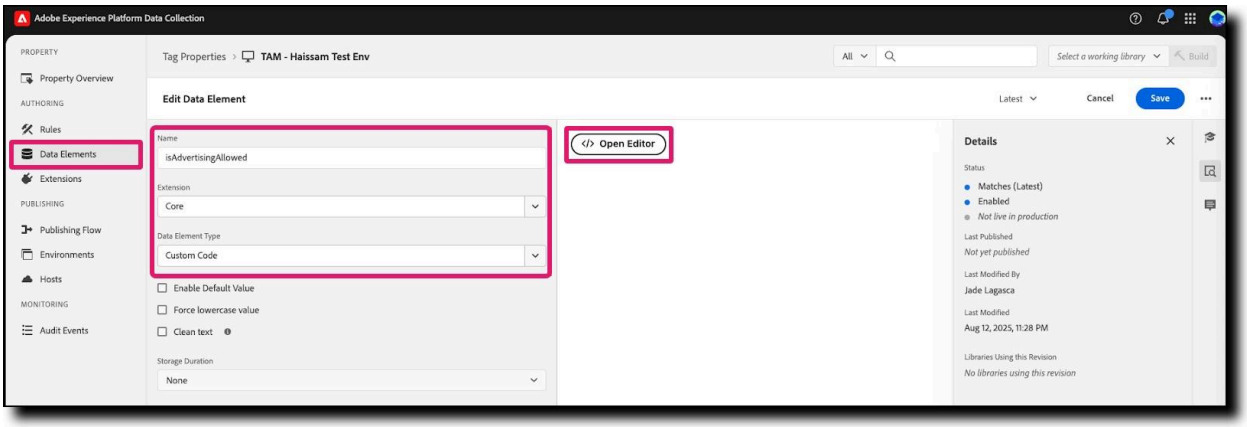
5. Click **Save**.

isAdvertisingAllowed

In this Data Element, you add a custom code to check other Data Elements if it allows Functional Cookies, which will return the value **'true'** or **'false.'**

To configure this Data Element, follow the steps below:

1. In the *Data Elements* tab, click **Add Data Element**.
2. Enter the following information in each given fields:



- **Name**: *isFunctionalAllowed*
- **Extension**: *Core*
- **Data Element Type**: *Custom Code*

3. Click **Open Editor**.
4. Enter the code below based on the setup that you want to follow:
    - **Option 1**: Using **trackerBehavior** and **trackerConsent**

IMPORTANT: The code snippet needs to be adjusted according to your **region-based** Consent Manager Experience. The snippet below provides several options, and we strongly recommend testing to ensure that the website visitor's preferences are being honored correctly. This includes verifying that tags and cookies are properly blocked or removed when consent is not given. Please feel free to modify the code as needed to meet your specific requirements.

```javascript
var pref = _satellite.getVar('trackerConsent');
var experience = _satellite.getVar('trackerBehavior');

var isadvertisingallowed = false;

/*This is the code you can use if you are using notice_behavior for Zero-Tracker Load
(expressed behavior)*/
isadvertisingallowed = (pref == undefined &&
experience.match(/\b[^;]*\bexpressed\b[^;]*/)) || (pref != undefined &&
pref.match(/^permit\s+(?:\d+,)*3(?:,\d+)*$/));

/*This is the code you can use if you are using notice_behavior for Zero-Tracker Load
(using EU Manager only)*/
//isadvertisingallowed =  (pref == undefined && experience.match(/\b[^;]*\beu\b/)) ||
(pref != undefined && pref.match(/^permit\s+(?:\d+,)*3(?:,\d+)*$/));

/*This is the code you can use if the CM configuration is Implied or Non-ZTL (All
Countries)*/
//isadvertisingallowed =  ((pref == undefined) || (pref != undefined &&
pref.match(/^permit\s+(?:\d+,)*3(?:,\d+)*$/)));

/*This is the code you can use if you are using notice_behavior to identify if it's
Zero-Tracker or Standard Load using (EU and US value)*/
//isadvertisingallowed =  (pref == undefined && experience.match(/\b[^;]*\beu\b/)) ||
(pref != undefined && pref.match(/^permit\s+(?:\d+,)*3(?:,\d+)*$/)) || (pref ==
undefined && experience.match(/\b[^;]*\bus\b/));

return !!isadvertisingallowed;
```

    - **Option 2**: Using **consentModel** and **trackerConsent**

IMPORTANT: This option allows for more granular configuration by **country, US state, or Canadian province**. Adjust the code snippet according to your Consent Manager Experience. The snippet below provides several options, and we strongly recommend testing to ensure that website visitor preferences are being honored correctly. This includes confirming that tags and cookies are properly blocked or removed when consent is not granted. Please feel free to modify the code as needed to meet your specific requirements.

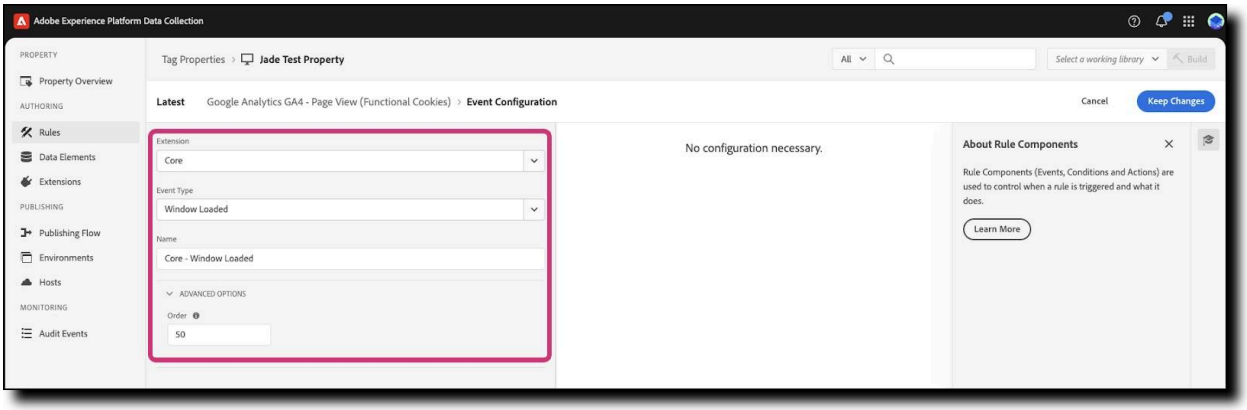
```javascript
var pref = _satellite.getVar('trackerConsent');
var experience = _satellite.getVar('consentModel');

var isadvertisingallowed = false;

/*This is the code you can use if you are using consentModel for Zero-Tracker Load (All Countries)*/
isadvertisingallowed =  (pref == undefined && experience == 'opt-out') || (pref != undefined && pref.match(/^permit\s+(?:\d+,)*3(?:,\d+)*$/));

return !!isadvertisingallowed;
```

5. Click **Save**.

Setting up a Functional Rule

Standard Event to Fire a Functional Tag

1. Click the **Add** icon again to add another *Event.*
2. Enter the following values on each of the given fields:



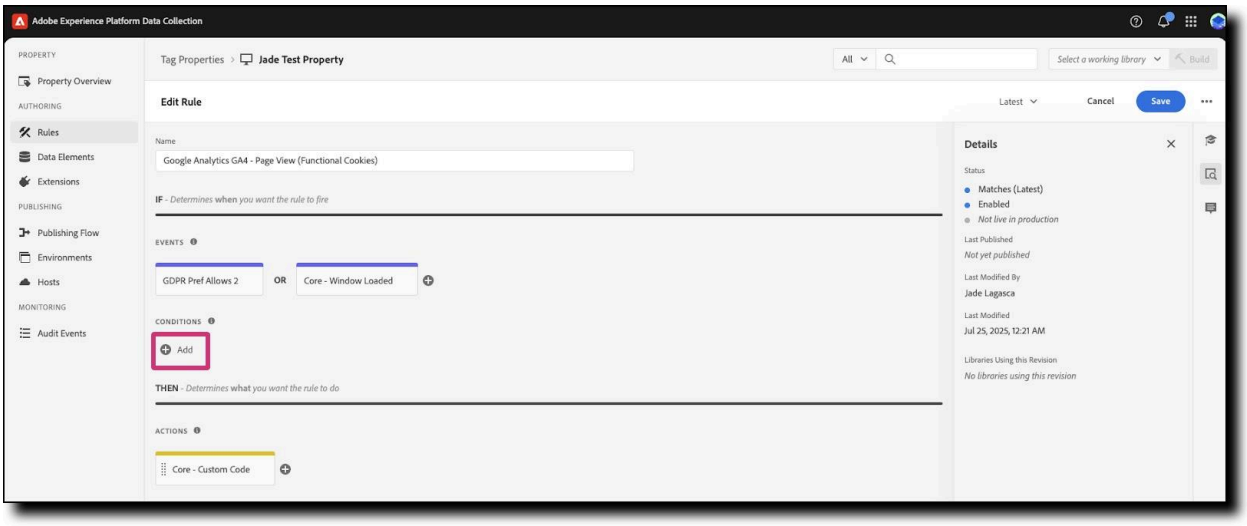
- **Extension**: *Core*
- **Event Type**: *Window Loaded*
- **Name**: Provide a new event **Name** or leave it as default.
- **Order**: Set a new **Order** value or leave it as default.

  NOTE: The default order for all rule components is **50**. If two rules have the same event type, the one with the lowest number runs first.

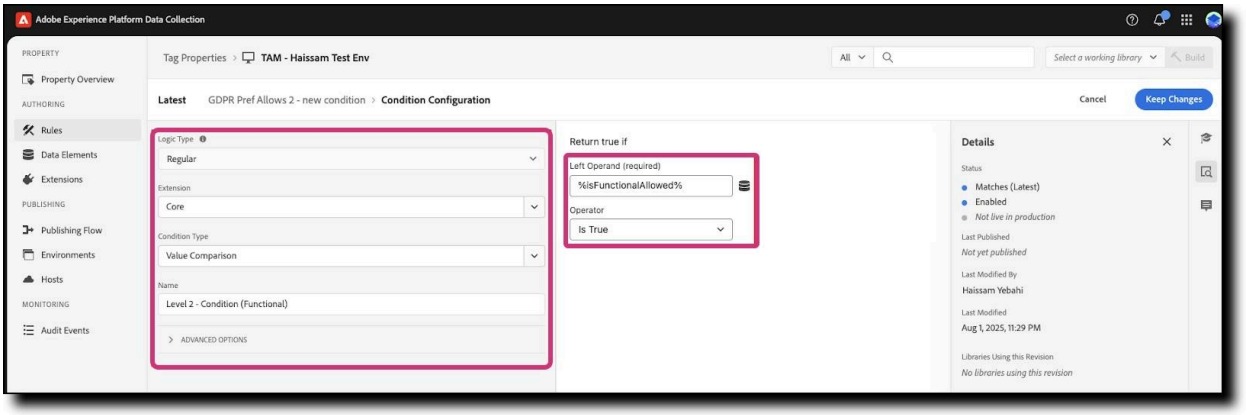
3. Click **Keep Changes**.

Condition

1. From the *Rules* tab, open an existing Rule that is considered as a Functional Cookie. For example, **GA4 - PageView**.
2. Click the **Add** icon under *Conditions*.



3. Enter the following values on each of the given fields:



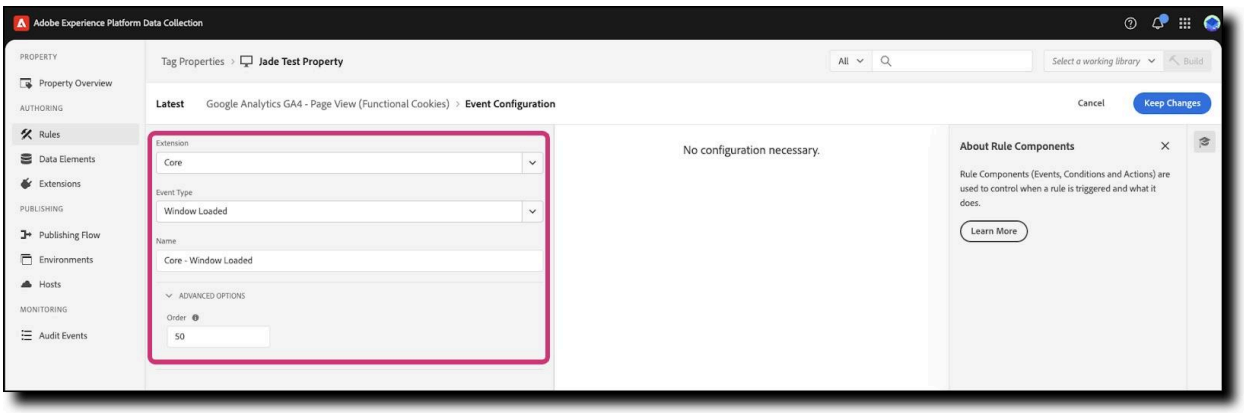
- **Logic Type**: *Regular*
- **Extension**: *Core*
- **Condition Type**: *Value Comparison*
- **Name**: *Level 2 - Condition (Functional)*
- **Left Operand**: *%isFunctionalAllowed%*
- **Operator**: *Is True*

4. Click **Keep Changes**.

Setting up an Advertising Rule

Standard Event to Fire an Advertising Tag

1. Click the **Add** icon again to add an *Event*.
2. Enter the following values on each of the given fields:



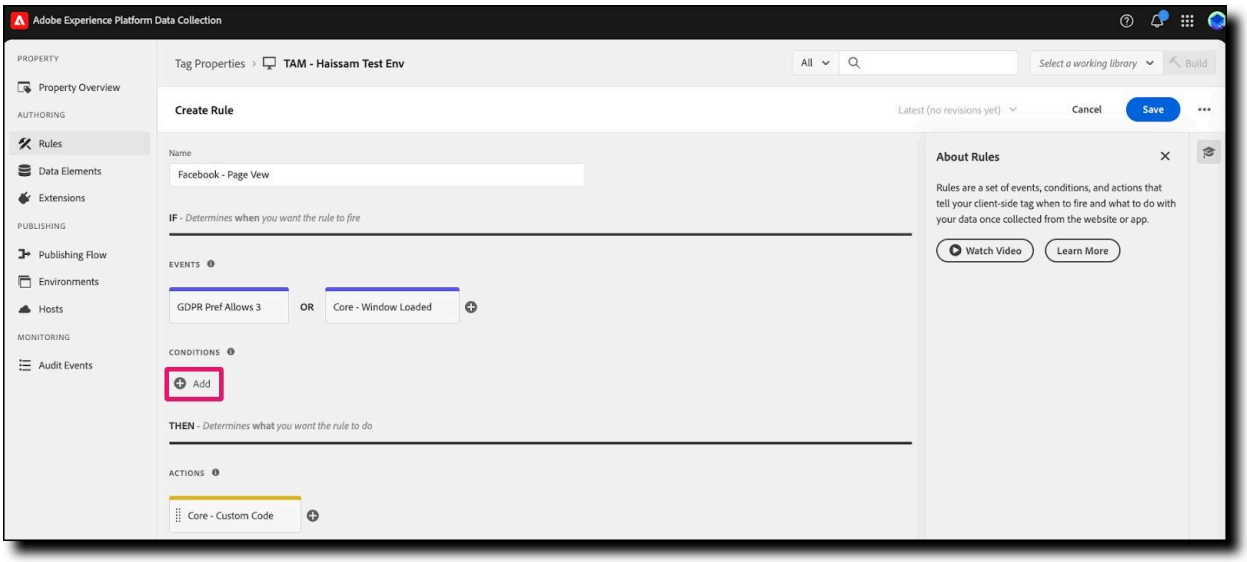
- **Extension**: *Core*
- **Event Type**: *Window Loaded*
- **Name**: Provide a new event **Name** or leave it as default.
- **Order**: Set a new **Order** value or leave it as default.

  NOTE: The default order for all rule components is **50**. If two rules have the same event type, the one with the lowest number runs first.

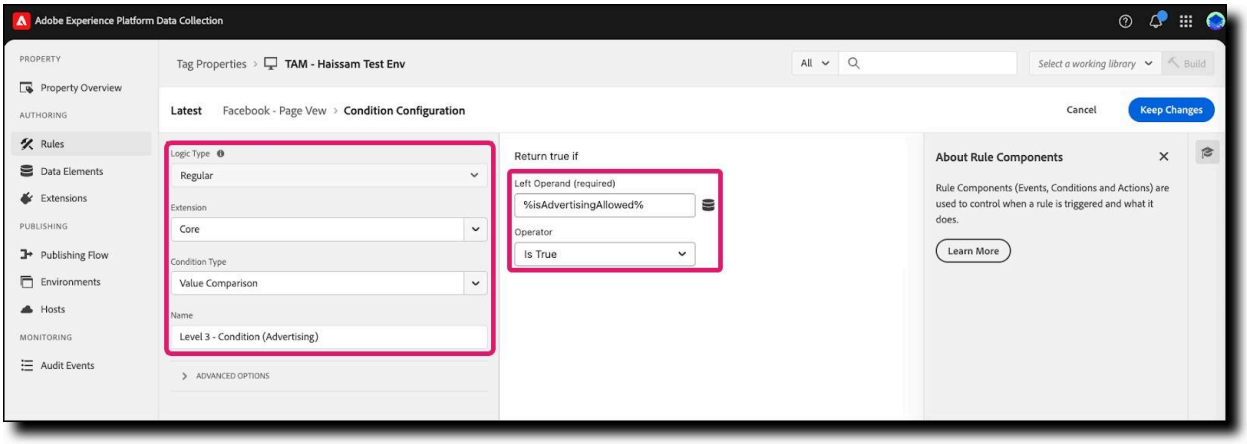
3. Click **Keep Changes**.

