Digital Control Room Ltd
www.digitalcontrolroom.com
info@digitalcontrolroom.com

Digital Control Room

Cookie Audit Service - Explicit Consent

February 2024

Table of contents

Introduction

The DCR cookie solution fully supports and enforces explicit consent for compliance with GDPR, CCPA/CPRA and other similar regulations. Our strategy is to prevent cookies being set by deactivating the script tags that initiate the setting of cookies (and other tracking technologies) until such time that the visitor has consented by opting in.

Full compliance requires researching unknown cookies (see Categorising of Unknown Cookies) and implementing explicit consent via any of the following methods:

1. **Tag Managers**
   Centralised control of tag management has many advantages for compliance and IT. We have working solutions with Google Tag Manager and Tealium. Other tag managers are also very likely to work.

   Google Tag Manager
   Please see Google Tag Manager for a step by step guide.

   Tealium
   Please see Tealium Tag Manager for a step by step guide.

2. **Server side code**
   Cookies set via the backend server code that are not functional are also subject to consent checks. We supply sample code in a few popular languages.

   Please see Server Side Consent

3. **HTML templates**
   The addition of DCR's attributes around script tags is a simple method to enable and disable tags automatically as consent is given/withdrawn.

   Please see HTML Templates

4. **Events**
   The DCR panel code triggers events which page authors can listen for, interpret and then make dynamic changes to the page.

   This can be used instead of deactivating script tags and instead injecting them to the page once the site visitor has opted-in.

   Please see Events for a full listing of events and their properties.

Logging of consent

To comply with the GDPR requirement to be able to demonstrate consent, DCR logs the consent selections made by your site visitors. The data points we store are as follows:

- Site visitor reference UUID
- UTC date and time
- Source address
- User agent (first 60 characters)
- Referrer (Hostname only)
- Policy storage key and language
- Consent selection and method (eg banner or accordion)

Authenticated users

If your site visitor authenticates with your backend systems your code can read our “preference cookie” or event payload to obtain the site visitor reference and store it in your database against the user - thus linking a specific consent proof with a real world user.

Over time the user may generate multiple tracking records, for example by logging in via different devices or intentionally clearing cookies. Therefore the database must accept many site visitor references for the user, ideally stored alongside a timestamp and optionally a copy of the consent preferences made by the user.

Categorising Unknown Cookies

Digital Control Room maintains a Common Cookie Database (CCDB) containing a large number of cookies and their unique attributes. When a website scan finds a cookie that does not have an entry in the CCDB, it is classified as an unknown cookie.

Cookies matching with the CCDB are placed in the appropriate category as defined by their purpose. Unknown cookies are instead placed in the "Strictly Necessary" category – this is in case one of these cookies are required for essential website functionality.

Digital Control Room frequently researches unknown third party cookies for inclusion in the CCDB. An organisation is responsible for researching the remaining unknown cookies which may be unique to the site or whose purpose cannot be determined. If the cookie is not familiar, work with your development teams or agencies to identify the cookie, its purpose and a public facing description. It is important to correctly classify cookies as soon as possible so that you can report an accurate policy to your visitors and implement their consent preferences.

In your policy, please check for any cookies called "Under review" and for each cookie supply to DCR:

1. The cookie's purpose (see table below),
2. The name of the company setting the cookie,
3. A site visitor friendly description informing your visitors about the cookie's usage.

   Note: Please supply descriptions for all applicable policy languages.

This will ensure that the cookies are placed into the correct category for consent.

Category	Purposes
Category 1: Necessary	Your visit Authorisation
Category 2: Site experience	User preferences Navigation
Category 3: Performance & operation	Analytics Performance / Networking
Category 4: Marketing, anonymous cross site tracking	Survey Advertising Social networking
Category 5: Marketing, targeted advertising	Advertising Social networking

Google Tag Manager

Google Tag Manager is an advanced solution supporting multiple methodologies for withholding tags within a container until consent is received. Of the many possible implementation methods, we have outlined three common options below which may suit your environment:

1. Trigger tags based on a DataLayer Variable
2. Trigger tags based on the value of our consent cookie
3. Trigger tags based on Consent Mode V2

Trigger tags based on a DataLayer Variable

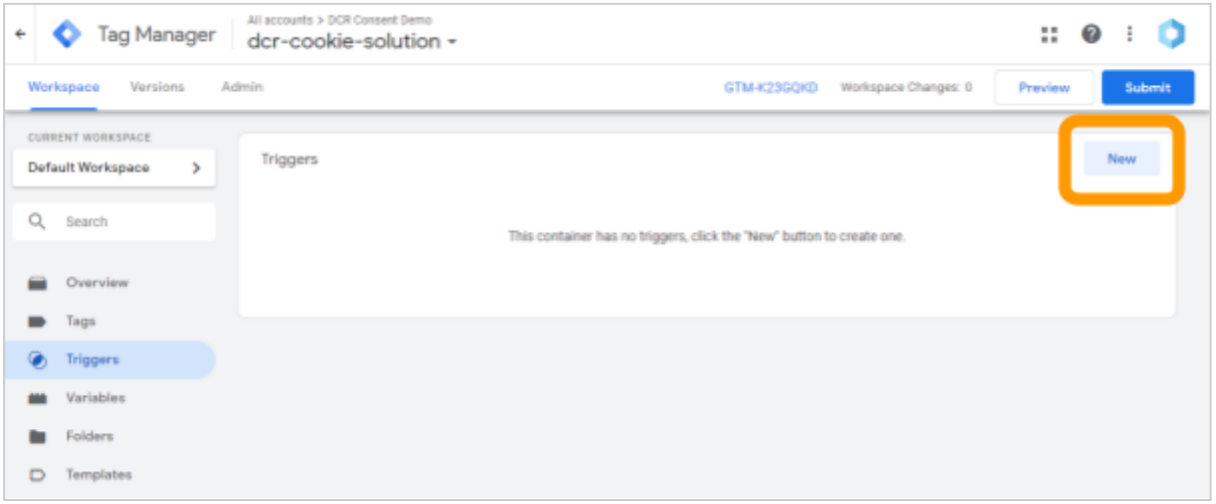
To integrate with the DCR solution we need to check the value of the Data Layer Variable that sends the preferences of a visitor whenever they are changed.

The following guide creates a trigger which can be applied to tags as required.

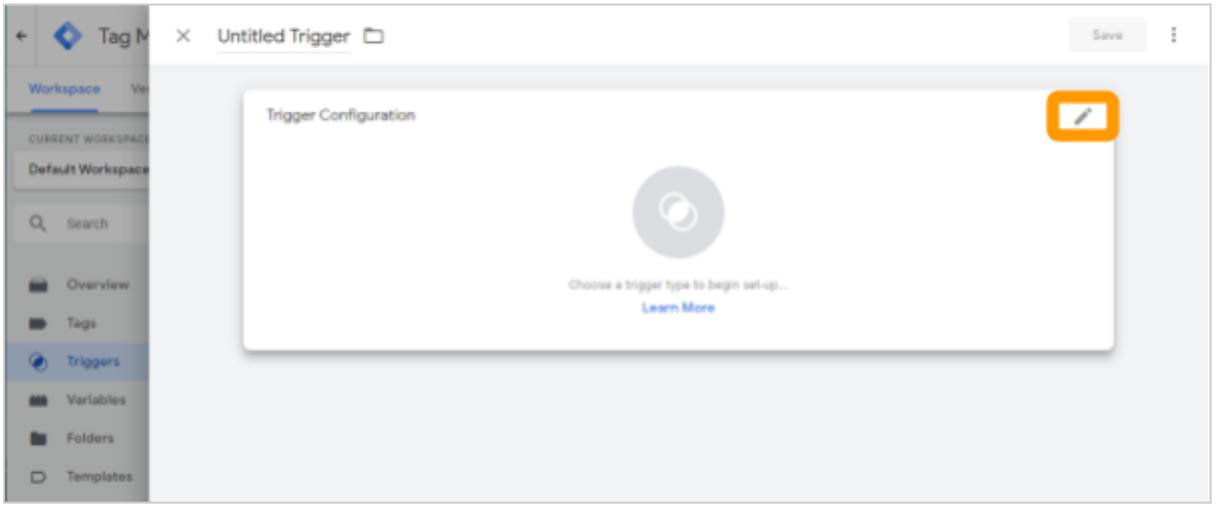
We issue the following `dataLayer.push()`

Data Layer Event:	`wscr.consent`
Arguments	`wscr` is an object with the following keys: • `wscr.consent` - a serialized string representing current opt-in consent, for example: `1=true&2=false&3=true&4=false&5=true`
Trigger:	Triggered after the visitor clicks the OK button to dismiss the banner or accordion