Condition

1. From the *Rules* tab, open an existing Rule that is considered as a Functional Cookie. For example, **Facebook - Page View**.
2. Click the **Add** icon under *Conditions*.



3. Enter the following values on each of the given fields:



- **Logic Type**: *Regular*
- **Extension**: *Core*
- **Condition Type**: *Value Comparison*
- **Name**: *Level 3 - Condition (Advertising)*
- **Left Operand**: *%isAdvertisingAllowed%*
- **Operator**: *Is True*

4. Click **Keep Changes**.

Implementing ECID Opt-in Services

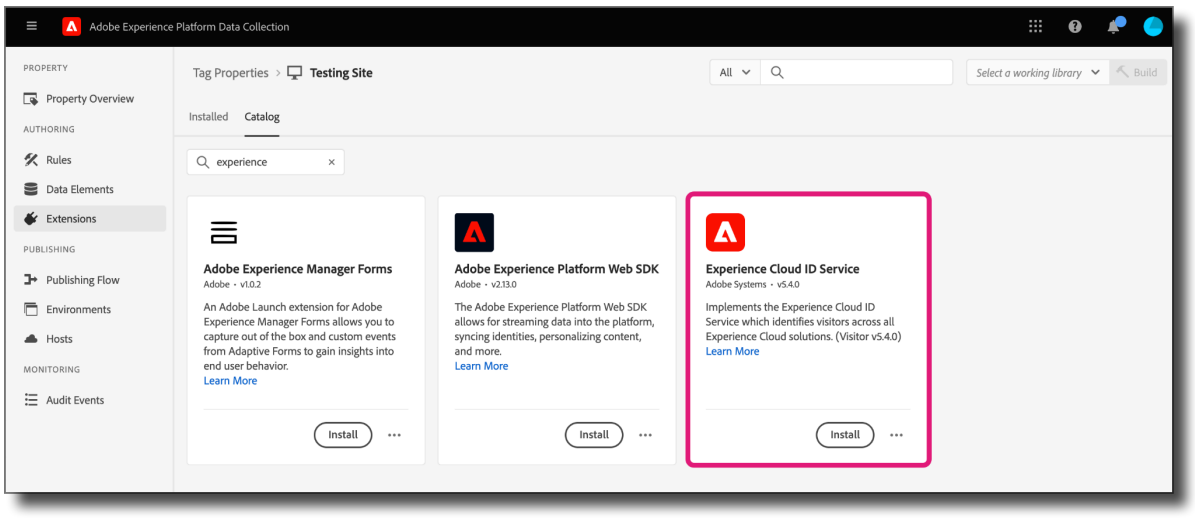
The Experience Cloud ID (ECID) service, integrated into Adobe Experience Platform's data collection technologies, is designed to manage which Adobe Experience Cloud solutions can create cookies on a web page. This service plays a crucial role in establishing a consent opt-in workflow, allowing you to control the deployment of Adobe product tags both before and after a user provides their consent or expresses their preferences.

By utilizing the ECID service, you can configure your site to permit certain Adobe cookies to load with pre-consent, enabling essential functionalities while still respecting user choice. Subsequently, you can set up rules to ensure that other Adobe product tags are deployed only after the user has granted their consent.

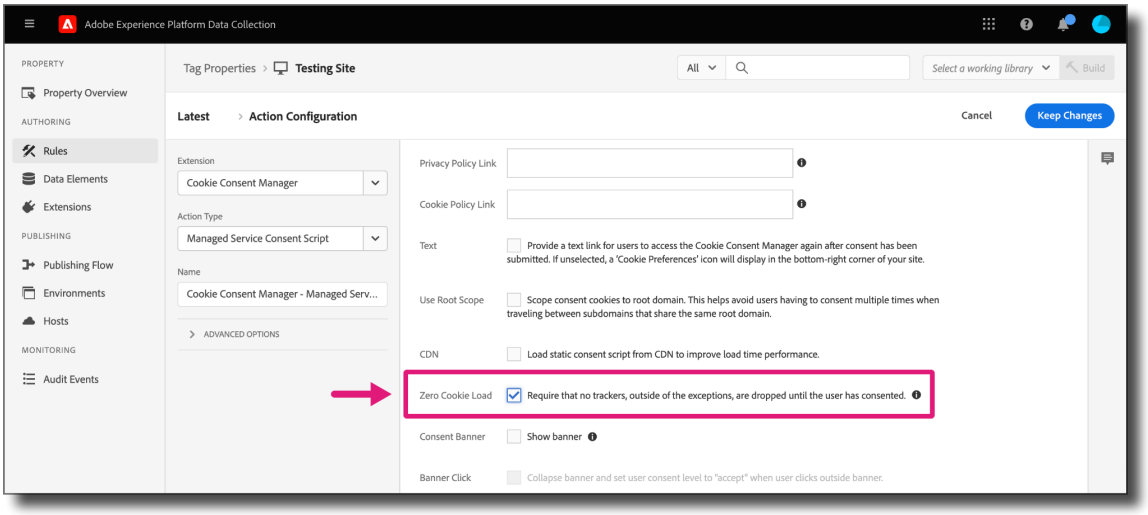
NOTE: While the ECID service facilitates the management of Adobe cookies based on consent, it does not function as a direct mechanism for gathering user consent preferences or as a repository for these preferences. Its primary purpose is to enable the dynamic control of Adobe product tags in alignment with your established consent framework.

Pre-requisites

1. Adobe Experience Platform Setup
    - **Experience Cloud ID Service** extension must be installed.



2. Truste Notice Tag
    - Tag Manager support enabled
        - If using the Cookie Consent Manager (Trustarc) extension, **Zero Cookie Load** should be enabled.



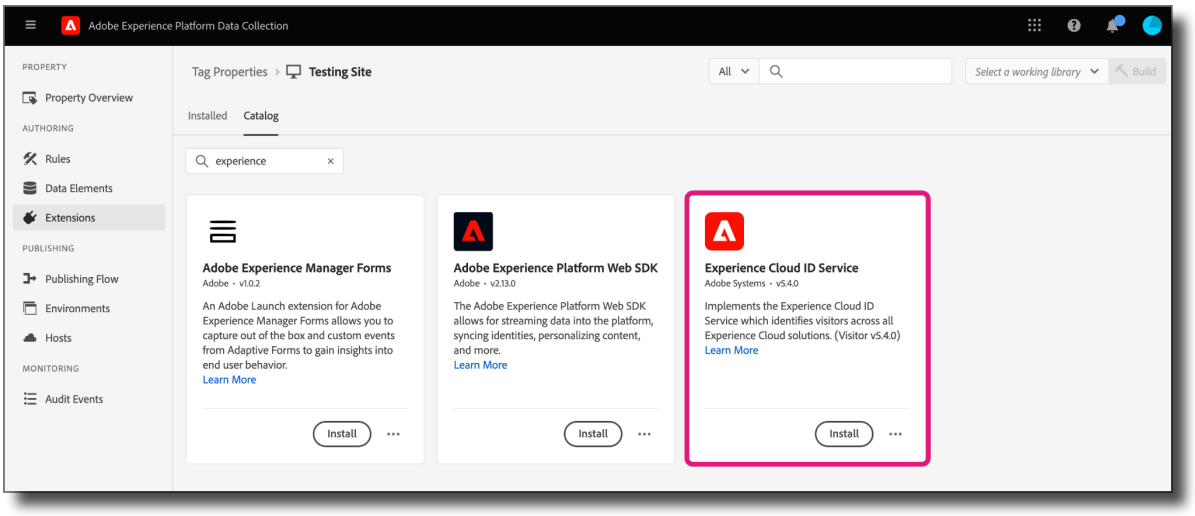
        - If manually creating/adding notice tag, pass **gtm=1** as parameter.

Disclaimer: Plugins and browser extensions can block your tag managers and then, block the cookie consent manager. To prevent this behavior, consider adding the Cookie Consent Manager script directly to the HEAD tag of the site code rather than relying solely on the tag manager.

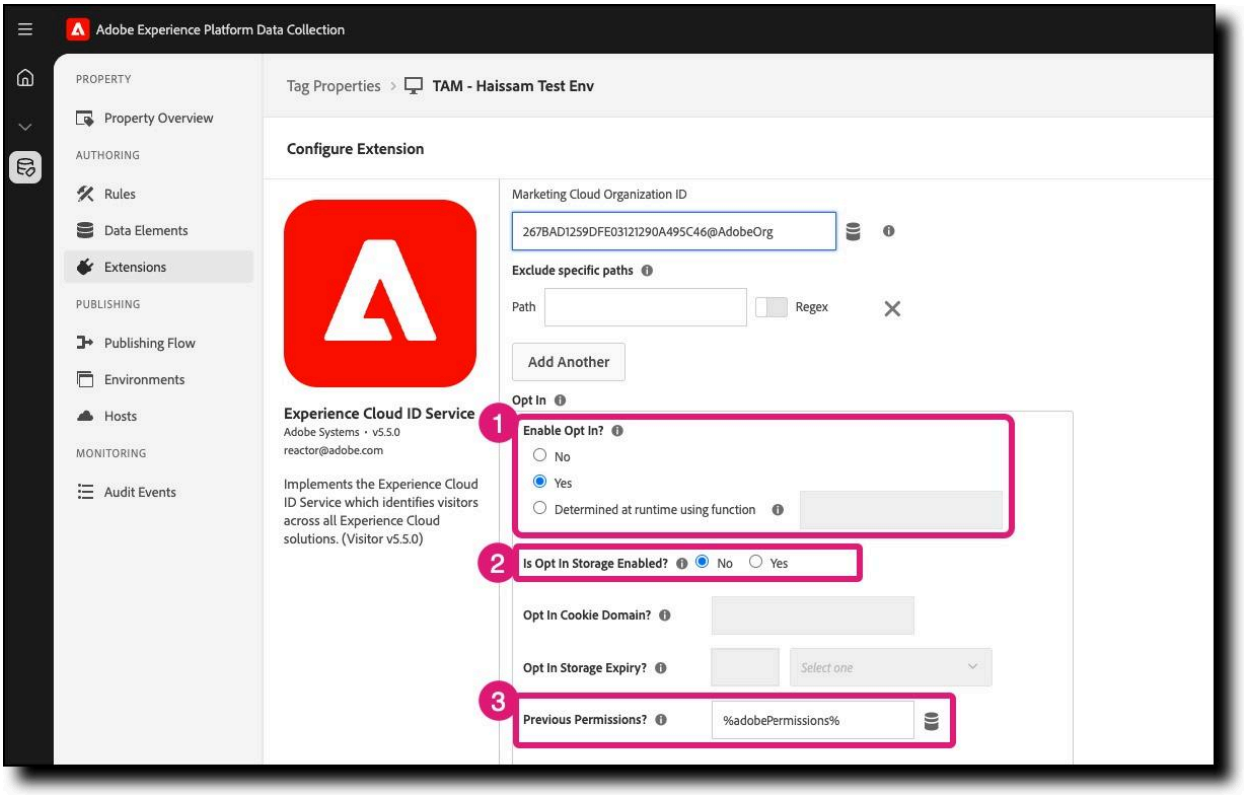
Setup

Configure ECID Service Extension

1. Under *Catalog*, search and install **Experience Cloud ID Service**.



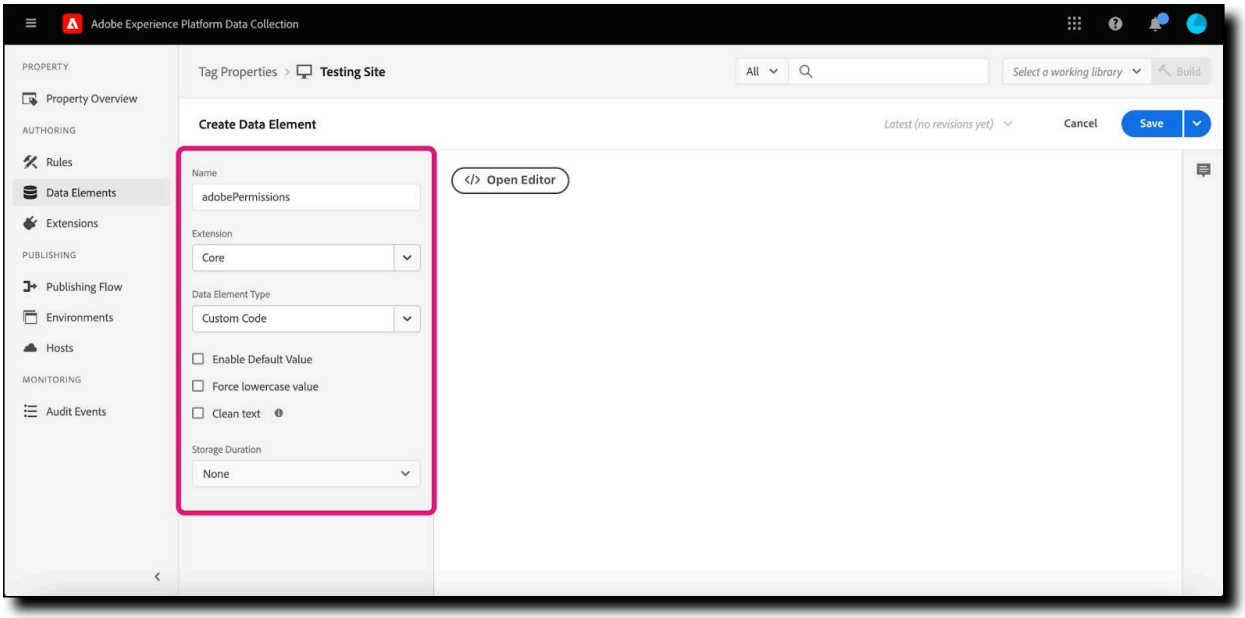
2. Under the *Opt In* column, select **Yes** (1) to enable Opt In. Under the *Is Opt In Storage Enabled?* option, select **No** (2), then enter *%adobePermissions%* in the **Previous Permissions** field (3), and click **Save**.



Create Previous Permissions Function

This function is used to inform ECID about the current permissions for Adobe Products. Whenever the user consents, the settings will be persisted.

1. From the *Data Elements* tab, click **Add Data Element**.
2. Enter the following values in each of the given fields:



- **Name**: Enter *adobePermissions* for this data element name.
- **Extension**: *Core*
- **Data Element Type**: *Custom Code*

3. Click the **</> Open Editor** button and enter the following script:

JavaScript

```
var functionalallowed   = false;
var advertisingallowed = false;

var consentreturn = {
    "aam":      false, //Adobe Audience Manager (AAM) for audience targeting
activities
    "aa":       false, //Adobe Analytics (AA)
    "target":   false, //Adobe Target
    "ecid":     false, //Experience Cloud Visitor ID (ECID)
    "adcloud":  false, //Adobe Advertising Cloud
    "campaign": false, //Adobe Campaign Manager
    "livefyre": false  //Livefyre content curation and audience engagement platform
};

if(document.cookie === "") {
  console.log('nocookieset!');
  return consentreturn;
}


functionalallowed = _satellite.getVar('isFunctionalAllowed');
```

```
advertisingallowed = _satellite.getVar('isAdvertisingAllowed');


consentreturn = {
    "aam":       advertisingallowed, //Adobe Audience Manager (AAM)
    "aa":        functionalallowed,  //Adobe Analytics (AA)
    "target":    advertisingallowed, //Adobe Target
    "ecid":      advertisingallowed, //Experience Cloud Visitor ID (ECID)
    "adcloud":   advertisingallowed, //Adobe Advertising Cloud
    "campaign":  advertisingallowed, //Adobe Campaign Manager
    "livefyre":  advertisingallowed //Livefyre content curation and audience
engagement platform
};

return consentreturn;
```

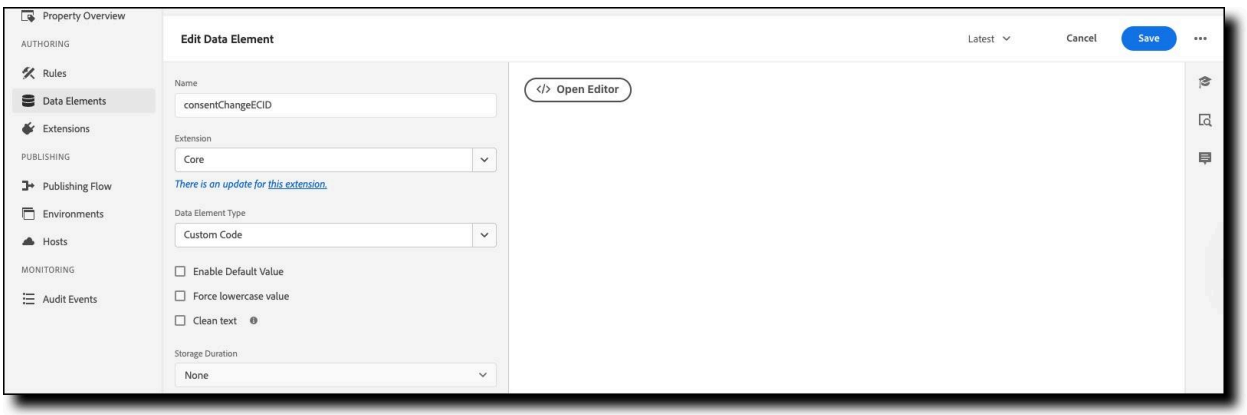
4. Click **Save** to create the data element.

Create Consent Change Rule

This rule listens for consent change events and updates ECID categories to fire accordingly.

First you need to create a data element. This data element is used to inform ECID about consent changes without a page reload.

1. From the *Data Elements* tab, click **Add Data Element**.
2. Enter the following values in each of the given fields:



- **Name**: Enter a **Name** for this data element: *consentChangeECID*
- **Extension**: *Core*
- **Data Element Type**: *Custom Code*

4. Click the **</> Open Editor** button and enter the following script:

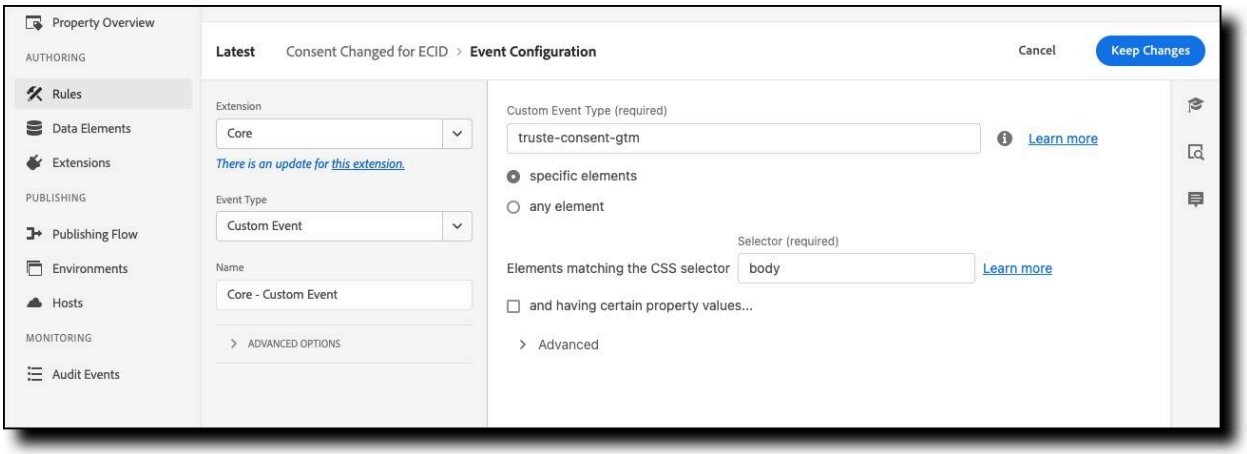
JavaScript

```
console.log('Executing consentChangeECID');
var adobepermissions = _satellite.getVar('adobePermissions');

// Mapping keys to Adobe Opt-In categories
const categoryMap = adobe.OptInCategories;

const approveList = [];
const denyList = [];

// Loop through the object
for (const [key, value] of Object.entries(adobepermissions)) {
  if (key in adobepermissions) {
    if (value) {
      approveList.push(key);
    } else {
      denyList.push(key);
    }
  }
}

// Build the functions
if (approveList.length > 0) {
  console.log(`adobe.optIn.approve(${JSON.stringify(approveList)}, true);`);
  adobe.optIn.approve(approveList, true);
}
if (denyList.length > 0) {
  console.log(`adobe.optIn.deny(${JSON.stringify(denyList)}, true);`);
  adobe.optIn.deny(denyList, true);
}
console.log('adobe.optIn.complete();');
adobe.optIn.complete();

return true;
```

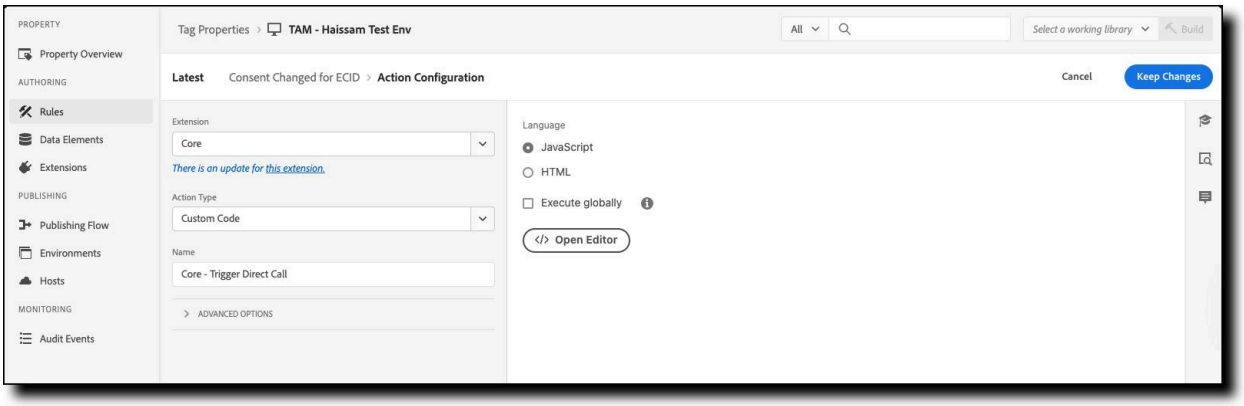
5. Click **Save** to create the data element.

After the steps above are completed, proceed creating the rule.

1. From the list of properties, select a **Property** where you want to apply the rule.
2. From the *Rules* tab, click **Add Rule**.
3. Enter a **Name** for the rule. For example, *Consent Changed for ECID*.
4. Click the **Add** icon under *Events*.
5. Enter the following values on each of the given fields:



   - **Extension**: *Core*
   - **Event Type**: *Custom Event*
   - **Name***:* Enter a **Name** for this event or leave it as default.
   - **Custom Event Type (required)**: Enter **truste-consent-gtm**. Ensure that **specific elements** are selected.
   - **Elements matching the CSS selector**: Enter **body**.
6. Click **Keep Changes**. This sets the event you want the rule to look for before firing.
7. Click the **Add** icon under *Actions*.


PROPERTY
Property Overview
AUTHORING
Rules
Data Elements
Extensions
PUBLISHING
Publishing Flow
Environments
Hosts
MONITORING
Audit Events
Tag Properties
TAM - Haissam Test Env
All
Select a working library
Build
Latest
Consent Changed for ECID
Action Configuration
Cancel
Keep Changes
Extension
Core
There is an update for this extension.
Action Type
Custom Code
Name
Core - Trigger Direct Call
ADVANCED OPTIONS
Language
JavaScript
HTML
Execute globally
Open Editor

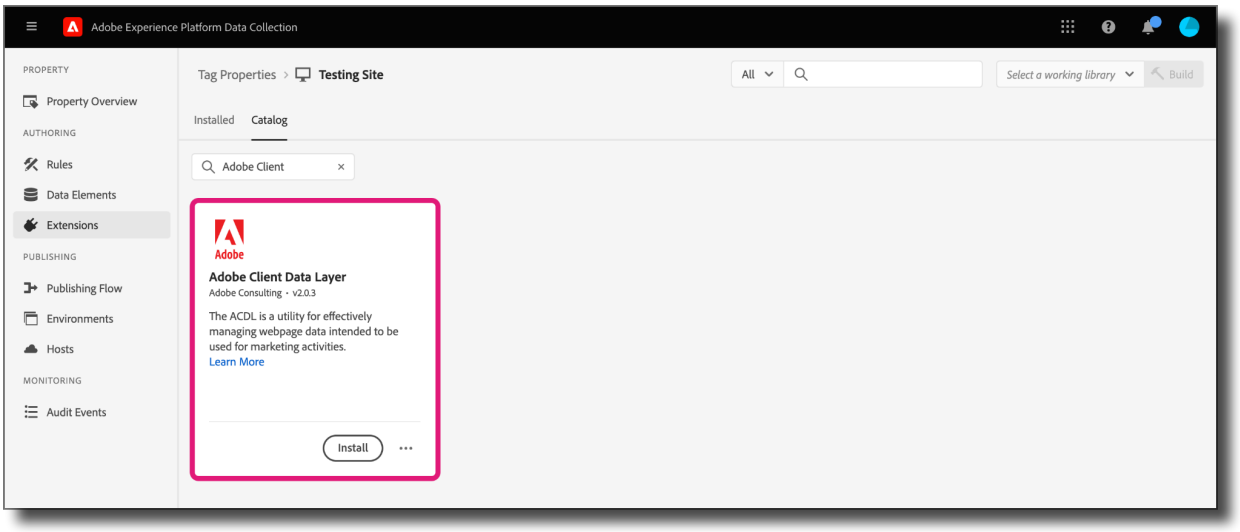
Using Custom Events for Firing Tags Dynamically

Working with Adobe Event Listener

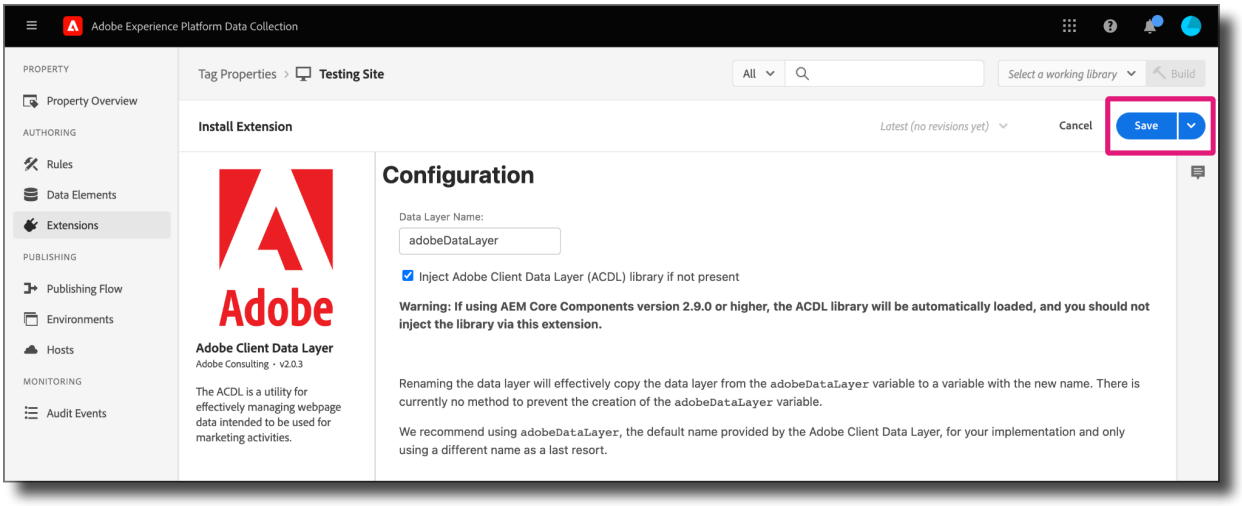
This rule listens for the `adobeDataLayer` event, which is fired on consent set and checks if consent cookie is Functional or Advertising level. This is only needed if you are dynamically firing tags after a website visitor changes their preferences.

IMPORTANT: If you have a page reload in place, firing triggers dynamically after a user changes consent is not recommended, as tags will load accordingly once the page loads.

1. Ensure that Adobe Client Data Layer extension is installed. Under **Extensions > Catalog**, search and install **Adobe Client Data Layer**.



2. Click **Save** to install the *Adobe Client Data Layer* extension.

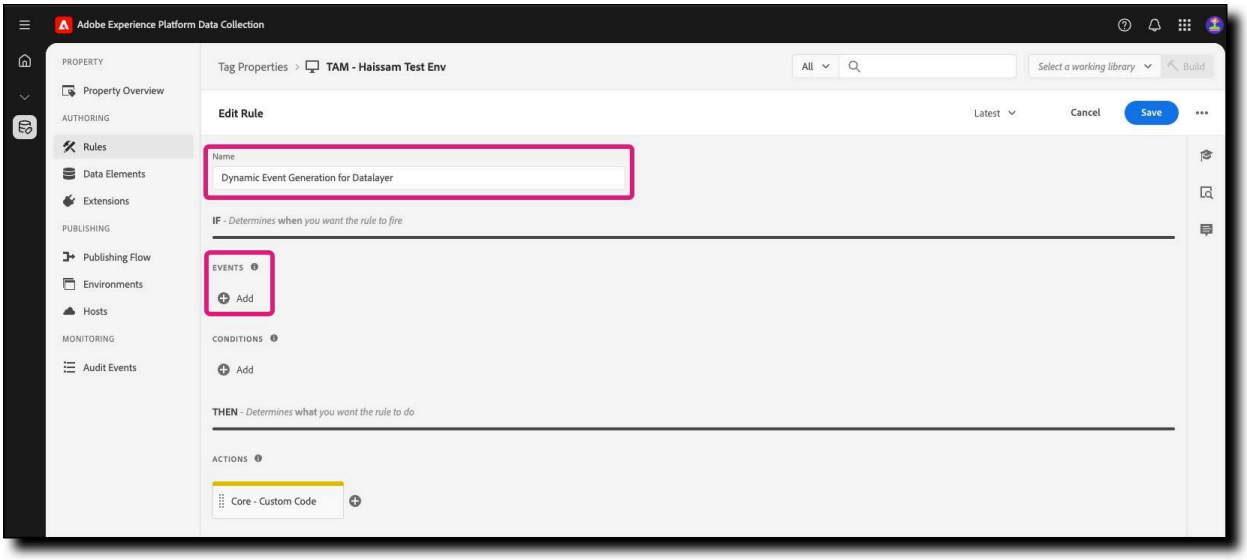


Adding a Dynamic Event Generation for Datalayer

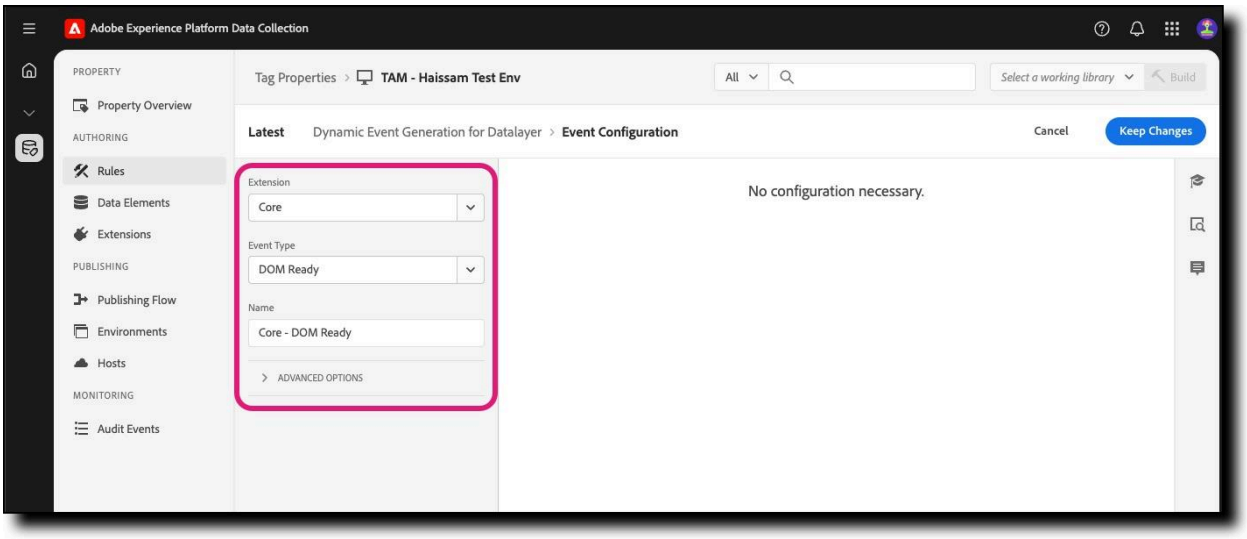
The Dynamic Event Generation for Datalayer pushes custom events to reload the page when an event is pushed to the `adobeDataLayer`. This allows tags to load dynamically according to the user's preferences. By reloading the page, any additional cookies the user consented to are then executed.

To add this script, follow these steps:

1. From the list of properties, select a **Property** where you want the rule to be applied.
2. From the *Rules* tab, click **Add Rule**.
3. Enter a **Name** (1) for the load script, such as *Dynamic Event Generation for Datalayer*.
4. Click the **Add** (2) icon under *Events*.