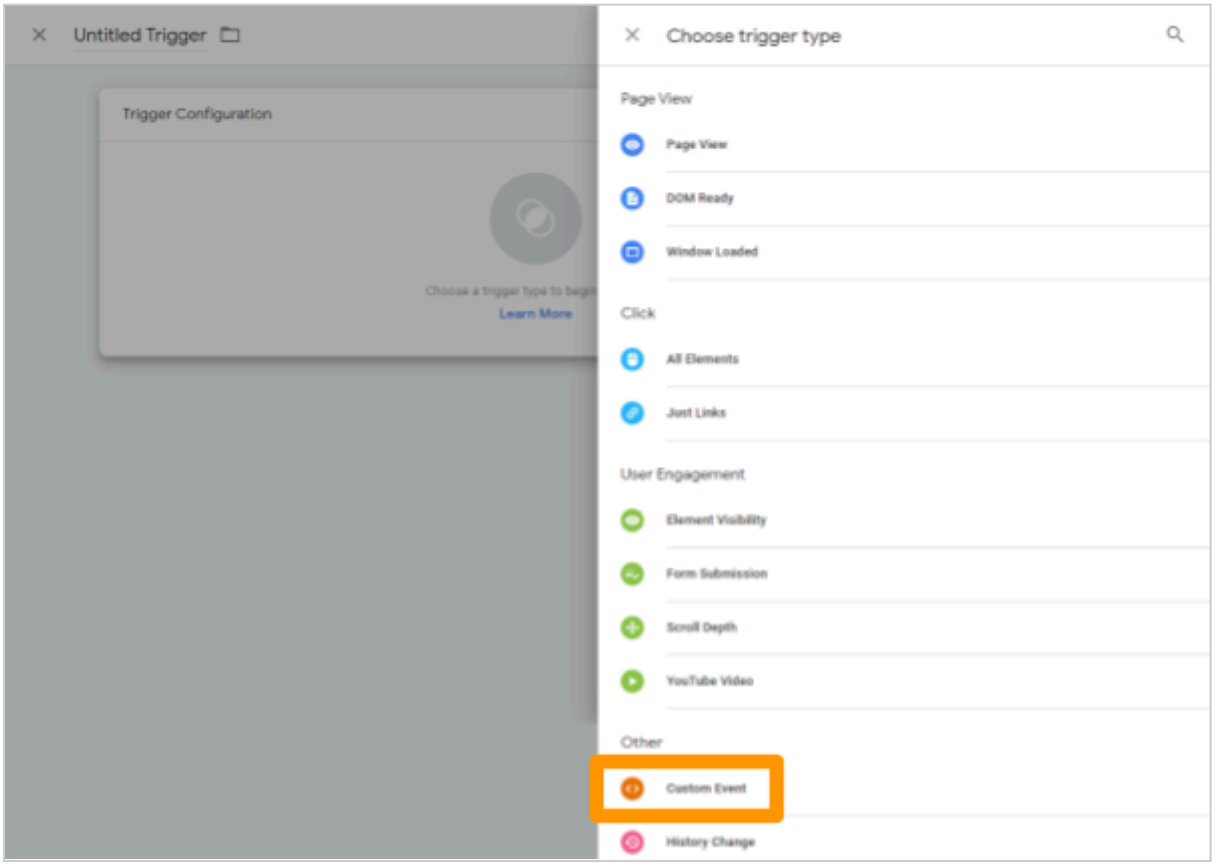
1 - Click “New” on the “Triggers” page



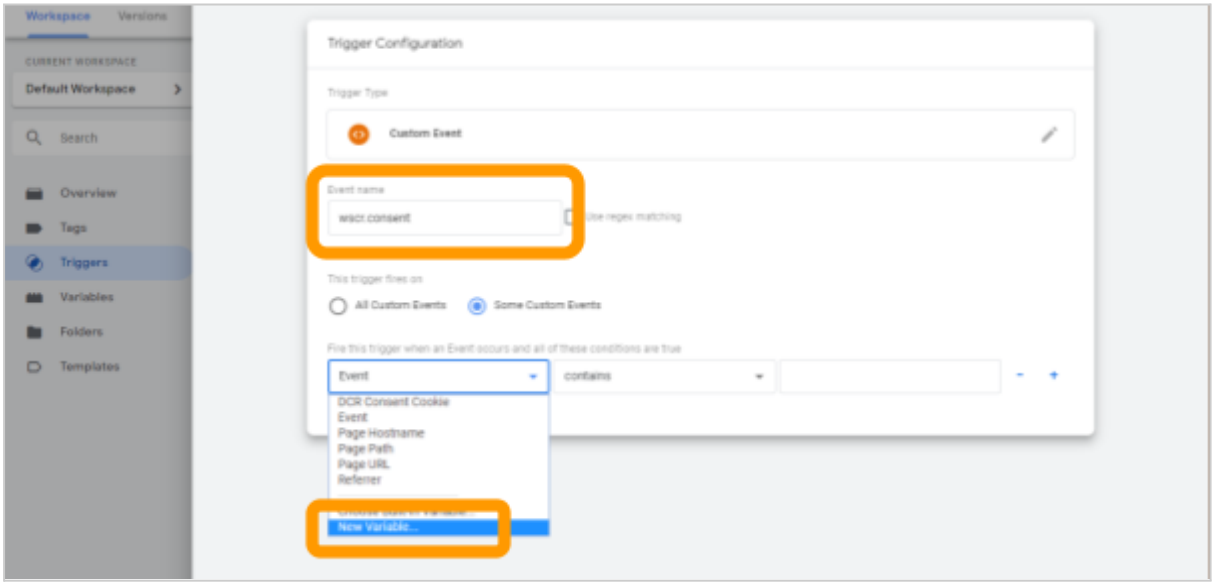
2 - On the new page click on the pencil icon to edit the Trigger Configuration:



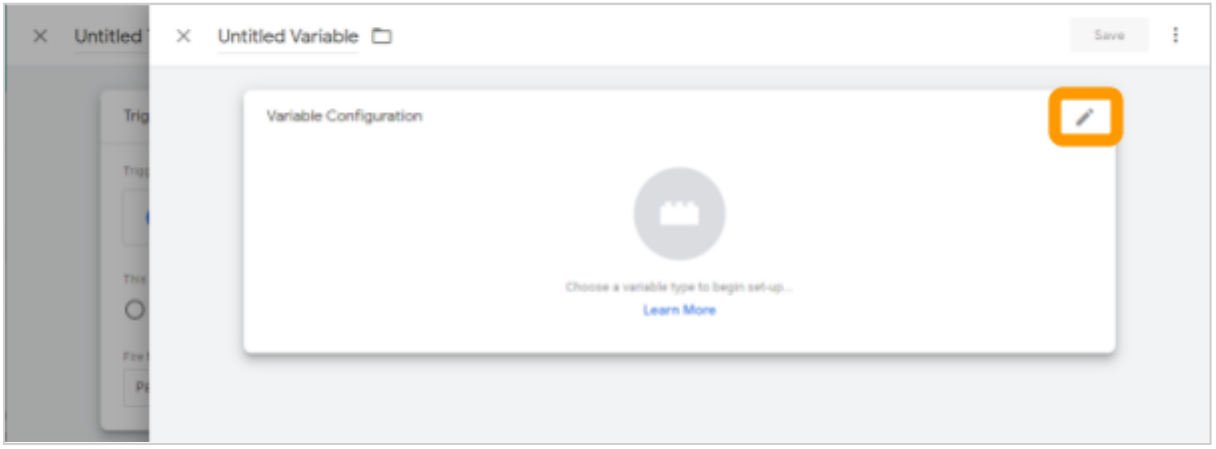
3 - Select “Custom Event” as the “Trigger type”:



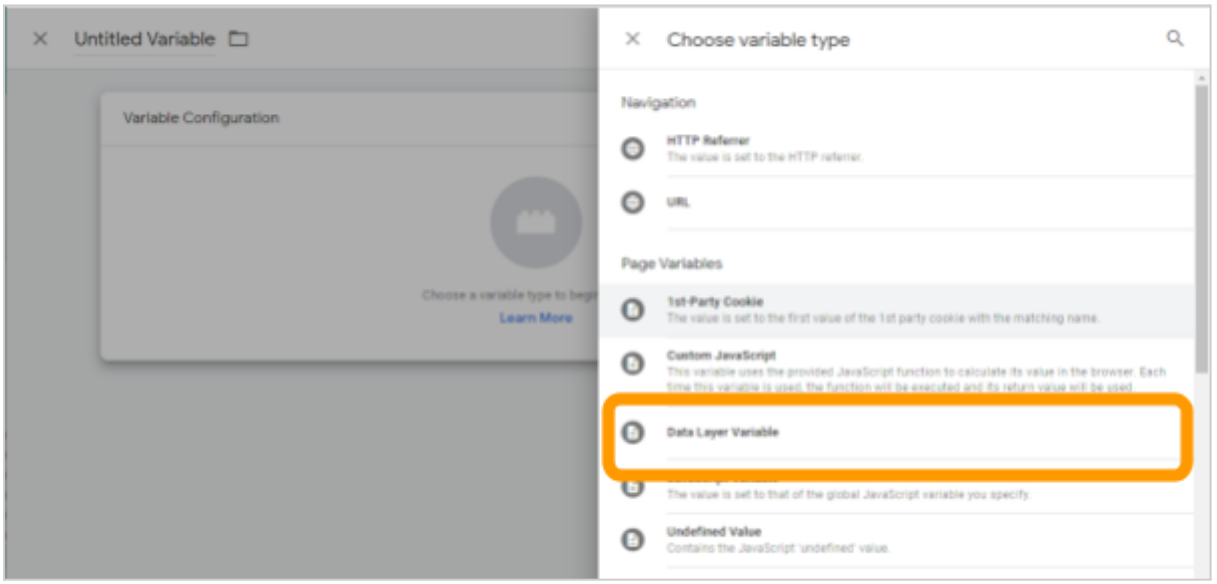
4. Enter “wscr.consent” as “Event name”, and select "This trigger fires on... Some Custom Events" and then ”New variable” from the first drop down of events:



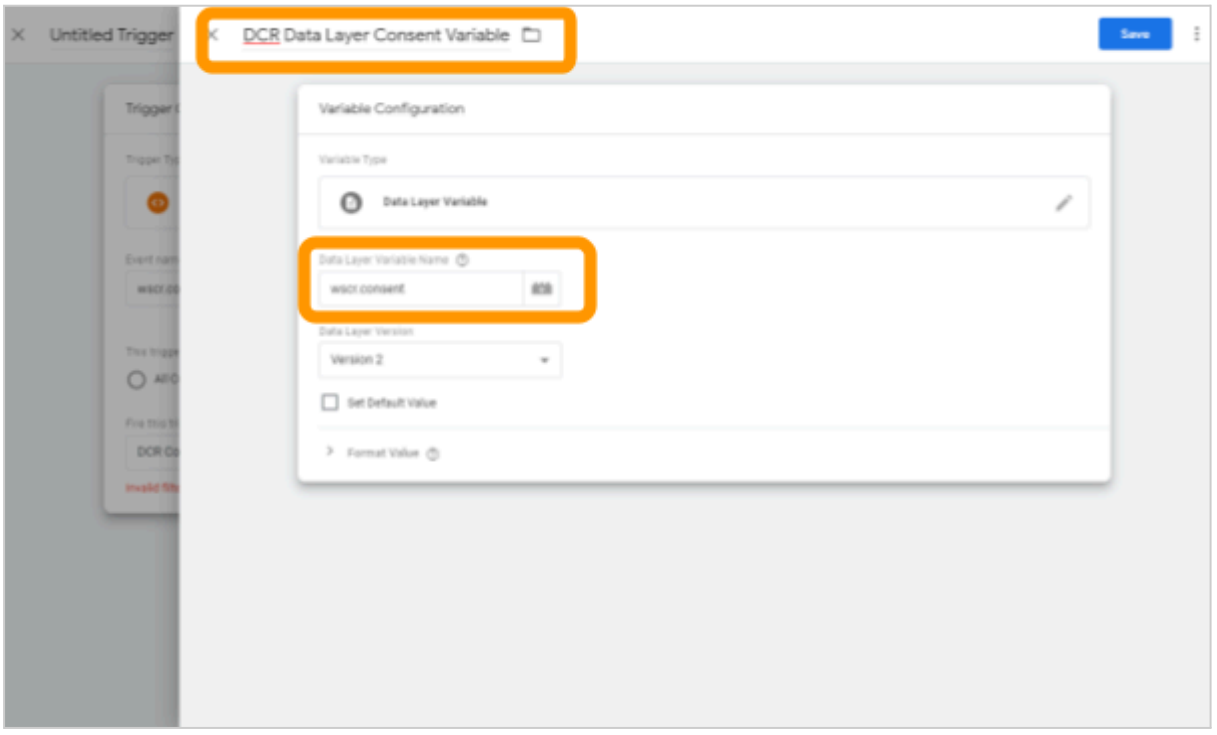
5 - On the new page click on the pencil icon to edit the Variable Configuration:



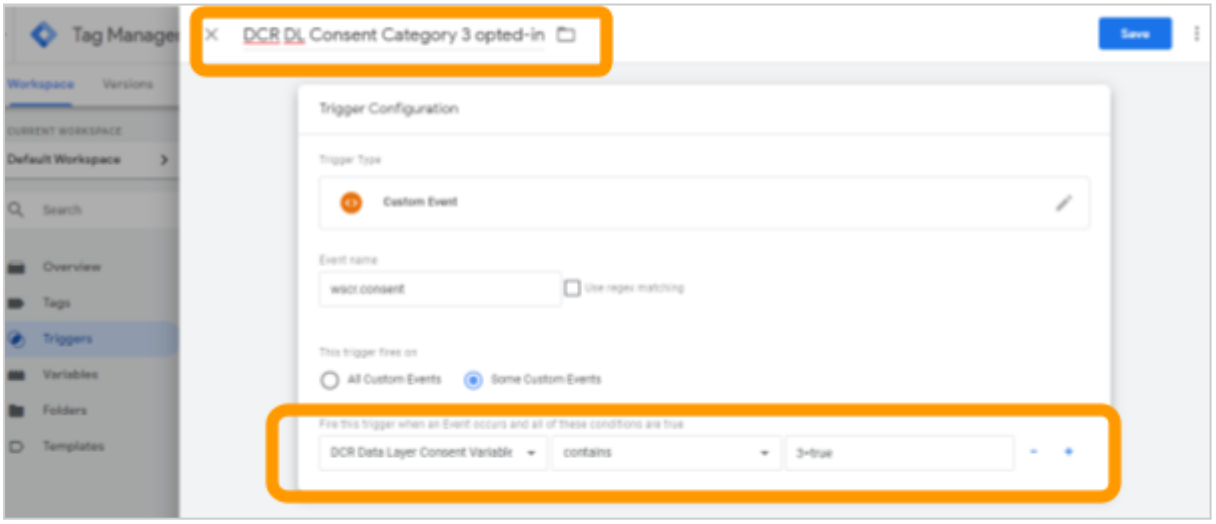
6 - Select “Data Layer Variable” variable type:



7 - Enter the “Data Layer Variable Name” as “wscr.consent” and then name the variable, we suggest “DCR Data Layer Consent Variable”:



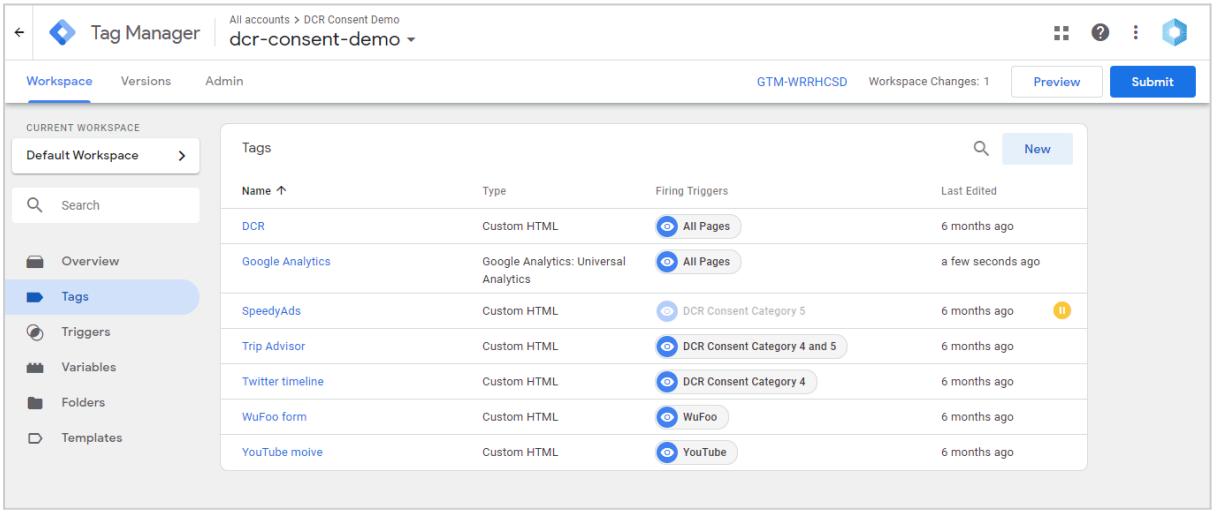
8 - After saving you will be taken back to the “Trigger Configuration” page. Leave the event as “DCR Data Layer Consent Variable” (that we just created), leave the check as “contains” and then enter the consent preference you wish to use. For example here we are checking that category 3 to be opted in and therefore entered “3=true” and named the trigger accordingly.



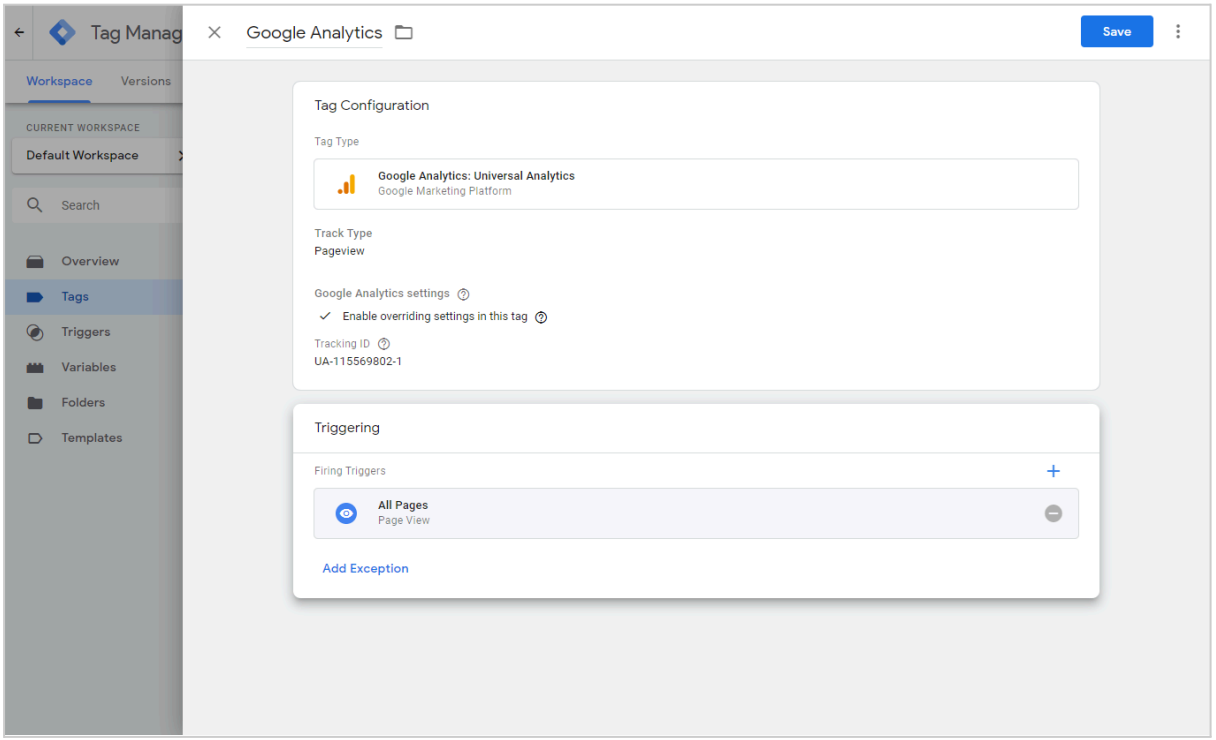
The values to check depend on the category you wish to enforce:

Category Designation	Title	State	Value to check
Category 1	Necessary	Opted in	`1=true`
		Opted out	`N/A`
Category 2	Site experience	Opted in	`2=true`
		Opted out	`2=false`
Category 3	Performance & operation	Opted in	`3=true`
		Opted out	`3=false`
Category 4	Marketing, anonymous cross site tracking	Opted in	`4=true`
		Opted out	`4=false`
Category 5	Marketing, targeted advertising	Opted in	`5=true`
		Opted out	`5=false`

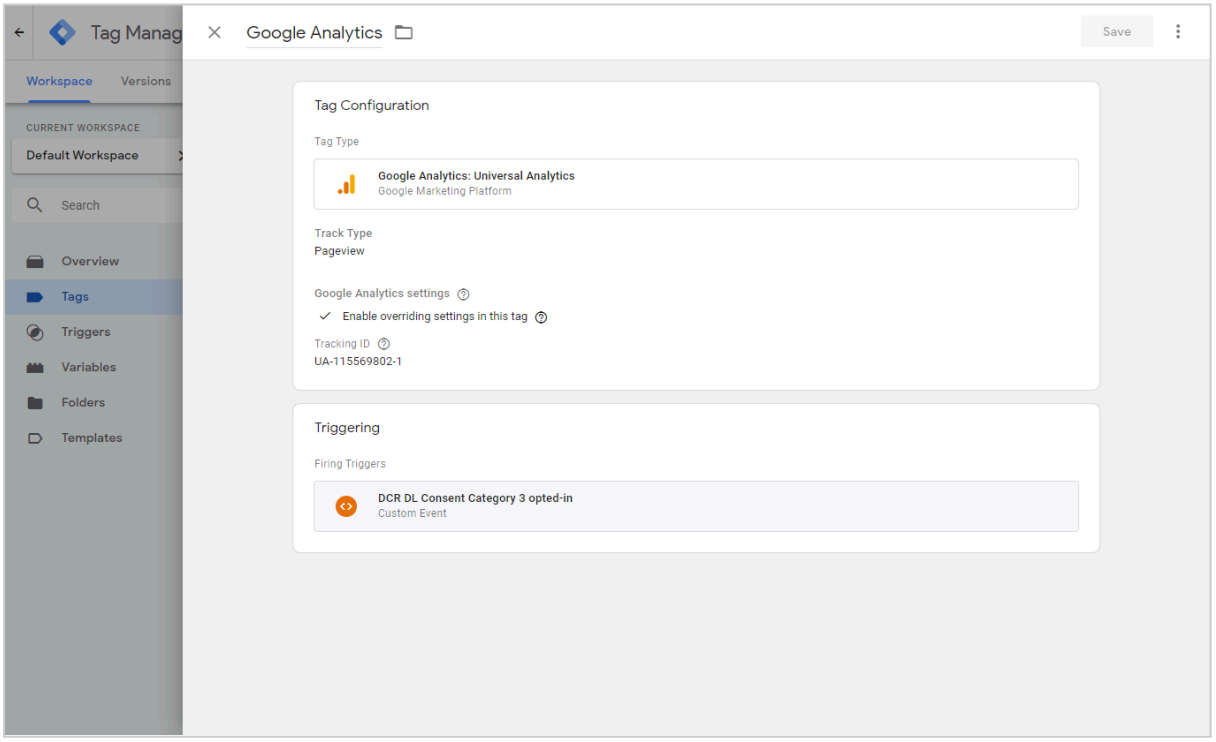
9 - Apply the trigger to your tags. Click on “Tags”:



10 - Select the tag you wish to restrict by clicking on the name:



Typically, as shown in the previous image, the trigger for a GTM tag is “All Pages” or similar. Click the Add sign and select the Data Layer Category Trigger you created earlier and then remove “All Pages”:



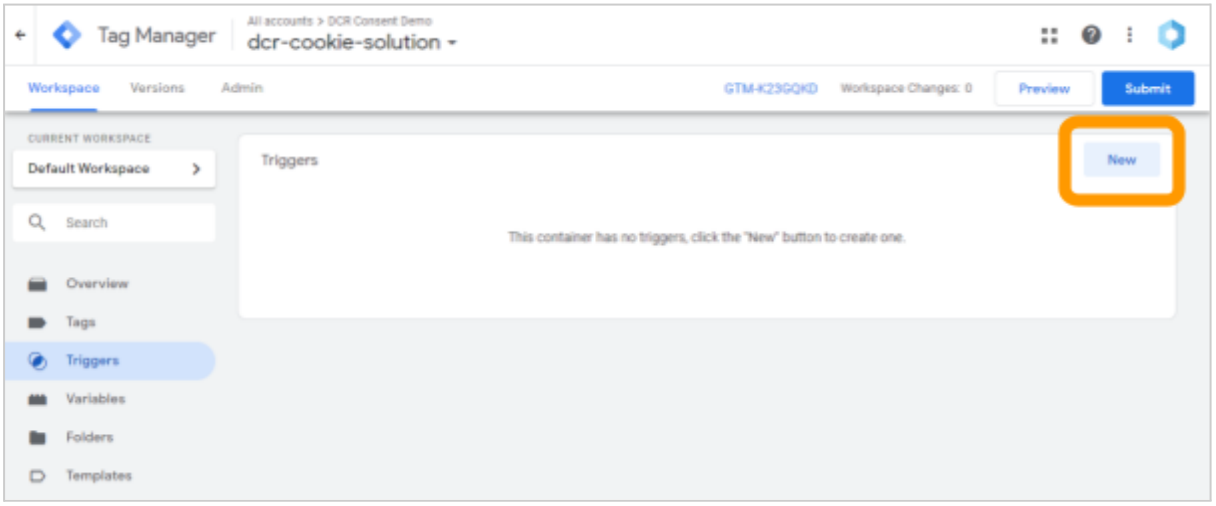
11 - Save

Trigger tags based on the value of our consent cookie

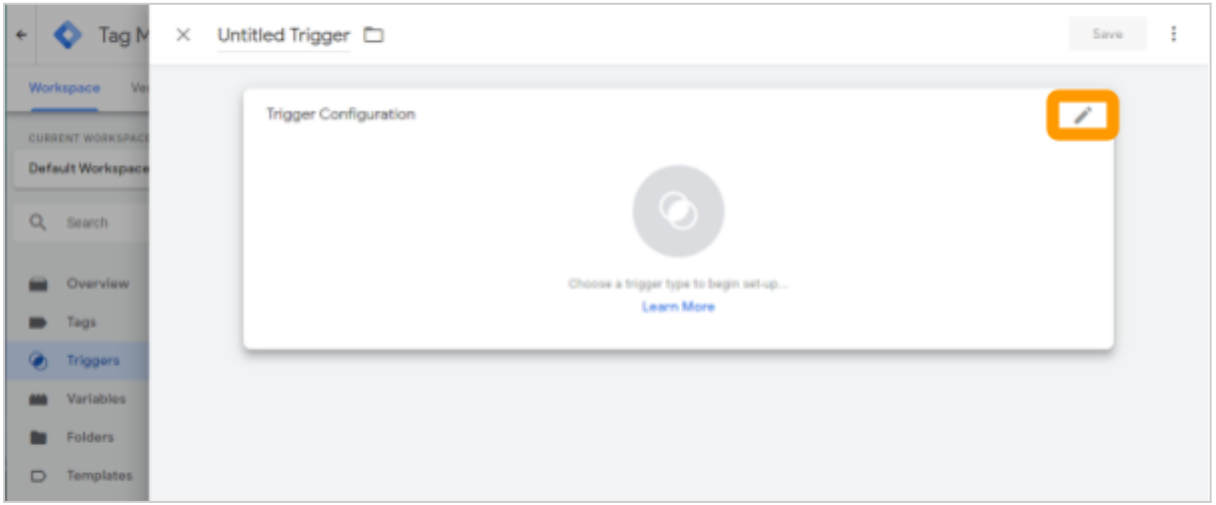
To integrate with the DCR solution we need to check the value of the cookie that stores the preferences of a visitor. The cookie is 1st party and named “wscrCookieConsent”.

The following guide creates a trigger which can be applied to tags as required.