5. Enter the following values in each of the given fields:

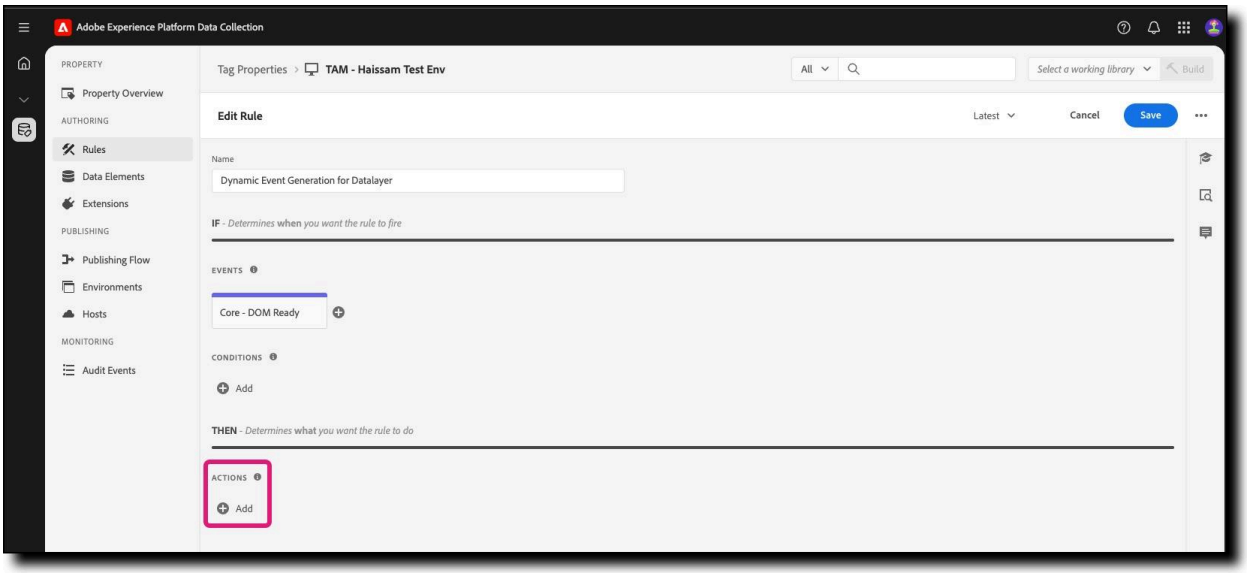


- **Extension**: *Core*
- **Event Type**: *DOM Ready*
- **Name**: Enter a new **Name** for this event or leave it as default.
- **Order**: Set the new **Order** value or leave it as default.

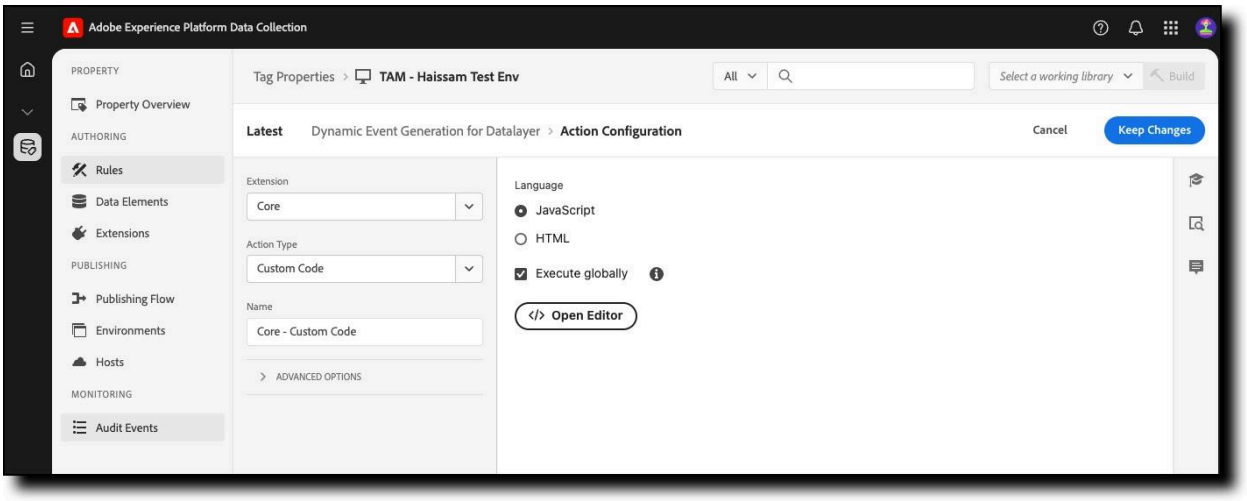
  NOTE: The default order for all rule components is **50**. If two rules have the same event type, the one with the lowest number runs first.

6. Click **Keep Changes**.

7. Click the **Add** icon under *Actions*.



8. Enter the following values on each of the given fields:



- **Extension**: *Core*
- **Action Type**: *Custom Code*
- **Name**: Enter a new **Name** for this action or leave it as default.
- **Language**: Select **Javascript** and **Execute globally** checkbox.

9. Click **</> Open Editor** and enter the following code:

JavaScript

```
var __dispatched__ = {}; //Map of previously dispatched preference levels
/*
First step is to register with the CM API to receive callbacks when a preference
update
occurs. You must wait for the CM API (PrivacyManagerAPI object) to exist on the page
before
registering.
*/
var __i__ = self.postMessage && setInterval(function() {
    if (self.PrivacyManagerAPI && __i__) {
        var apiObject = {
            PrivacyManagerAPI: {
                action: "getConsentDecision",
                timestamp: new Date().getTime(),
                self: self.location.host
            }
        };
        self.top.postMessage(JSON.stringify(apiObject), "*");
        __i__ = clearInterval(__i__);
    }
}, 50);
/*
Callbacks will occur in the form of a PostMessage event. This code listens for the
```

```
appropriately formatted PostMessage event, gets the new consent decision, and then
pushes
the events into the GTM framework. Once the event is submitted, that consent decision
is
marked in the 'dispatched' map so it does not occur more than once.
*/
self.addEventListener("message", function(e, d) {
    try {
        if (e.data && (d = JSON.parse(e.data)) &&
            (d = d.PrivacyManagerAPI) && d.capabilities &&
            d.action == "getConsentDecision") {
            var newDecision = self.PrivacyManagerAPI.callApi("getGDPRConsentDecision",
self.location.host).consentDecision;
            newDecision && newDecision.forEach(function(label) {
                if (!__dispatched__[label]) {
                  self.adobeDataLayer && self.adobeDataLayer.push({ "event": "GDPR
Pref Allows " + label});
                            __dispatched__[label] = 1;
                    }
                });
            }
    }
    catch (xx) {
        /** not a cm api message **/
    }

});
```

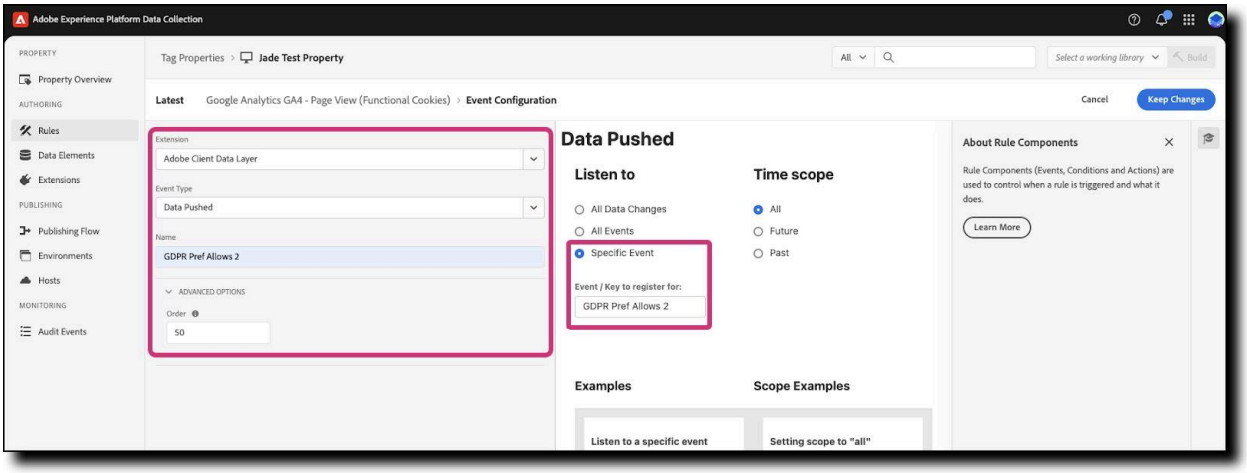
10. Click **Keep Changes**. The actions are performed once an event is triggered and all conditions and exceptions are evaluated.
11. Click **Save** to save the script.

Event Listener for Dynamic Loading Based on Consent

IMPORTANT: Use this if you want to dynamically load tags after a user makes a change to consent. Note that triggering a page reload is not recommended if you are doing so.

Custom Event for Functional Tags

1. From the *Rules* tab, open an existing Rule that is considered as a Functional Cookie. For example, **GA4 - PageView**.
2. Click the **Add** icon under *Events*.
3. Enter the following values on each of the given fields:



- **Extension**: *Adobe Client Data Layer*
- **Event Type**: *Data Pushed*
- **Name**: *GDPR Pref Allows 2*
- **Order**: Set a new **Order** value or leave it as default.

  NOTE: The default order for all rule components is **50**. If two rules have the same event type, the one with the lowest number runs first.

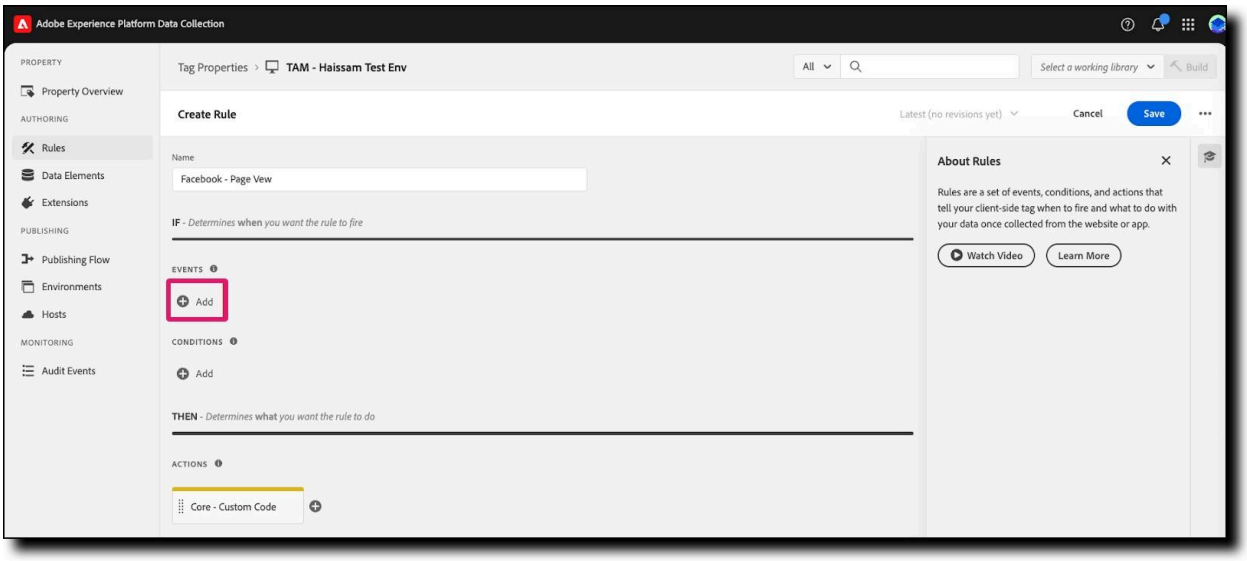
- **Listen to**: Select **Specific Event** in the column and set a value in the **Event / Key to register for** field; for example, *GDPR Pref Allows 2*.

4. Click **Keep Changes**.

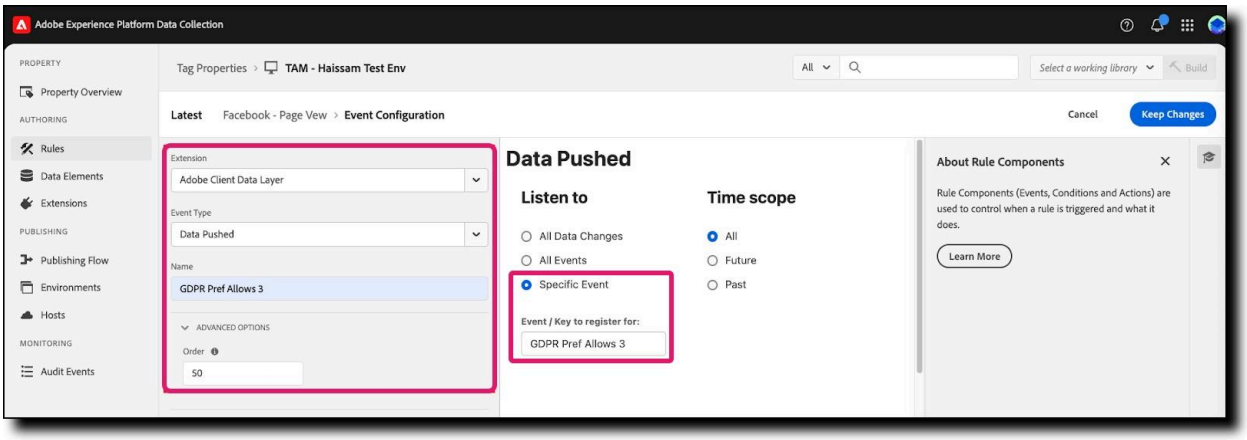
Custom Event for Advertising Tags

1. From the *Rules* tab, open an existing Rule that is considered as a Functional Cookie. For example, **Facebook - Page View**.

2. Click the **Add** icon under *Events*.



3. Enter the following values on each of the given fields:



- **Extension**: *Adobe Client Data Layer*
- **Event Type**: *Data Pushed*
- **Name**: *GDPR Pref Allows 3*
- **Order**: Set a new **Order** value or leave it as default.

  NOTE: The default order for all rule components is **50**. If two rules have the same event type, the one with the lowest number runs first.

- **Listen to**: Select **Specific Event** in the column and set a value in the **Event / Key to register for** field; for example, GDPR Pref Allows 3.

4. Click **Keep Changes**.

Tealium Tag Manager

Tealium is a provider of a tag management system, including analytics, advertising, and affiliate enterprise websites. TrustArc Cookie Consent can be used inside the Tealium portal as a way to manage the loading of your tags. The steps in this section will guide you through the process.

The sections outlined in this document provide the instructions to utilize Tealium Data Sources and Load Rules in a Zero-Tracker Consent Manager deployment for both Standard and Zero-Tracker Loads.

Before using this integration guide, please make sure the following items are completed:

- You have identified all tracking tags on your site and their corresponding Bucketing Category.
- All tracking tags corresponding to a non-Required tracker have been moved into Tealium.
- You have added your Tealium script to all of your site's pages.

Consent Behaviors

The Consent Manager can utilize two different consent behaviors, Implied Consent (uninterrupted user navigation on the site) and Expressed Consent (interrupted user navigation on the site until the user provides consent), and each country's consent behavior can be set independently.

For Implied Consent configured countries, the Consent Manager is available for the user to launch, from either the Cookie Preferences button or the Implied Consent Banner, but they are not forced to provide consent before interacting with the site. When a user launches the Consent Manager in an Implied Consent country they can close the Consent Manager at any time via an **X** or **Cancel** button.

For Expressed Consent configured countries, the Consent Manager is launched automatically when a user with no valid consent preference is detected. The user must provide their consent before interacting with the site and the Consent Manager cannot be closed until consent is provided. Once consent has been provided the user can launch the Consent Manager from the *Cookie Preferences* button to review or change their selection but they are not required to provide a new submission.