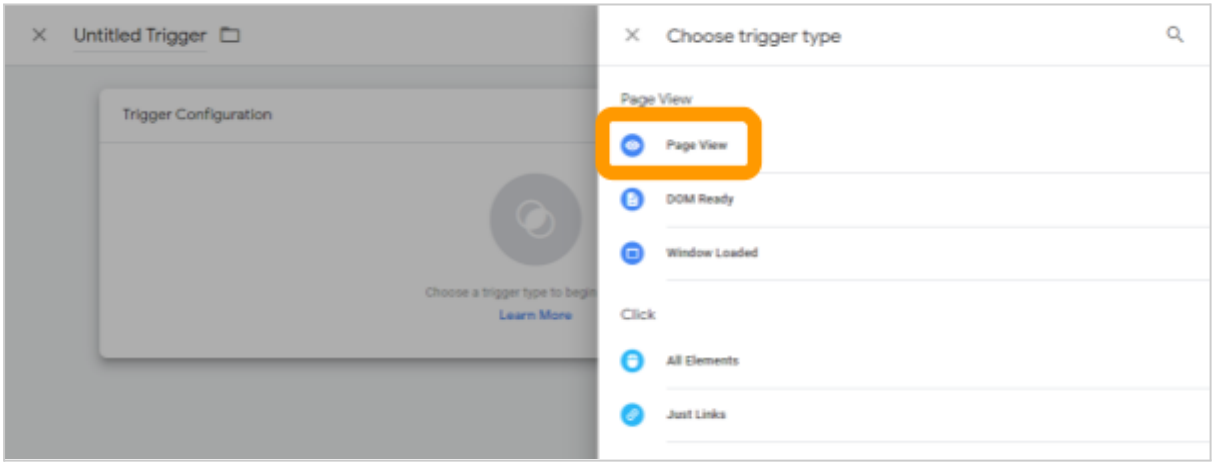
1 - Click “New” on the “Triggers” page



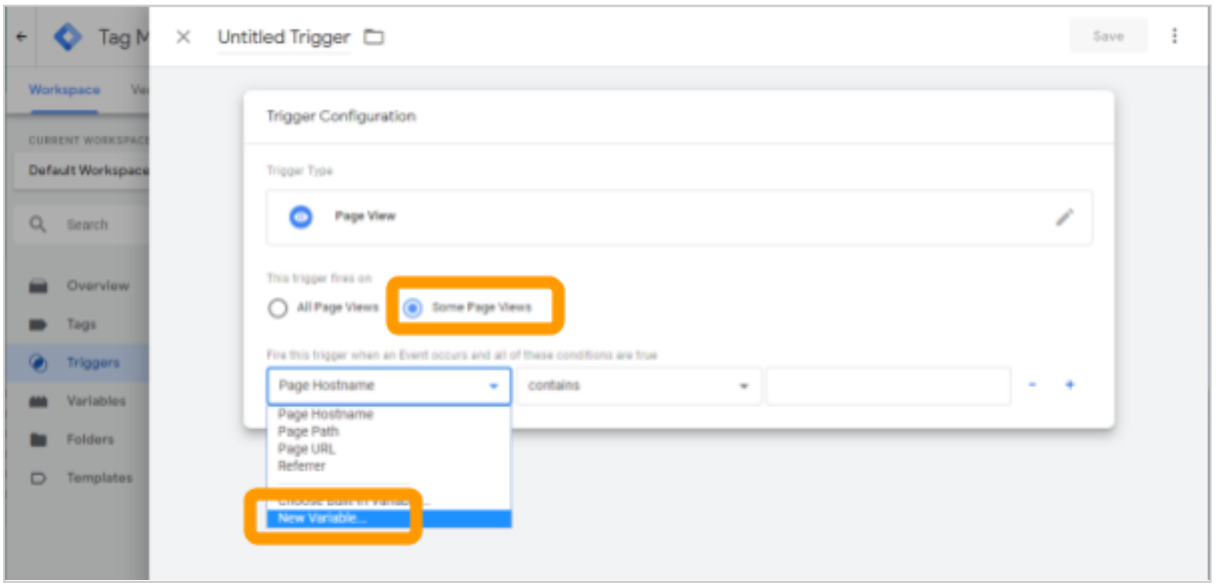
2 - On the new page click on the pencil icon to edit the Trigger Configuration:



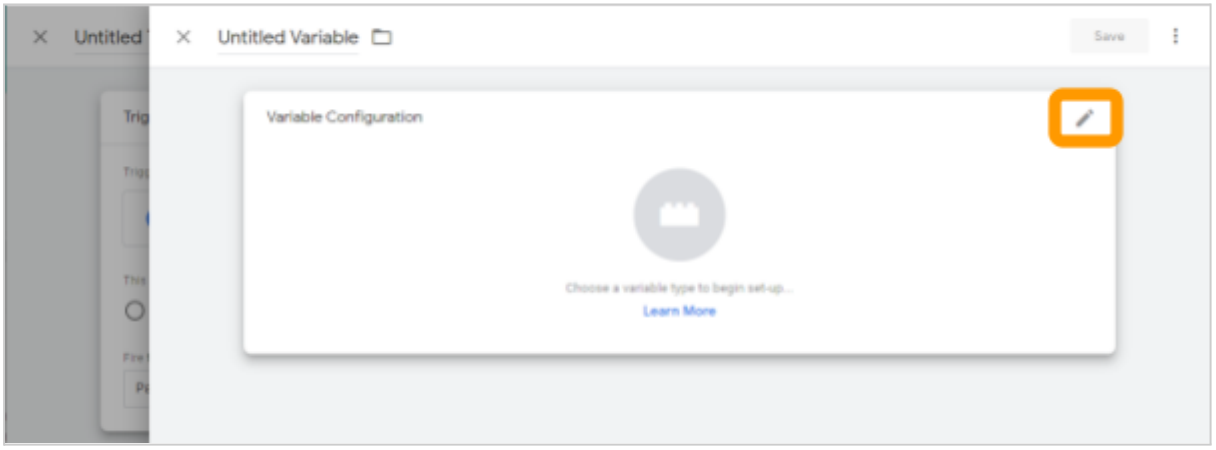
3 - Select “Page view” as the “Trigger type”:



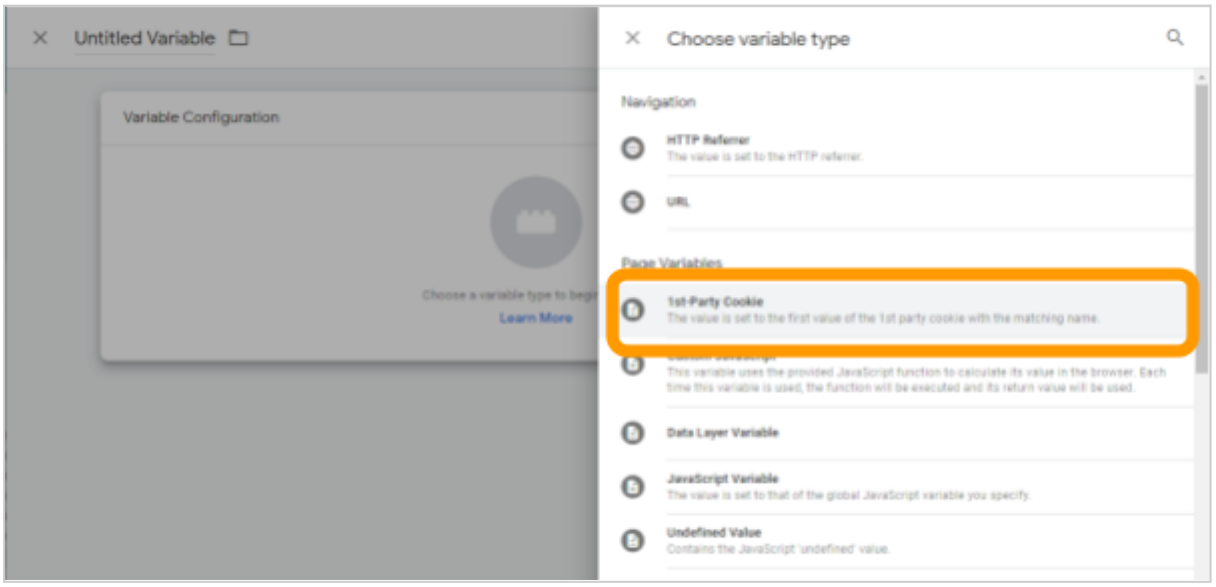
4 - Select "This trigger fires on... Some Page Views" and then ”New variable” from the first drop down of events:



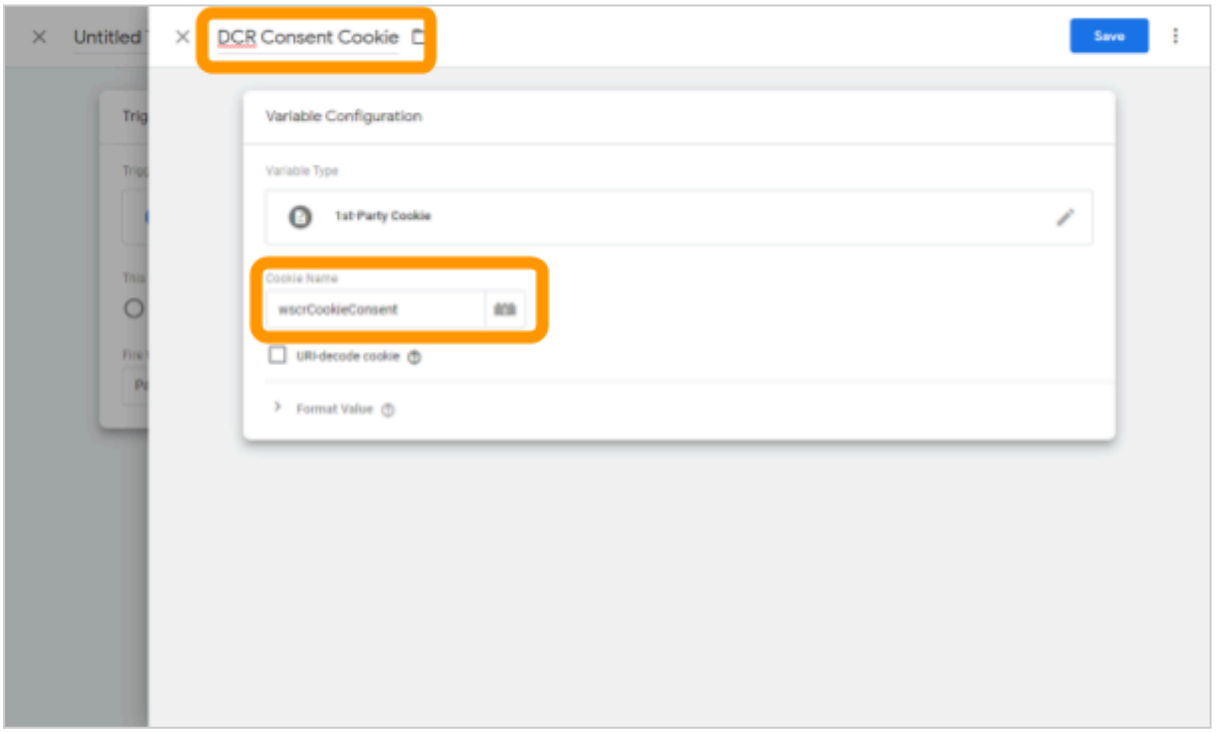
5 - On the new page click on the pencil icon to edit the Variable Configuration:



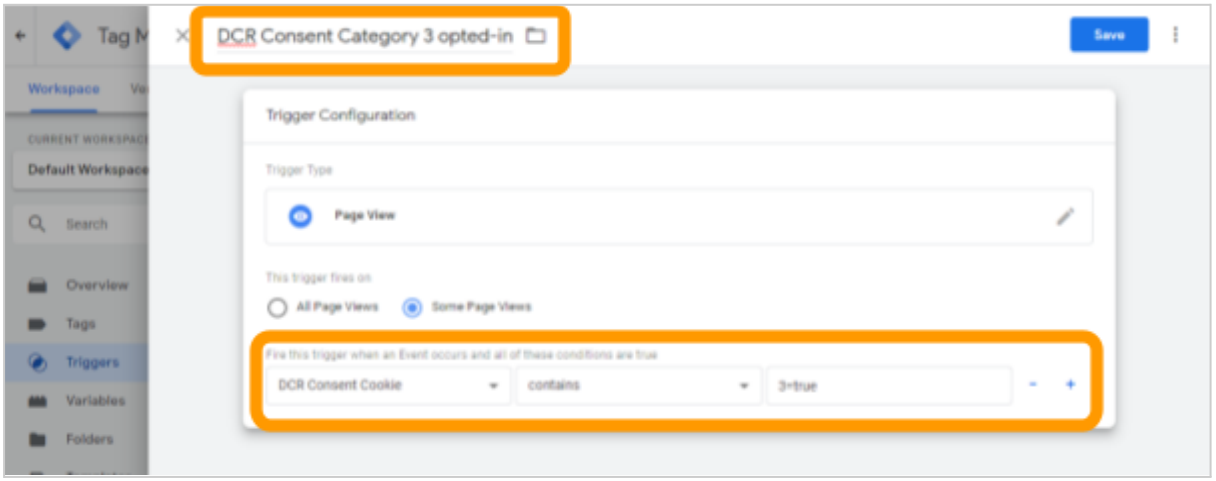
6 - Select “1st Party Cookie” variable type:



7 - Enter the “Cookie Name” as “wscrCookieConsent” and then name the variable, we suggest “DCR Consent Cookie”:



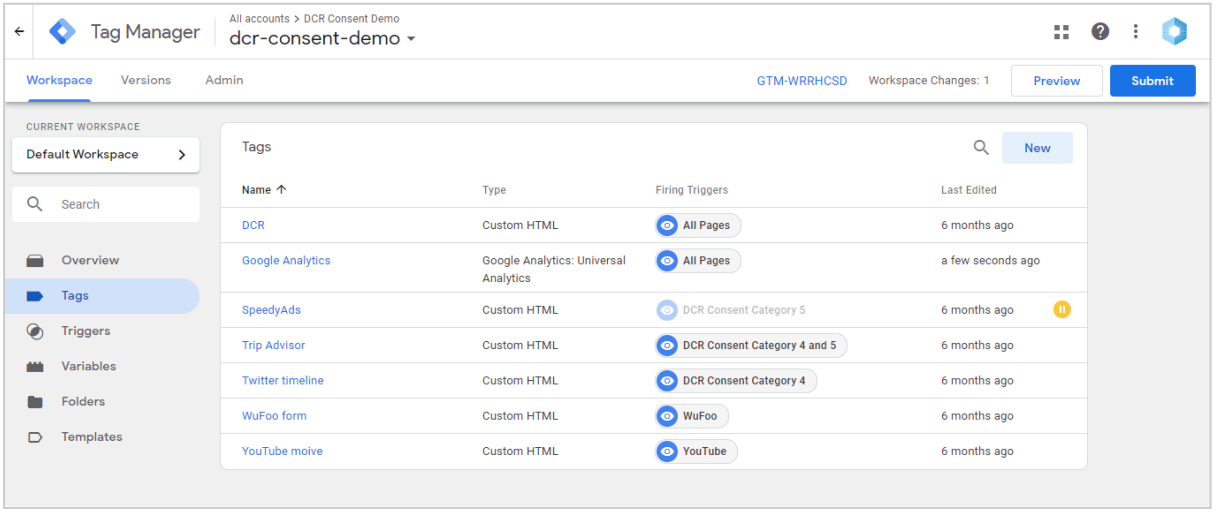
8 - After saving you will be taken back to the “Trigger Configuration” page. Leave the event as “DCR Consent Cookie” (that we just created), leave the check as “contains” and then enter the consent preference you wish to use. For example here we are checking for category 3 to be opted in and therefore entered “3=true” and named the trigger accordingly.



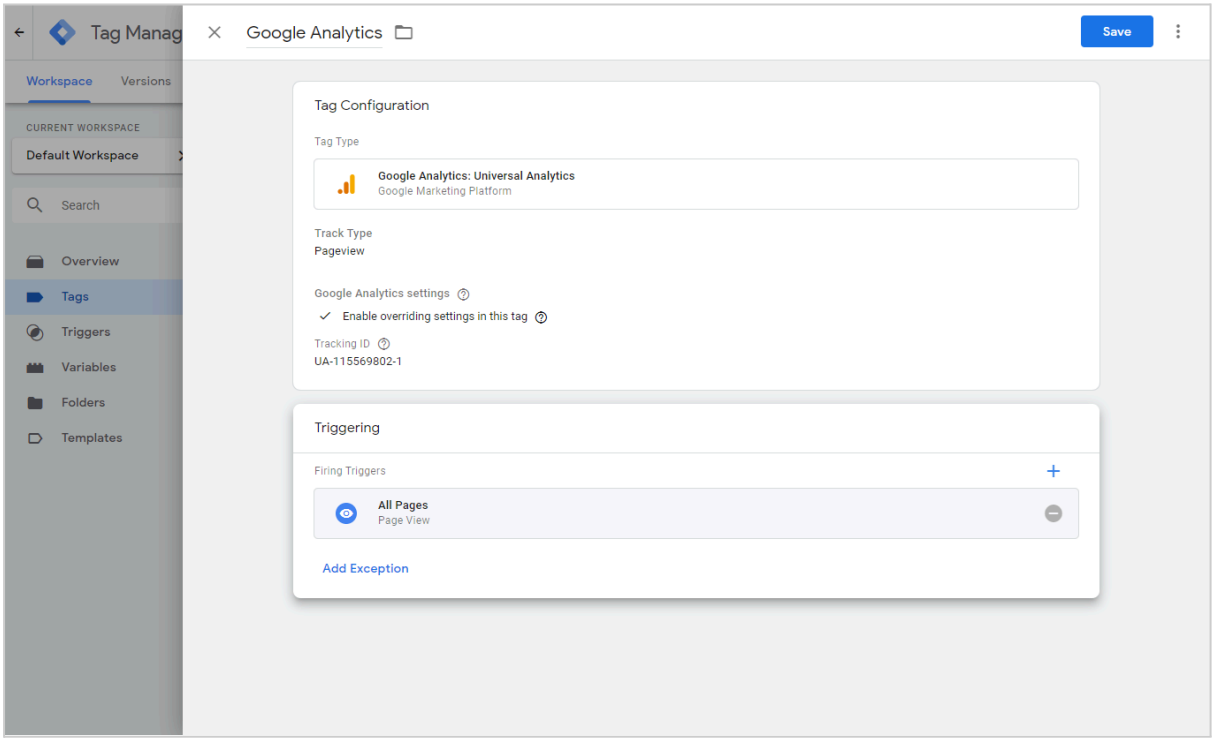
9 - The values to check depend on the category you wish to enforce:

Category Designation	Title	State	Value to check
Category 1	Necessary	Opted in	`1=true`
		Opted out	`N/A`
Category 2	Site experience	Opted in	`2=true`
		Opted out	`2=false`
Category 3	Performance & operation	Opted in	`3=true`
		Opted out	`3=false`
Category 4	Marketing, anonymous cross site tracking	Opted in	`4=true`
		Opted out	`4=false`
Category 5	Marketing, targeted advertising	Opted in	`5=true`
		Opted out	`5=false`

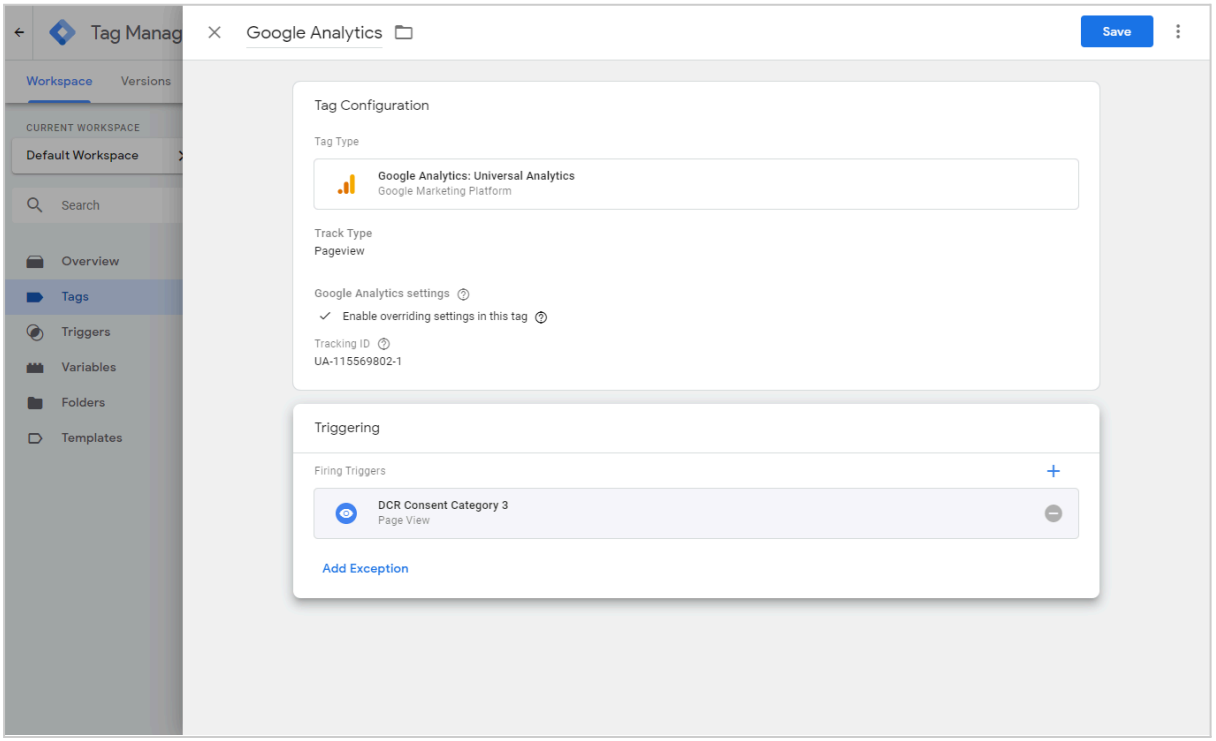
10 - Apply the trigger to your tags. Click on “Tags”:



11 - Select the tag you wish to restrict by clicking on the name:



Typically, as shown in the previous image, the trigger for a GTM tag is “All Pages” or similar. Click the Add sign and select the Cookie Category Trigger you created earlier and then remove “All Pages”:



12 - Save

Trigger tags based on Consent Mode V2

Google's Consent Mode enables website owners to manage and respect consent preferences related to advertising and analytics. Websites can adjust the behaviour of tags based on the consent status of their users as selected via the DCR banner.