Deployment Strategies

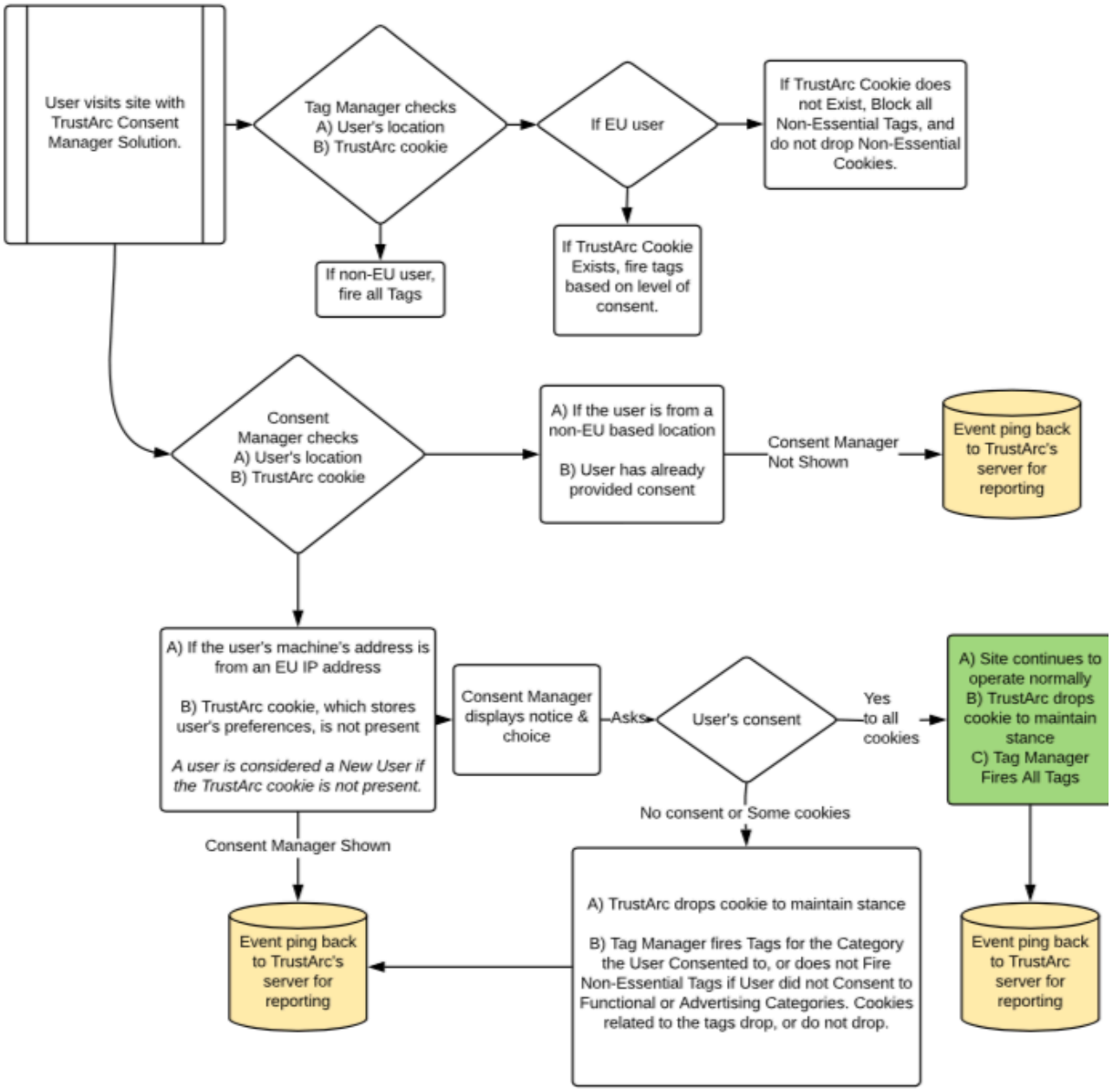
Independent of the consent behavior, the Consent Manager can be deployed on a site in one of two methods – **Standard** or **Zero-Tracker**. In a Standard deployment, the Consent Manager is added to the site and when a user opts out, it will initiate any available Third Party Vendor opt-out mechanisms. Please note that in a **Standard** deployment no trackers are blocked and vendor opt-out mechanisms are only applicable to Cookie and LSO tracking technologies. In a **Zero-Tracker** deployment, in addition to adding Consent Manager to the site, it is integrated with Tealium to block the tracker code from being injected into the page based on the user's consent.

Please note that a **Zero-Tracker** deployment allows for the blocking of all tracking technologies, including technologies such as JavaScript and Web Beacons.

When the Consent Manager is deployed in a **Zero-Tracker** deployment, there are two different tracker loads which can be used – **Standard Load** or **Zero-Tracker** Load. With **Standard Load**, all trackers are fired until the user *opts out* of tracking. With **Zero-Tracker** Load, no non-Required trackers are fired until the user *opts in* to tracking. The tracker load used can be either layout dependent (EU / US), consent behavior dependent (Implied / Expressed), or a combination of both.

User Flow

Below is a decision flow chart based on a new user first encountering a CM which has been integrated with Tealium, where Tealium is blocking or controlling the firing of tags and the dropping of trackers based on the end user's level of consent in the CM. In the flow chart, non-EU users experience a **Standard Load** while EU users experience a **Zero-Tracker** Load.


User visits site with TrustArc Consent Manager Solution.
Tag Manager checks
A) User's location
B) TrustArc cookie
If EU user
If TrustArc Cookie does not Exist, Block all Non-Essential Tags, and do not drop Non-Essential Cookies.
If non-EU user, fire all Tags
If TrustArc Cookie Exists, fire tags based on level of consent.
Consent Manager checks
A) User's location
B) TrustArc cookie
A) If the user is from a non-EU based location
B) User has already provided consent
Consent Manager Not Shown
Event ping back to TrustArc's server for reporting
A) If the user's machine's address is from an EU IP address
B) TrustArc cookie, which stores user's preferences, is not present
A user is considered a New User if the TrustArc cookie is not present.
Consent Manager displays notice & choice
Asks
User's consent
Yes to all cookies
A) Site continues to operate normally
B) TrustArc drops cookie to maintain stance
C) Tag Manager Fires All Tags
No consent or Some cookies
Consent Manager Shown
A) TrustArc drops cookie to maintain stance
B) Tag Manager fires Tags for the Category the User Consented to, or does not Fire Non-Essential Tags if User did not Consent to Functional or Advertising Categories. Cookies related to the tags drop, or do not drop.
Event ping back to TrustArc's server for reporting
Event ping back to TrustArc server for reporting

Consent Manager Assets

This section outlines the TrustArc specific code which is required to deploy a Consent Manager in a Tealium integrated Zero-Tracker deployment. This code can be deployed either directly to the page code or through Tealium.

Important: Please review the important note at the end of the *Using Tealium to Deploy CM Assets* section regarding deploying Consent Manager code via Tealium.

Consent Manager Code

You will need to add your instance-specific Consent Manager code, provided by your Technical Account Manager, to each page to be covered by the Consent Manager. An example of this code would be:

```
None
<div id="consent-banner"></div>
<div id="teconsent">
<script type="text/javascript" async="async"
src="//consent.trustarc.com/v2/notice/{cmId}"></script>
</div>
```

You will need to place the `consent_banner` div (highlighted in **red**) on the page location where you would like the consent banner to display. Note that this should be visible to the user when they first view the page so it should be either static at the top of the page or floating at either the top or bottom of the screen. The `teconsent` div (highlighted in **purple**) should be placed where you would like the Cookie Preferences button/link to display.

Important: It takes approximately 500ms for the Consent Manager to identify and provide the necessary information for Tealium to correctly block trackers for users who have not previously set their consent. If you are using a Zero-Tracker Load, you will need to place the Consent Manager script (highlighted in **blue**) to run as early as possible on the page, preferably as the first script to run in the header, to ensure Tealium has the required information to properly block the necessary tags.

Please refer to TrustArc's Consent Manager Deployment Guide for more information.

Disclaimer: Plugins and browser extensions can block your tag managers and then, block Consent Manager. Please check if your tag manager works with the popular Adblockers. If your tag manager is blocked, you may consider firing the TrustArc scripts outside of the tag manager in the HEAD tag.

Tealium Event Listener

Important: This feature will work only with domains that have GDPR layout. When these events are fired, the cookie preferences will not be reflecting the changes in the user's preferences. Please consider using the event listed here if you are looking to retrieve the updated preferences.

The event listener is a standard Javascript listener that will listen for custom events. The code outlined below will listen for the following event types:

Event Type	Description	Notes
`truste-click-button-submit`	An event is triggered when the **Submit Preferences** button is clicked.	
`truste-click-toggle-category-<level>-on`	An event is triggered when the **specified category** is set to ON.	Default Categories: • `truste-click-toggle-category-1-on` for **functional** • `truste-click-toggle-category-2-on` for **advertising**

CCM Professional provides three Tracker Categorizations out of the box:

- Required Cookies – **Level 0**
- Functional Cookies – **Level 1**
- Advertising Cookies – **Level 2**

NOTE: If you have customized your tracker categorizations or added additional categorizations, the levels will match the listing order in your CCM instance(s).

In order to generate the required Tealium events, you must add the following code to every page on which the CM is deployed. You can modify the code by adding lines for additional categories depending on how many categories you have on your domain.

```
None
//Target "YES" toggle for "Functional" category
var analyticsYes = function() {
    utag.link({
        "event_name":"consent_toggle",
```

```
        "event_category":"functional",
        "event_label":"yes"
    });
}

//Target "Submit Preferences" button in GDPR modal
var tealClick2 = function() {
    utag.link({
        "event_name":"consent_submit",
        "event_category":"submit",
        "event_label":"submit"
    });
};

document.body.addEventListener('truste-click-button-submit', tealClick2);
document.body.addEventListener('truste-click-toggle-category-1-on', tealClick2);
```

You can create additional listeners for the different category levels from CCM, for example:

```
None
var AdvertisingYes = function() {
    utag.link({
        "event_name":"consent_toggle",
        "event_category":"advertising",
        "event_label":"yes"
    });

document.body.addEventListener('truste-click-toggle-category-2-on', tealClick2);
```

Please reach out to your Tealium Account Manager for more information about the Tealium Event Listener.

Using Tealium to Deploy CM Assets

Step 1: Add TrustArc Cookie Consent Tag into Tealium Portal

Tealium has a TrustArc tag template, which you can use to create your TrustArc Cookie Consent tag. To do this, in the my.tealiumiq.com portal:

1. Select the **Tags** tab.
2. Click on the green **+ Add Tag** button.
3. In the search bar, enter **truste**. This will display the TrustArc template at the top of the search results.
4. Click **+ Add** on the TrustArc EU Solution tag template.

   This will open up a new frame where you can enter information into fields. Please use the fields provided to you by your TrustArc Account Manager. Specifically, the Domain and DIV ID fields should match your account information.
5. Click on **Next** to apply *Load Rules* to the tag. Apply the appropriate *Load Rule* so this tag is added to the pages you wish to allow your site users to be able to set and manage their TrustArc Cookie Consent preferences.

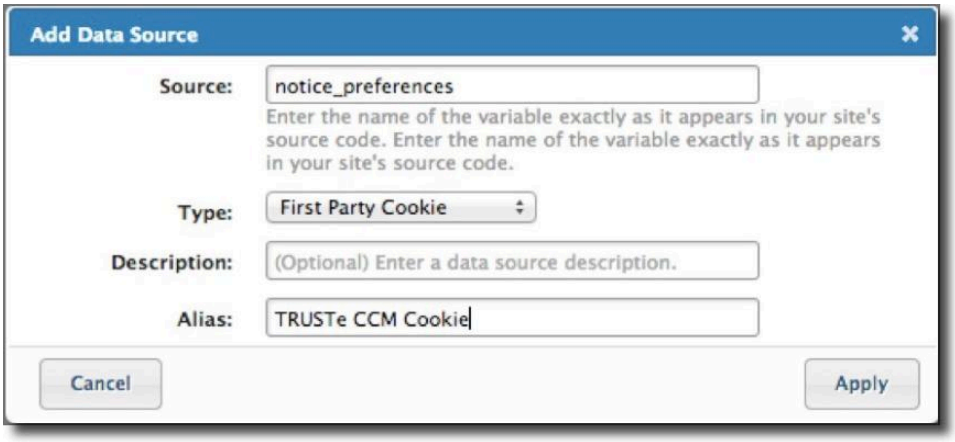
   Important: When using a Zero-Tracker Load, it is NOT recommended to use Tealium to add the CM script to the page. The recommended placement of the CM script for a Zero-Tracker Load is to run as early as possible on the page, preferably as the first script to run in the header, to ensure Tealium has the required information to properly block the necessary tags.

   NOTE:

   - The TrustArc Cookie Consent tag by default drops opt-out cookies when the user submits preferences. In order to leverage only the Tealium rules to honor user preferences, the script must have the Tag Management System feature enabled in the *Setup > Advanced Settings* section of the Self-Service UI Configuration Wizard.
   - If the TrustArc EU Solutions Tag template does not include the necessary configurations needed for your TrustArc Cookie Consent Manager, please create a custom tag instead of using the built-in Tealium Template.

Step 2: Create the Data Source

1. Select the **Data Layer** tab.
2. Click the **+ Add Data Source** button.
3. Change the Type to **First Party Cookie**.
4. Enter in the Source Field: **notice_gdpr_prefs**
5. Enter an Alias for the data source, we suggest **TrustArc CCM Cookie**.
6. Select **Apply**.



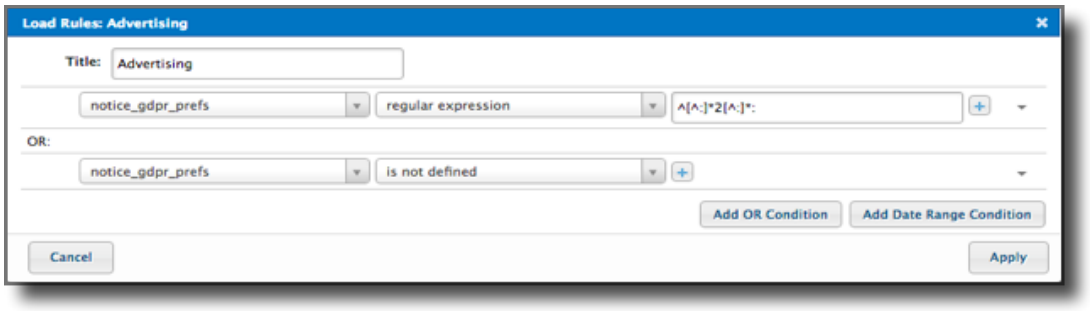
Step 3: Attach This Data Source to All Load Rules for Tags Managed by the TrustArc Cookie Consent Preference

NOTE: Implementation of both **3.A** and **3.B** below allows for all tags to load before the user submits a consent preference. **For "zero-cookie load" implementation where no tags are loaded until a consent preference is submitted do not apply 3.b below.**

A. Attach this Data Source to an existing Load Rule for Functional and Advertising tags.
    1. Select the **Load Rule** tab.
    2. Select the Load Rule you wish to apply the Condition to.
    3. Click the **Edit** button in the *Load Rule Configuration*.
    4. To the right will be a **+** button, click on this to apply an **AND Condition**.
    5. Change the first column to the **TrustArc CCM Cookie** data source you just created.
    6. Change the middle column to **regular expression**.
    7. Change the last column to the value determined below:
        I. If the Load Rule is for Advertisers, enter **^[^:]*2[^:]*:**
        II. If the Load Rule is for Functional tags, enter **^[^:]*1[^:]*:**

B. Add additional Condition below to load tags when no TrustArc Cookie Consent preference exists yet:

1. Click the **Add OR Condition** button.
2. Select the **TrustArc CCM Cookie** data source.
3. Select **is not defined** in the middle column.
4. Click **Apply** to finish.



NOTE: When a user sets their TrustArc Cookie Consent preference, that preference will be applied starting on the next page the user visits.

Tealium Data Sources

Tracker Consent

notice_preferences

- Source: **notice_preferences**
- Type: **First Party Cookie**
- Alias: **Tracker Consent**
- Cookie Purpose: **This cookie indicates the user's preference level election specifically on the legacy "Slider UI" version of Consent Manager. The slider combines levels in an inclusive manner. For example, when one has the slider set to Advertising, this includes consent for Functional.**
- Expiration: **13 months**
- Possible Values:
    - **0**: Opted out of Functional/Advertising
    - **1**: Opted out of Advertising only
    - **2**: Accepted all consent categories
    - **3+**: Refers to preference selections for custom buckets.
    - **100**: Where user visits from a country that has not been provisioned in the TrustArc backend (not available by default)

notice_gdpr_prefs

- Source: **notice_gdpr_prefs**
- Type: **First Party Cookie**
- Alias: **Tracker Consent**
- Cookie Purpose: **This cookie indicates the end user's preference level selection specifically on the Granular UI Version of Consent Manager.**
- Expiration: **13 months**
- Possible Values:
    - **0**: Refers to preference selections for Required (always present as user cannot opt out of Required)
    - **1**: Refers to preference selections for Functional
    - **2**: Refers to preference selections for Advertising
    - **3+**: Refers to preference selections for custom buckets

- **100**: Where user visits from a country that has not been provisioned in the TrustArc backend (not available by default)

Tracker Behavior

notice_behavior

- Source: **notice_behavior**
- Type: **First Party Cookie**
- Purpose: This cookie indicates the behavior configured for the country from which the end user's IP address originates and corresponds to the actual backend CM display configuration for the country. The first value indicates if the country is configured for Implied or Expressed Consent. The second value corresponds to which visual layout will be displayed (values are either EU or US). For example, "EU" might denote the end user is from the EU region and will see the EU layout, and "US" might denote the user is from the U.S. region and will see the US layout. An EU layout could also be assigned to a non-EU region, for example if we need to show a banner for non-EU countries. This is a value that is configured on the TrustArc backend.
- Enabled: **Consent Manager instance configuration**
- Expiration: **13 months**
- Possible Values:
    - **expressed|eu** (Europe)
    - **expressed|na** (North America)
    - **expressed|an** (Antartica)
    - **expressed|af** (Africa)
    - **expressed|as** (Asia)
    - **expressed|sa** (South America)
    - **expressed|oc** (Oceana)
    - **implied|eu**
    - **implied|na**
    - **implied|an**
    - **implied|af**
    - **implied|as**
    - **implied|sa**
    - **implied|oc**
    - **none**

Tealium Load Rules

This section outlines how to update the Tealium Load Rules to appropriately control the blocking of tags based on the users consent. The examples below are for applying the blocking conditions for TrustArc's default *Functional* and *Advertising* trackers consent categories with a **Zero-Tracker Load** for users from countries configured to use Expressed Consent. Your Tealium configuration should include blocking conditions corresponding to your CM instance's non-Required categories.

Expressed Consent

Functional Tags

For each tag included in the Functional Tracker category you will need to add the following conditions to any existing conditions:

- **Tracker Consent** greater than or equal to ***1***
  OR
- **Tracker Consent** is not defined ***AND*** **Tracker Behavior** contains ***implied***

Advertising Tags

For each tag included in the Advertising Tracker category you will need to add the following conditions to any existing conditions:

- **Tracker Consent** greater than or equal to ***2***
  OR
- **Tracker Consent** is not defined ***AND*** **Tracker Behavior** contains ***implied***

In both of the examples above, **Tracker Consent** is looking for a **Tracker Behavior** we know not to be true. Therefore, when **Tracker Behavior** is set to Expressed Consent in Consent Manager, any tracker that has one of the rules above applied will not load by default.

Implied Consent

Blocking conditions for TrustArc's default *Functional* and *Advertising* trackers consent categories with a **Zero-Tracker Load** for users from countries configured to use Implied Consent can be seen below.

Functional Tags

For each tag included in the Functional Tracker category you will need to add the following conditions to any existing conditions:

- **Tracker Consent** greater than or equal to ***1***
  OR
- **Tracker Consent** is not defined ***AND*** **Tracker Behavior** contains ***Expressed***

Advertising Tags

For each tag included in the Advertising Tracker category you will need to add the following conditions to any existing conditions:

- **Tracker Consent** greater than or equal to ***2***
  OR
- **Tracker Consent** is not defined ***AND*** **Tracker Behavior** contains ***Expressed***

As per the setup for Expressed Consent, we have set firing rules we know cannot be met. When a mix of Expressed and Implied Consent is used by a client, **Tracker Behavior** contains ***None*** may be used.

Verification and Troubleshooting

This section outlines how to verify a Zero-Tracker Tealium integration and some common troubleshooting steps.

Verifying a Zero-Tracker Integration

The process of verifying your Tealium Zero-Tracker Integration will vary depending on the specifics of your deployment, CM configuration, and overall tools at your disposal. However, in general you will be testing that the correct blocking occurs in three situations:

- A new user with Standard Load
- A new user with Zero-Tracker Load
- Changing the existing consent for each category

It is recommended to verify the mechanics and functionality of the Tealium Integration with a single tag of each consent category prior to applying the integration to your Tealium implementation as a whole.

A new user with Standard Load

When verifying a new user who should see a Standard Load, we are simply ensuring that all tags are fired as expected on the initial load. You should see the same tags fire after the integration as before the integration.

A new user with Zero-Tracker Load

When verifying a new user who should see a Zero-Tracker Load, we are verifying that all of the tags that have the blocking conditions applied have NOT been fired when coming from a region requiring a Zero-Tracker Load. This should be done before any user interaction with the Consent Manager.

Once consent has been submitted, and if consent has been provided, the appropriate tags should fire on the next page load.

Changing the existing consent for each category

When verifying for a user who has changed their consent we are verifying that the correct tags are blocked or fired based on the change. For example, when verifying the change in consent from Advertising to Functional we would expect to see any Advertising tags NOT to be fired on the next page

load. Conversely, a change from Functional to Advertising should cause any Advertising tags to be fired. A change both to and from each consent category is recommended.

Important: To ensure consent preferences are appropriately honored, once a tag script or code has been executed on a page (i.e., a tag has fired), you will need to configure an automatic page refresh in order for the new consent preference to be reflected. Failure to execute a refresh will result in the prior tracking behavior (e.g., tracking will continue as if an opt-out had not occurred) to persist until a manual refresh is done by a web visitor themselves.

Troubleshooting

I'm not seeing the expected Zero-Tracker Load occur

The most common cause of a Zero-Tracker Load not occurring when expected is that the Tracker Behavior Data Source does not have data when the tags firing trigger conditions are met. This frequently occurs when Tealium is used to deploy the CM script to the page and in these cases the CM script should be moved to fire as early in the header of the page as possible.

Auto-block Feature

This section provides guidance on the Cookie Consent Manager (CCM) Auto-block feature. This feature allows only required/strictly necessary trackers to fire until a user provides consent to any other cookie category or vendor.

Introduction

Under the GDPR regulations, consent is required for any cookies that are not related to the functionality of the website. Essentially, this means any cookies should be blocked on the user's browser if it does not fall under the category strictly necessary cookies and until the user has given their consent to use it.

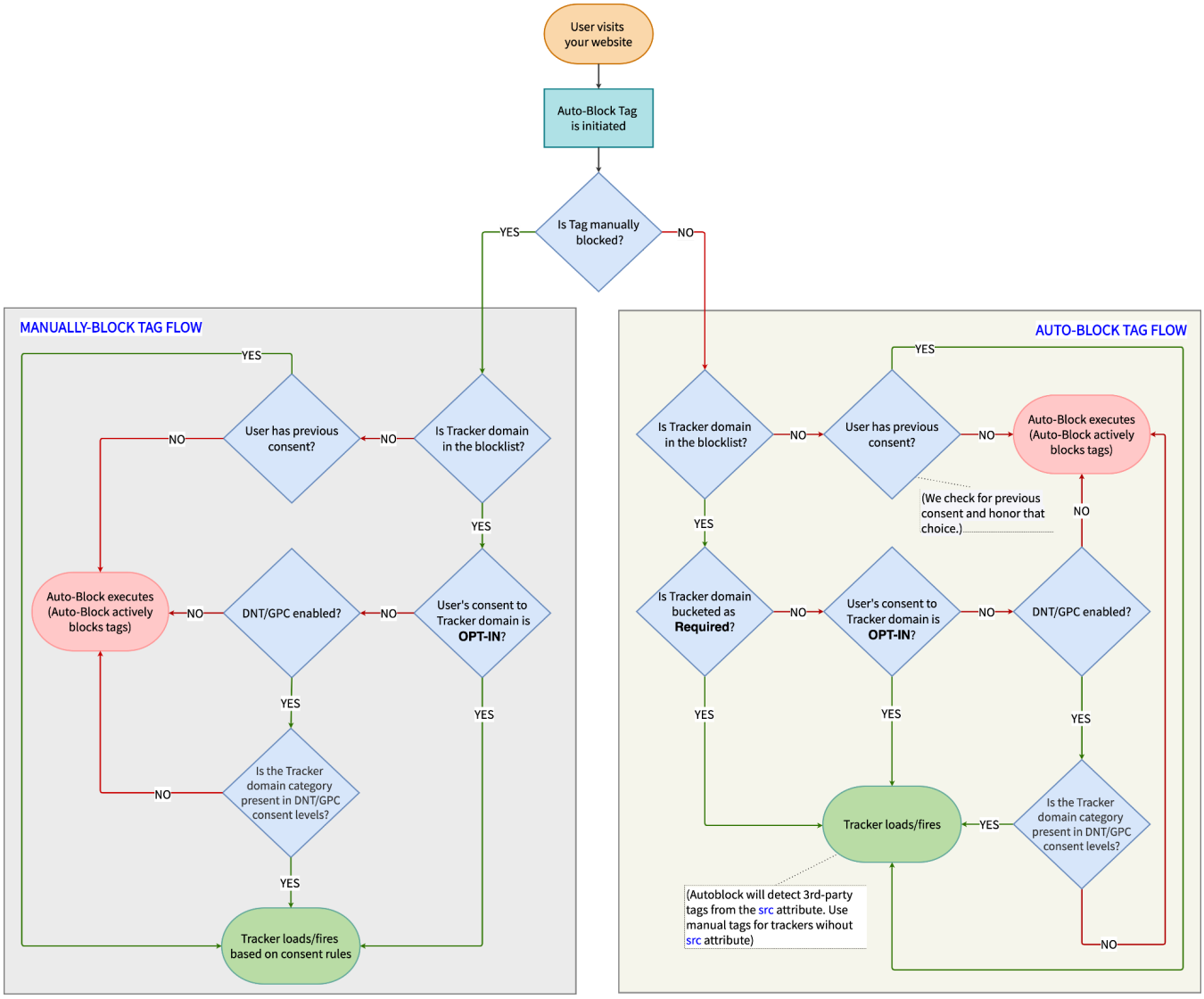
If the user has given no prior consent or chooses to reject the use of all cookies that are not strictly necessary/required, then all these cookies should be blocked and data should not be collected from the user.

The Cookie Consent Manager's Auto-block feature enables you to block the scripts of the HTML tags that can potentially drop cookies on your website until the users have given consent. This feature blocks non-required tags from firing until the user opts into any cookie category or vendor. These tags (also referred to as Trackers) are a piece of code added to a website URL that is fired from the location that is inserting the ad. This requires a user's consent before execution. The CCM Auto-block feature supports `script`, `iframe`, and `img` tags.

In cases where these tag types: `script`, `iframe`, and `img`, do not have an `src` attribute, a *Manual Block Implementation* is provided to allow our customers to identify and add these other tags to the Auto-block functionality.

User Flow

Below is a user flow chart based on how Auto-block works. Upon initial load, the Auto-block feature identifies whether a 1st Party or 3rd Party Tag is set for Manual Block or Auto-block. Here's how it works:



The next sections of this guide outline the steps on how to set up the Auto-block feature and validate its intended function.

Auto-block Implementation

This section covers the steps on how to set up the Auto-block feature for a customer's domain.

Setting Up Auto-block Script Tag

To set up Auto-block, follow these steps:

1. Place the following scripts directly after the `<head>` tag of your website. Ensure that these are placed before any other script. Place the following code into all sites of your website.

    - **core.min.js** – This script contains the auto-blocking core functionalities which can block `script`, `img`, and `iframe` tags from being executed or loaded. This script must be placed before `blocklist.min.js`.

      NOTE: The `&ipaddress` is optional and is used in the sample script below for testing purposes.

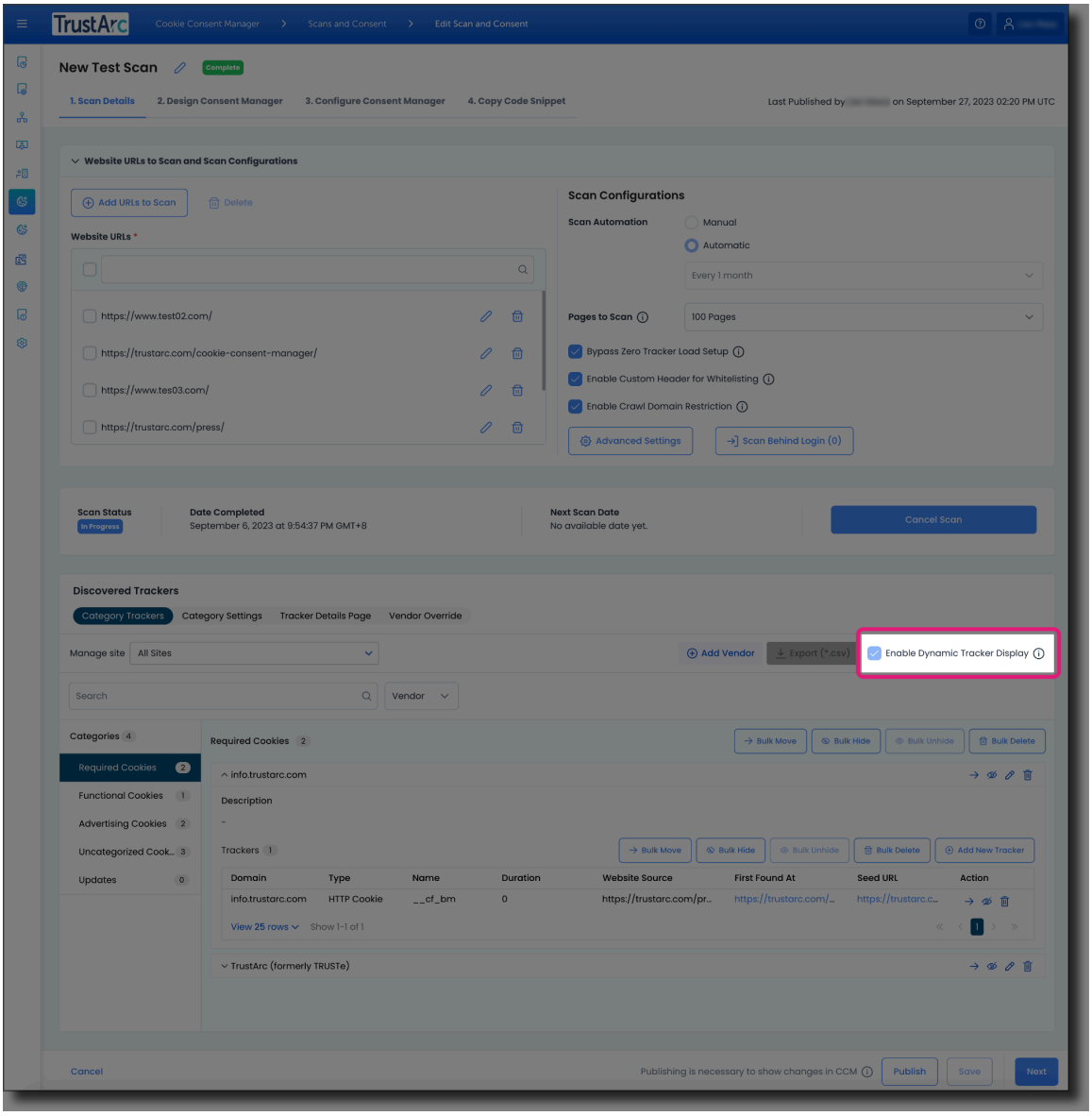
```
None
<script
src="https://consent.trustarc.com/v2/autoblockasset/core.min.js?cmId=<CM_ID>&ipaddress
=<IPADDRESS>"></script>
```

    - **blocklist.min.js** – This script contains the customer's list of categories and domains. Once this script is loaded, it will kickstart the process of auto-blocking.

      NOTE: The `&ipaddress` is optional and is used in the sample script below for testing purposes.

```
None
<script
src="https://consent.trustarc.com/v2/autoblock?cmId=<CM_ID>&ipaddress=<IPADDRESS>"></s
cript>
```

      NOTE: If the **Enable Dynamic Tracker Display** feature is enabled in step 1 of the Cookie Consent Manager creation process, you have two options to kickstart the process of auto-blocking.



- **Option 1** - Call the blocklist directly. The `fullURL` parameter is OPTIONAL. This parameter is only needed if the **referer header** does not match the site needed.

```
None
<script
src="https://consent.trustarc.com/v2/autoblock?cmId=<CM_ID>&ipaddress=<IPADDRESS>&full
URL=<FULL_URL>"></script>
```

- **Option 2** - This option is only needed if the **referer header** does not match the needed site and `fullURL` needs to be the URL of the current page.

```
None
<script>
    const fullurl = window.location.href;
    document.write(`<script
src="https://consent.trustarc.com/v2/autoblock?cmId=<CM_ID>&ipaddress=<IPADDRESS>&full
URL=${fullurl}"><\/script>`);
  </script>
```

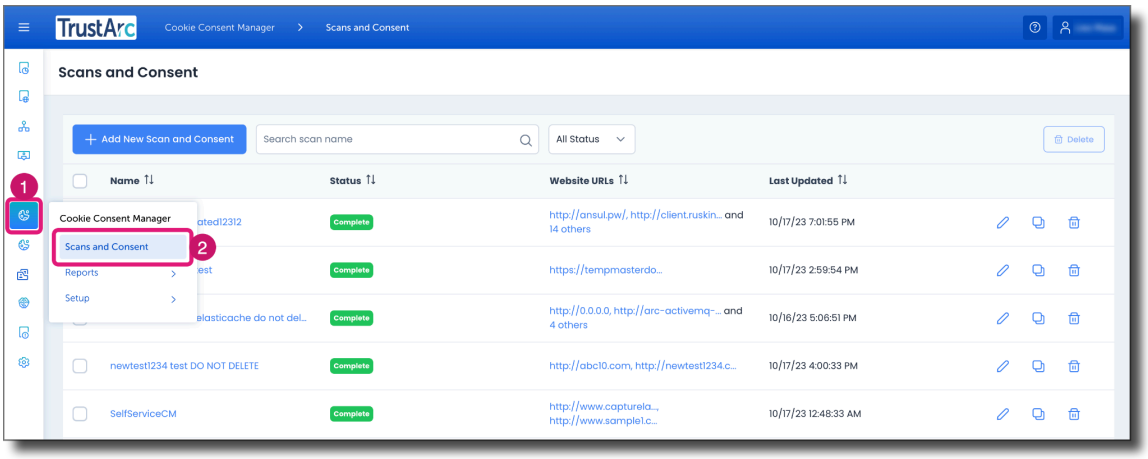
Example:

```
None
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Cookie Auto Blocker Test Page</title>

  <!-- This script tag will NOT be detected by the auto-blocker -->
  <script></script>

  <script
src="https://consent.trustarc.com/v2/autoblockasset/core.min.js?cmId=cakidv"></script>
  <script
src="https://consent.trustarc.com/v2/autoblock?cmId=cakidv&ipaddress=5.10.127.250"></s
cript>

<script
src="https://consent.trustarc.com/v2/notice/cakidv?ipaddress=5.10.127.250"></script>

  <!-- This script tag will be detected by the auto-blocker and will be checked if it
needs to be blocked -->
  <script></script>
</head>
<body>
<div id="teconsent"></div>
<div id="consent-banner"></div>

</body>
</html>
```

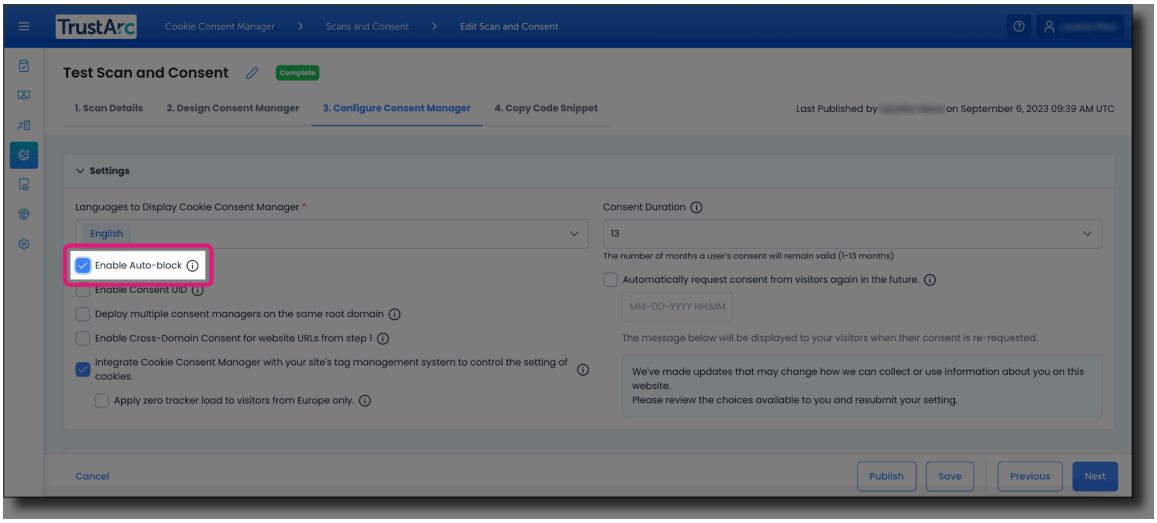
NOTE: The `ipaddress` parameter is for testing purposes only. User-facing implementation would rely on automatic geo-detection.