Consent Mode is now required to gather some of the core Google insights related to advertising.

Configuration requires:

1. Setting “defaults”
2. Setting the trigger
3. Configuring Tags for Consent Mode

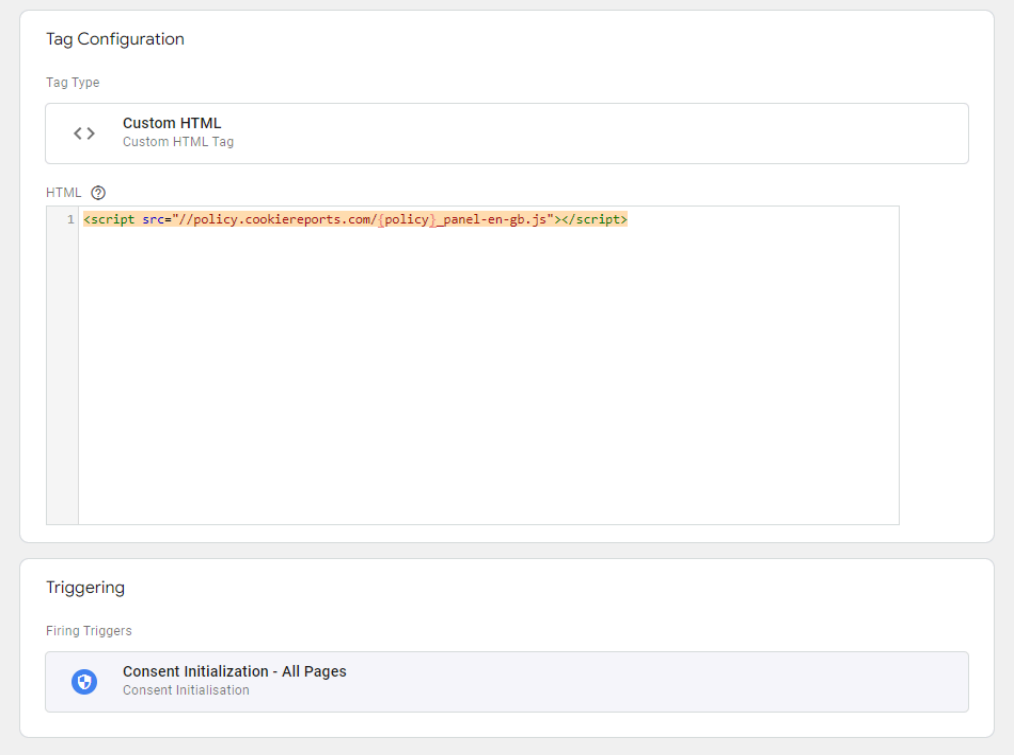
Setting “defaults”

When the DCR panel code runs it updates the consent settings for the page to match the current user preferences or the default preferences. These may differ by your configuration or by the location of the visitor.

We must update the page as soon as possible of the defaults / current selection.

Option 1 - GTM Consent Initialization

If you are using GTM to deploy the DCR cookie panel code we recommend instead of the “All pages” trigger that you select “**Consent Initialization - All Pages**” (which is designed to help ensure that consent settings are honoured before any other triggers fire).



Option 2 - HTML Defaults

If the DCR panel code is added to the page in a different way, insert the following code **before** any other scripts (including the GTM script block and the DCR panel code):

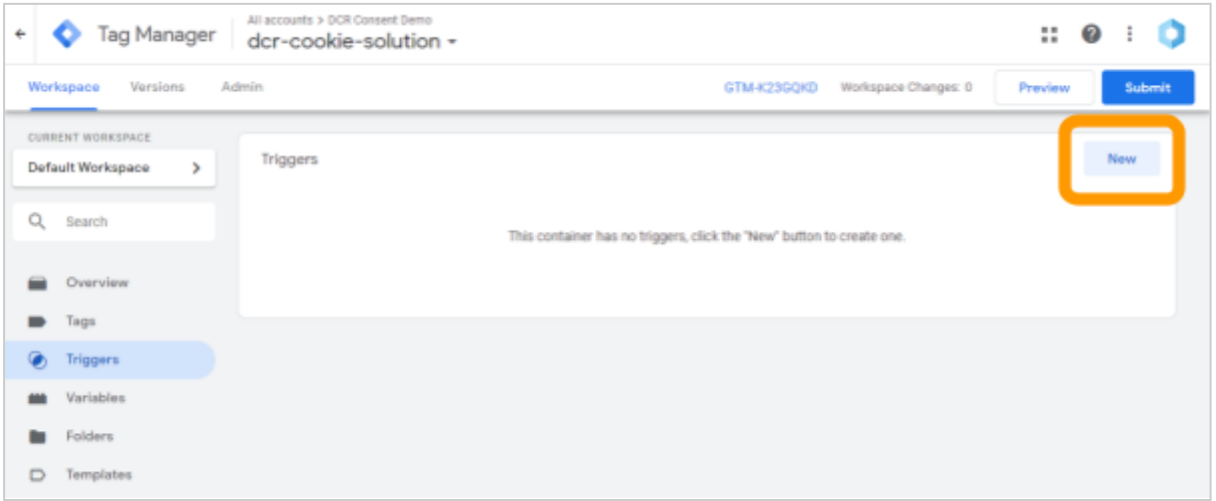
```
<script>
// Define dataLayer and the gtag function.
window.dataLayer = window.dataLayer || [];
function gtag() { dataLayer.push(arguments); }

/* Add defaults to gtag to deny by default */
gtag('consent', 'default', {
    'ad_personalization': 'denied',
    'ad_storage': 'denied',
    'ad_user_data': 'denied',
    'analytics_storage': 'denied',
    'functionality_storage': 'denied',
    'personalization_storage': 'denied',
    'security_storage': 'denied',
    'wait_for_update': 500  // milliseconds
});
/* Further restrict use of advertising cookies */
gtag('set', 'ads_data_redaction', true);
</script>
```

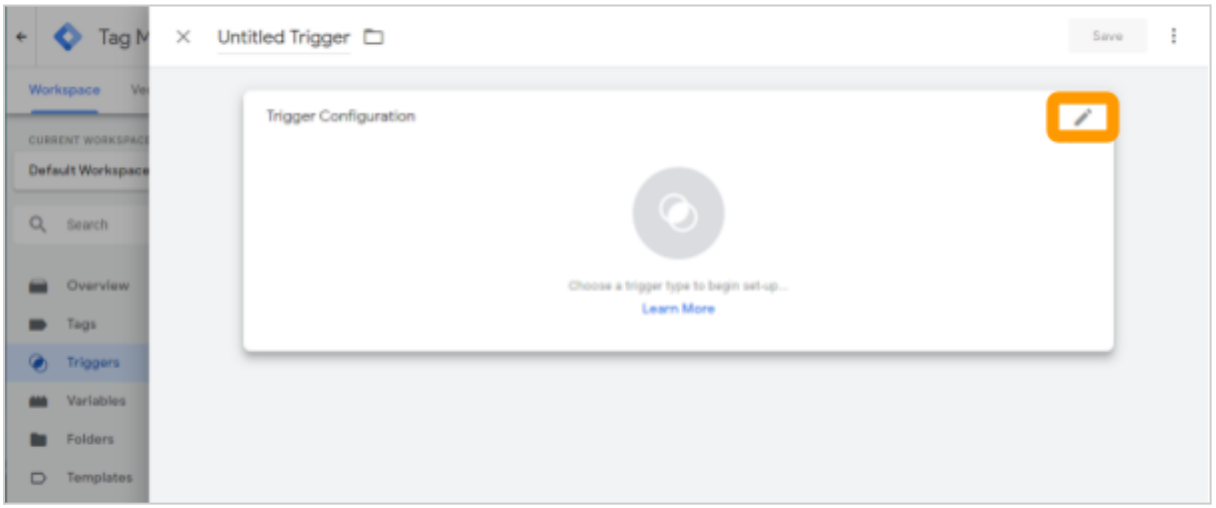
Setting the trigger

Most scripts that set cookies still need to be restricted by a trigger as seen in Trigger tags based on a DataLayer Variable and Trigger tags based on the value of our consent cookie in order to prevent the script from communicating until consent has been given.