2. Enable the Auto-block feature for your domain in the CCM EU Portal.

    a. From the left side of the page, hover the mouse over the *Cookie Consent Manager* icon (1), and then click **Scans and Consent** (2).

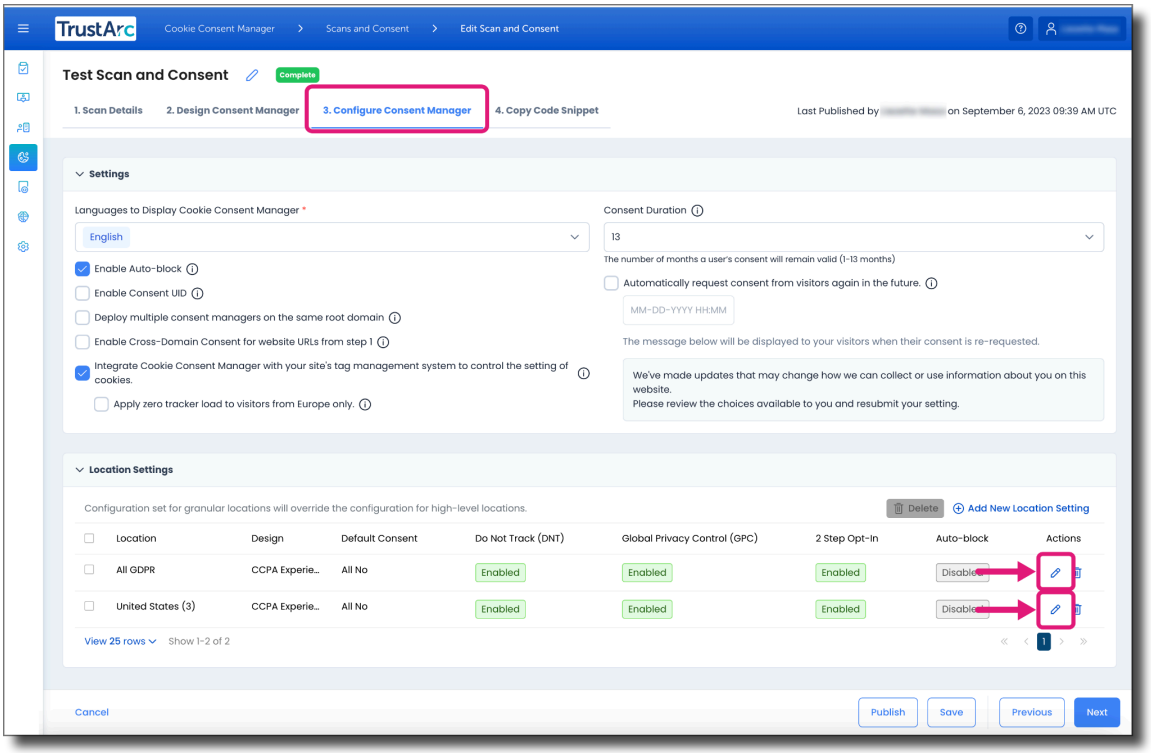


    b. Open the cookie consent manager. In the *Settings* section of the *Configure Consent Manager* tab, click the **Enable Auto-block** checkbox to enable the feature. For more information about this feature, hover over the tooltip button.

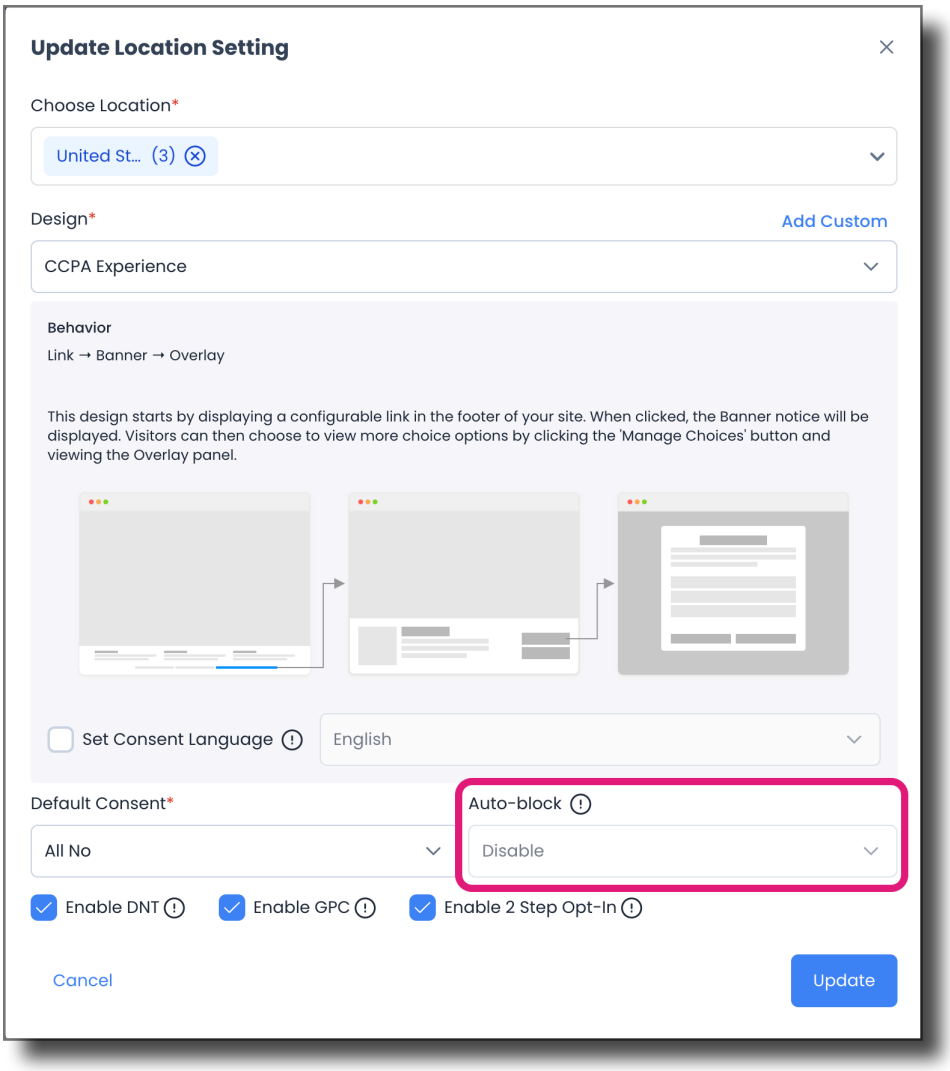


3. You can also disable or enable the auto-block feature per region, country, or state.

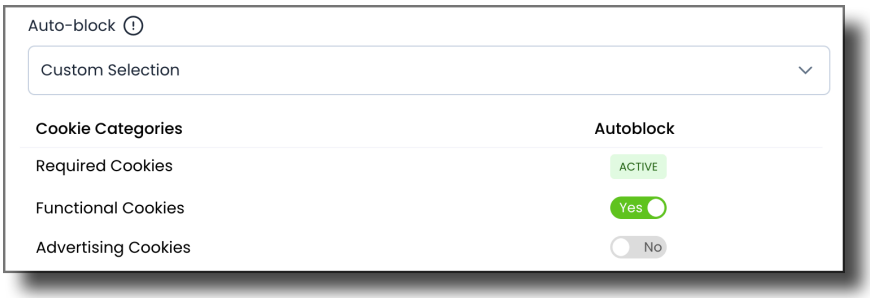
    a. In the *Settings* section of the *Configure Consent Manager* tab, click the *Edit* icon to the right of the region/country/state.



b. From the *Update Location Setting* modal, select **Enable** from the **Enable auto-block** dropdown list, and then click **Update**.



You can use the **Custom Selection** option to have more granular control over the firing of trackers by default, including the *Functional Category* trackers, until the user opts out. When this feature is enabled, the system displays the list of categories with Yes/No switches. When you select yes, the category is opted-in by default by Auto-block.



4. The auto-block feature will read the tracker categorizations from the Cookie Bucketing setup. Please ensure that the Cookie Bucketing process has been taken and mostly completed by the client. The `blocklist.min.js` file will determine the cookie categories and the tracking domains from the cookie bucketing step.
5. Wait for about an hour for recache. Your webpage should be ready once the recaching is complete.

Manual Block Implementation

This section covers the steps on how to set up the Manual block feature for your domain.

Setting Up Manual Blocking of Script Tag

This feature was provided to allow the customers the additional option to block tags whether or not the tags have an **`src`** attribute in the declaration.

In order to manually block a tag with `src`, set the `script type` of the tag script to **`ta-blocked`**. This only blocks scripts that you select to be blocked and suspends the load of the tag on your website until consent is given. Note that this script should be placed below the Auto-block script.

None
```
<!-- This is a manually blocked script tag with src and will not be executed until the user opts-in for the domain google.com -->
<script type="ta-blocked" src="https://google.com/script.js"></script>

<!-- This is a manually blocked script tag without src and will not be executed until the user submits a consent -->
<script type="ta-blocked"></script>
```

NOTES:

- If auto-block is disabled, manually blocked script tags will follow the natural behavior of the browser.
    - For script tag, type="ta-blocked" will cause the browser to block the script as detailed here. "*The embedded content is treated as a data block, and won't be processed by the browser.*"
- **`img`** and **`iframe`** are not supported for manual blocking. Type is not a valid attribute and will only be ignored.

Manually Blocking a Script Tag with a Domain

If you want to set a domain for a script tag that has no `src` attribute so that it can be unblocked if the user opts-in for that domain, then set the domain to the attribute `data-ta-domain`. Note that this script should be placed below the Auto-block script.

```
None
<!-- This is a manually blocked script tag with a domain and will not be executed
until the user opts-in for the domain; for example, google.com -->
<script type="ta-blocked" data-ta-domain="google.com">
  // some code here
</script>
```

Ignore Tags for Autoblock

You can mark tags managed by Google Tag Manager as *"ignored"* so that the autoblock feature does not block them.

Add `data-ta-type="ignore"` to ignore a `script`, `img`, or `iframe`. The ignored tags will not be processed by autoblock.

```
None
<script data-ta-type="ignore" src="https://google.com/script.js"</script>
```

Limitations

- `document.createElement('img')` , `document.createElement('iframe')`, and `new Image()` are not supported.
- Tags that are called/created inside an ignored script will not be ignored.

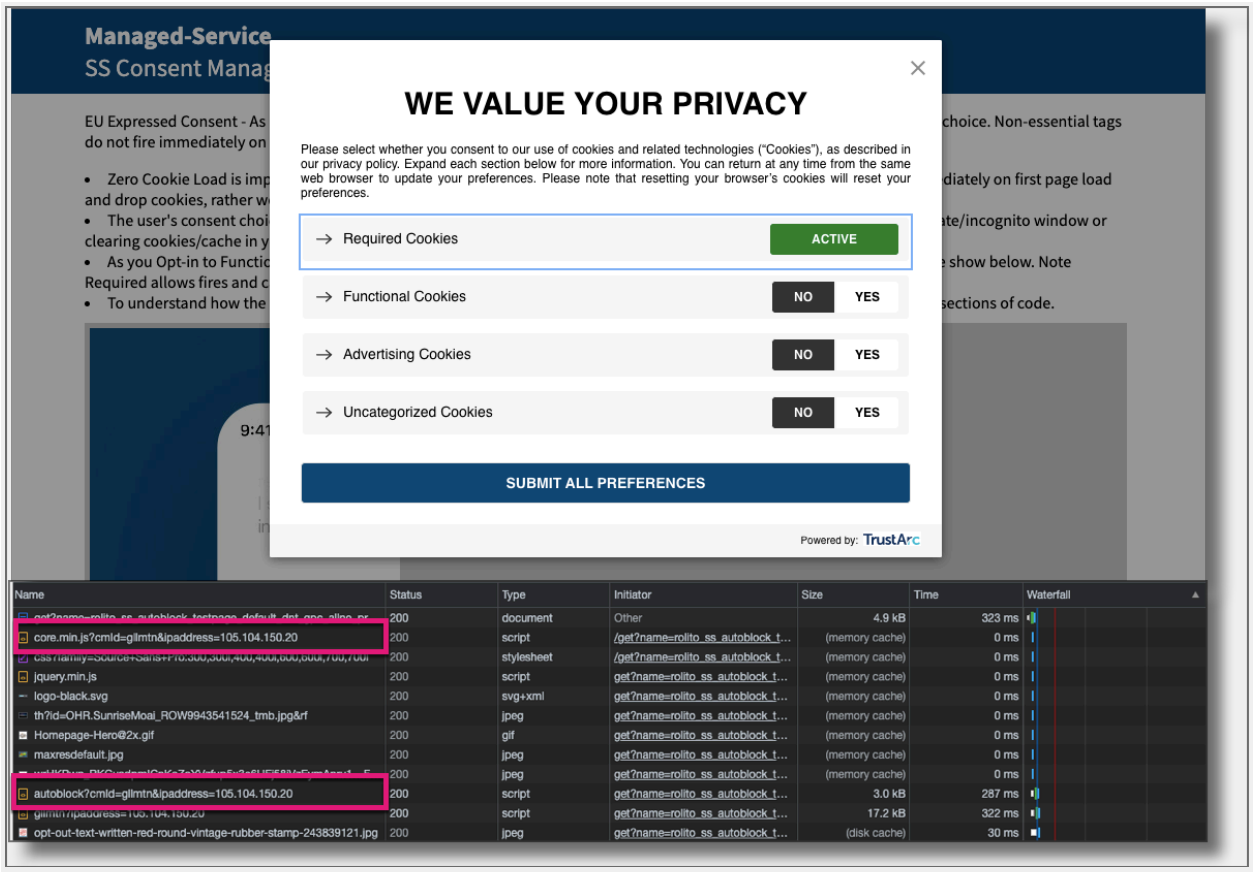
Validation

This section covers the steps on how to validate that the Auto-block feature functions as intended.

Verifying Auto-block Works

To ensure that the cookies are blocked after Auto-block is configured for your domain, follow these steps.

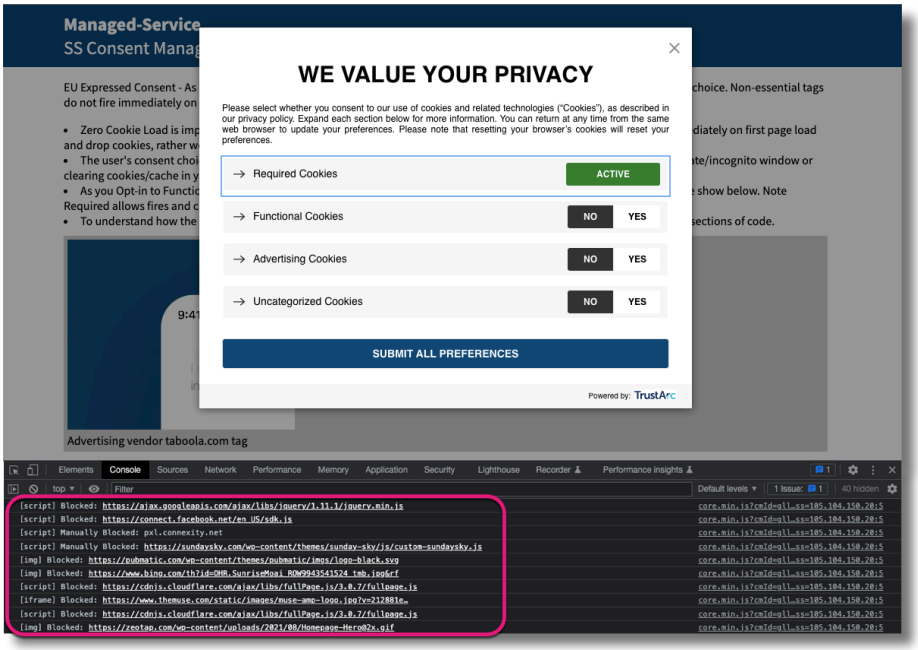
1. Load your website after clearing all the current cookies on the browser.
2. If you are using Google Chrome, right-click on the page and choose **Inspect**. This opens the Developer Console.
3. In the *Network* tab, check that the below 2 files have been called:
   - `core.min.js?cmId=<CM_ID>&ipaddress=<IPADDRESS>`
   - `autoblock?cmId=<CM_ID>&ipaddress=<IPADDRESS>`



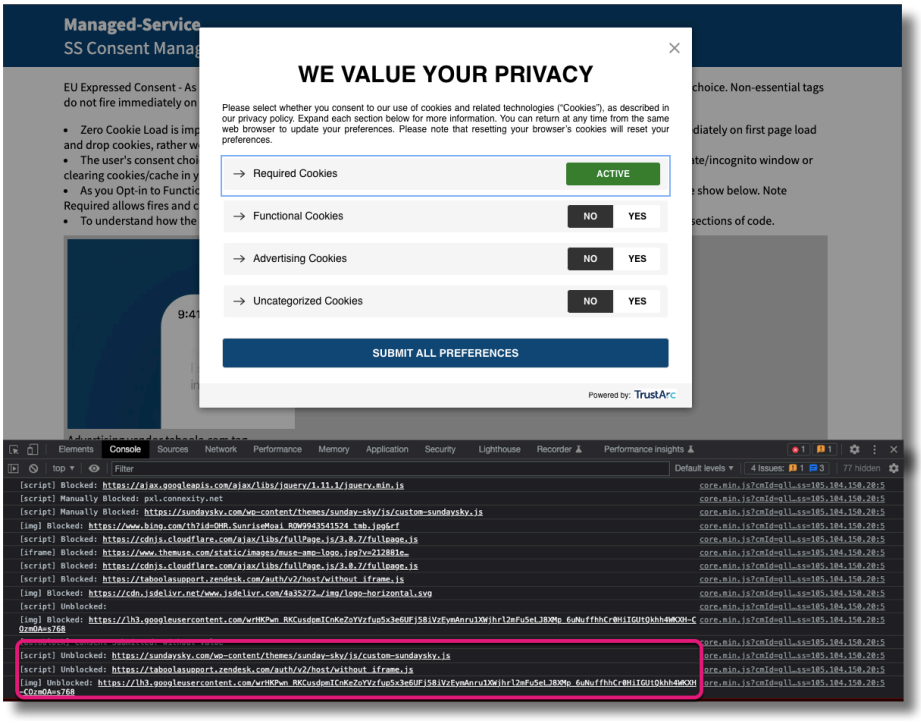
4. Click the *Console* tab to review which tags are blocked. The *Console* tab logs messages of what

tags are blocked and unblocked.

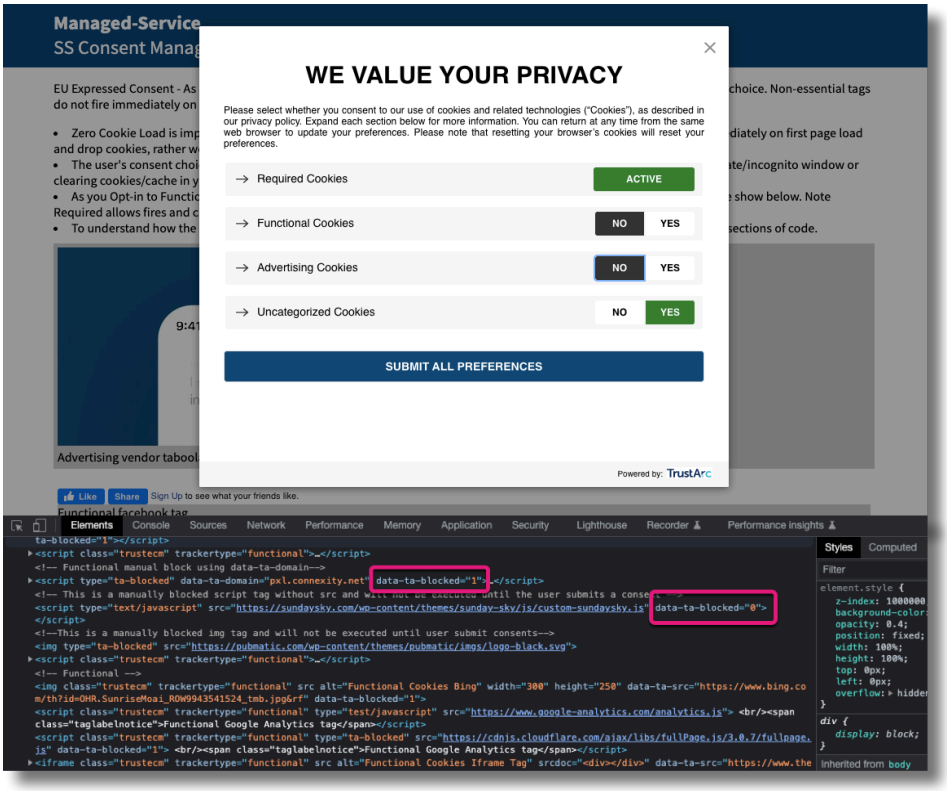
Blocked Tags



Unblocked Tags



5. In the *Elements* tab, a `data-ta-blocked` attribute is displayed. If `data-ta-blocked="1"`, this indicates that the tag is blocked. If `data-ta-blocked="0"`, this indicates that the tag is unblocked.



- For `script` tag, the type is set to "**`ta-blocked`**".

  Example:

```
None
<script src="https://google.com/script.js" type="ta-blocked"
data-ta-blocked="1"></script>
```

- For `img` tag, the actual `src` of the image is set to **`data-ta-src`** and the `src` will be empty.

  Example:

```
None
<img src="" data-ta-src="https://google.com/image.png" data-ta-blocked="1" />
```

- For `iframe` tag, the actual `src` of the image is set to **`data-ta-src`** and the `src` will be empty. The `srcdoc` is also set to "**`<div></div>`**".

  Example:

None

```
<iframe src="" srcdoc="<div></div>" data-ta-src="https://google.com/iframe.html"
data-ta-blocked="1"></iframe>
```

6. You can also check the UI which tags are blocked or unblocked. Upon first load, the tags should not be displayed on the page, except for *Required Cookies* tags. The Required tags are loaded because trackers under **Required Cookies** should not be blocked.

   If a user opts in to *Functional Cookies* or *Advertising Cookies,* then tags for these cookies should be unblocked or fired immediately. If a user opts out of Functional Cookies or Advertising Cookies, the tags will be blocked after the page is reloaded/refreshed.

   Please refer to the User Flow section to see the complete control flow for testing purposes.

Page Reload to Block Tags

Auto-block can automatically unblock tags when a user consents, but a page reload is needed to block tags.

Compliance Disclaimer: A page reload can be necessary for compliance purposes so that tags that should be blocked after an opt-out do not continue to collect personal data from a user.

Page reload after submitting consent choice from Modal window

Include the following script along with your CCM script to induce a page reload when the customer opts out via the modal popup window. This can reside in the HEAD or loaded via a tag manager.

JavaScript

```
<script>
window.addEventListener("message", function(event) {
    var eventDataJson = null;

    // We only care about TrustArc Events at this point. And TrustArc's even it
encoded in JSON
    try {
        eventDataJson = JSON.parse(event.data);
    } catch (e) {
        // Some other event that is not JSON.
        // TrustArc encodes the data as JSON
        // console.log(event.data);
    }

    // Safeguard to make sure we are only getting events from TrustArc
    if (eventDataJson && eventDataJson.source === "preference_manager") {
        // Means that the user has submited their preferences
        if (eventDataJson.message === "submit_preferences") {
            setTimeout(function() {
                window.location.reload();
            }, 20);
        }
    }

}, false);
</script>
```

Page reload after submitting consent choice from Banner

Include the following Script along with your CCM script to induce a page reload when the customer opts out from the cookie banner. This should reside in the BODY.

```
JavaScript
<script>
document.body.addEventListener("click", function(event) {
    if(event && event.target && event.target.id === 'truste-consent-button') {
       setTimeout(() => { window.location.reload(); }, 1000);
    }
});
</script>
```

Limitations

Below are some of the known limitations of the Auto-block feature.

- Auto-block has been tested to work as a standalone feature. We will verify its compatibility with existing Tag Management solutions and deployments in our future releases.
- Auto-block currently does not support the feature “deploy unique consent managers on the same root domain”. Auto-block will recognize the consent from the root domain level and block tags. We will support this feature in a future release.
- Auto-block can only work if the Auto-block script is placed inside the `<head>` tag of the website code. If it is installed under any other `script`, `img`, or `iframe`, the above `script`, `img`, or `iframe` cannot be blocked before loading.
- Auto-block code currently only identifies 3rd party tags of type `script`, `img`, or `iframe.`
- Auto-block only works with modern browsers (Chrome v.70+, IE 11, Edge 42+, Firefox v.60+, Safari v.11+).
- In certain scenarios, JS files will still be downloaded and will have a 200 status in the network (but it will not get executed).Cookies set via header of these calls are not able to be intercepted. Please use manual block in this case.
- If a tag/tracker is listed in 2 buckets and one bucket selects opted-in for that exact tracker, then the default would be opted-in for all categories. It must be an exact match for this scenario to

occur.

For example:

A user provides consent for Functional Cookies, but opts out of Advertising Cookies.

linkedin.com is in *Functional* bucket, **opted-in**.
linkedin.com is in the *Advertising* bucket, **opted-out**.
ads.linkedin.com is in the *Advertising* bucket, **opted-out**.

linkedin.com will be **opted-in** because it has been double-listed.
ads.linkedin.com is **opted out** because it is single listed in the *Advertising* bucket and will adhere to the opted-out consent.

The CCM UI will reflect **opted-in** for linkedin.com (Functional bucket is **opted-in** if you only list bucket opt-outs) and **opted-out** for Advertising bucket.

EasyView

This section is intended for TrustArc Technical Account Managers looking to test or demonstrate Cookie Consent Manager scripts without the need to build and upload an HTML file with the appropriate script and tags.

Overview

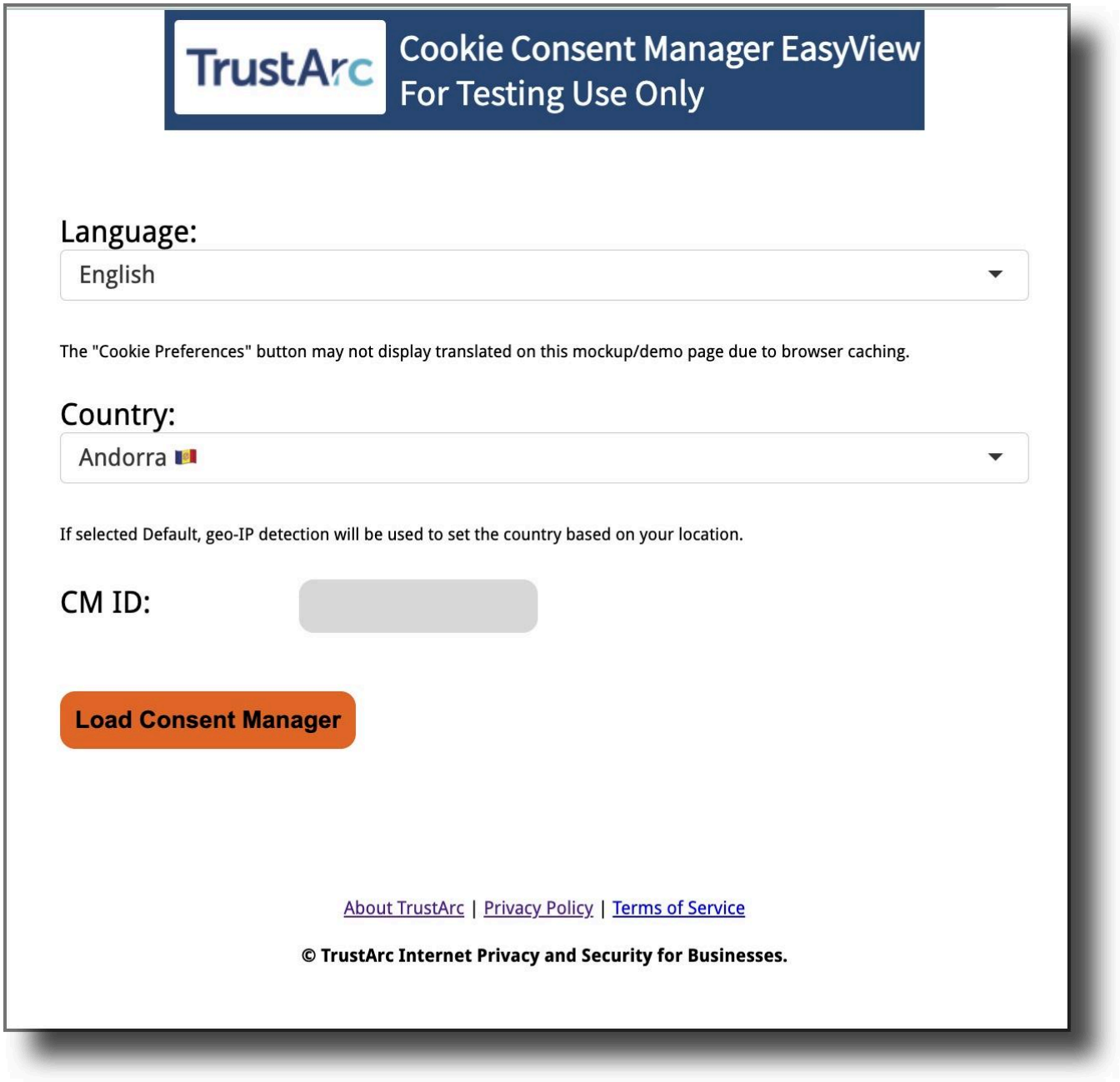
TrustArc's Cookie Consent Manager (CCM) leverages a script tag with a number of required and optional variables depending on the desired behavior. Implementation requires installation of this script tag as well as, in most cases, one or more DOM elements with specific IDs for use of the Cookie Preferences button/link (`#teconsent`) as well as the Cookie Consent Manager banner (`#consent-banner`).

To test CCM, these tags need to be added to a web page with the necessary parameters and elements. This tool is designed to simplify the process of creating test pages with CCM implementations for testing or demonstrating specific features. However, for demonstrating a finalized implementation plan, static HTML pages should still be provided.

This tool allows for the quick setting of common variables like country, language, and CM ID on a dynamically generated page. Not all variables or CCM permutations can be tested with this tool. Once the page is generated, it can be shared as a URL in one of several countries.

Settings

The various drop-downs and text boxes inside easy view control which DOM elements are rendered. None of the inputs are considered and no script will run unless and until the "**Load Consent Manager**" button is clicked.



Language

The Language setting is set to English by default, which means that the browser's language preference is not taken into account when using this tool. Instead, the language displayed in the tool will be the one

specified in the dropdown menu.

If the selected language is not enabled in Step 3 - Configure Consent Manager - Languages to Display Cookie Consent Manager, then the content in the modal and banner will be displayed in English. If the selected language is enabled but some text doesn't have translations available, those parts will also be displayed in English.

Country Code

The Country Code setting is a drop down, which means that once a specific Country and CCM are selected, the expected behavior (explicit or implied) will be determined as defined in Step 3 - Configure Consent Manager - Location Settings. If a country is indicated, the behavior linked to a particular country/region will be displayed regardless of the visitor's location.

CM ID

The CM ID setting is a free-form text field where you should provide the appropriate CCM ID value. This unique identifier can be found in Step 4 - Copy Code Snippet in the Cookie Consent Manager Portal https://ccm.trustarc.com/. It's important to note that if you provide an invalid CM ID, no script will load.

If no script is loaded after completing the required fields and clicking "**Load Consent Manager**," the following scenarios are the possible reasons:

- Country is not provisioned in this CM Instance.
- Incorrect CM ID value used.

Stealth Auto-Executing Link

In order to easily store and share Cookie Consent Manager setups, once you click the "**Load Consent Manager**" button, there will be a link that you can use internally. Loading this link will render a page with the previously set options, without the need to click the "**Load Consent Manager**" button, but while also hiding all of the configurations/settings. This may be helpful in avoiding confusion when you just want to share the Cookie Consent Manager behavior or look. On this page, you can optionally, n press **[control] + [v]** in order to show the settings.

Appendix

Supported Browsers and Versions

Visitors to your website

- Microsoft Edge 14+
- iOS 12.0 and higher for mobile
- Safari 12.0 and higher for mobile
- Safari 12.0 and higher for desktop
- Android OS 8.0 and higher
- Chrome version 70+ for mobile
- Chrome version 80+ for desktop
- Firefox version 70+ for mobile and desktop

Users of the Cookie Consent Manager portal

- Chrome (latest version)
- Microsoft Edge (latest version)
- Firefox (latest version)
- Safari (latest version)