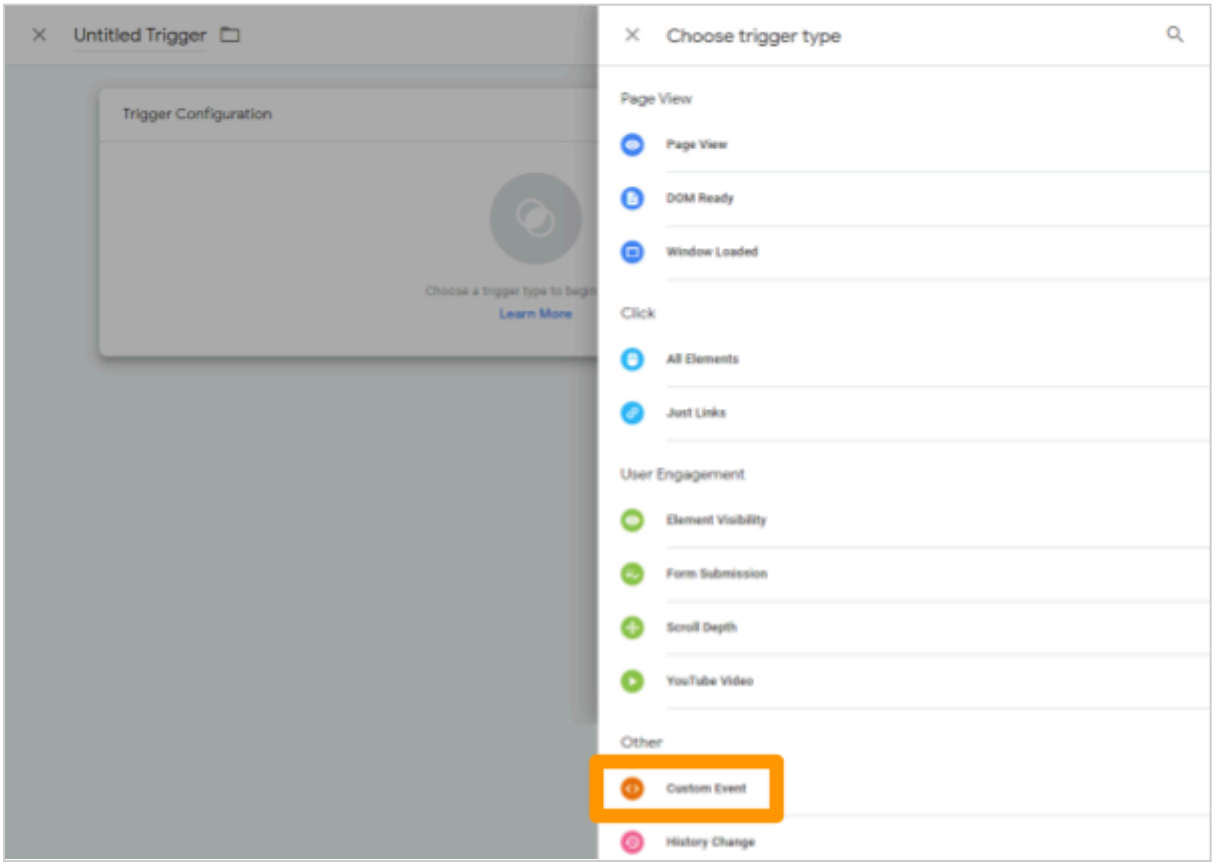
1 - Click “New” on the “Triggers” page



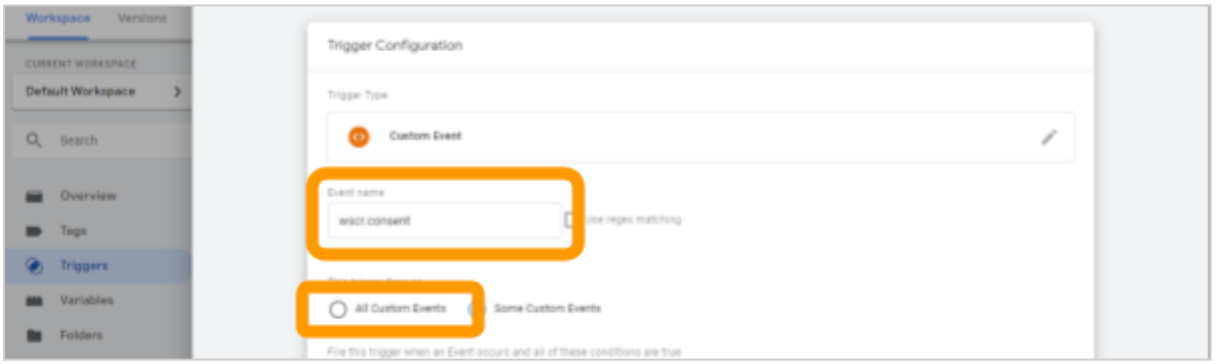
2 - On the new page click on the pencil icon to edit the Trigger Configuration:



3 - Select “Custom Event” as the “Trigger type”:



4. Enter “wscr.consent” as “Event name”, and select "This trigger fires on... All Custom Events"



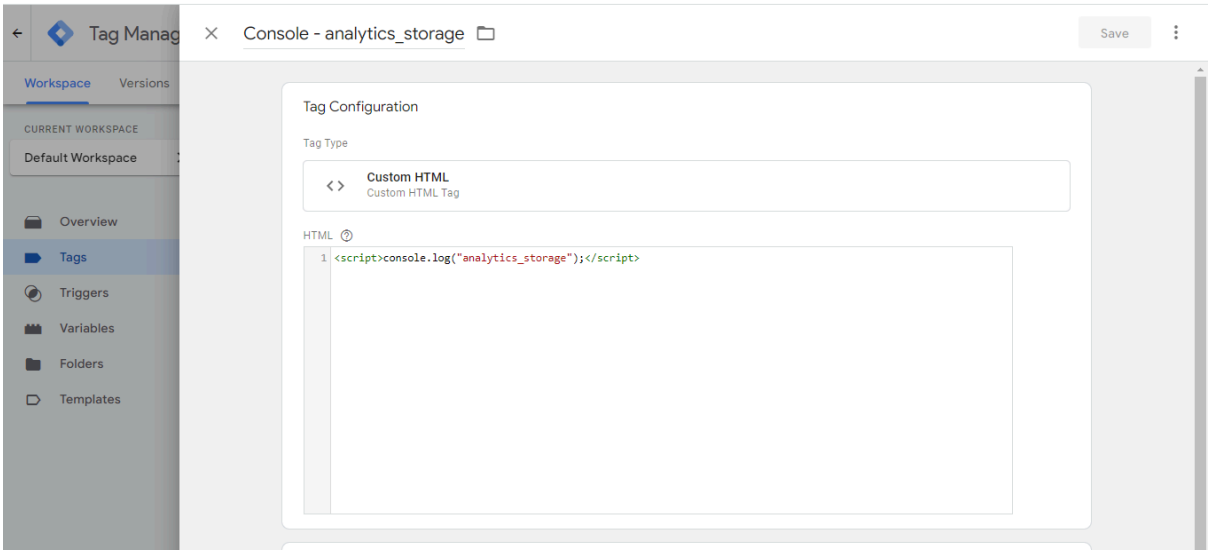
5. Name the Trigger (for example “DCR Event - consent”) and click Save

The consent type(s) you need to select depend on the category/categories you wish to enforce:

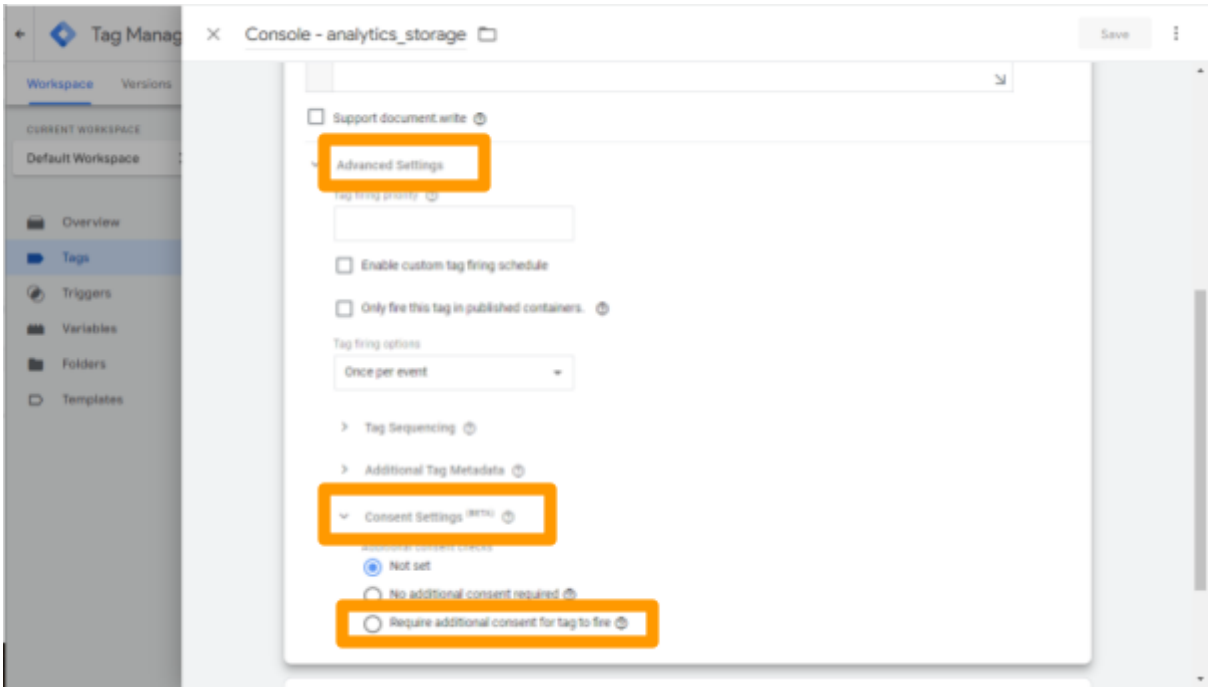
Configuring Tags for Consent Mode:

1 - Click on “Tags”

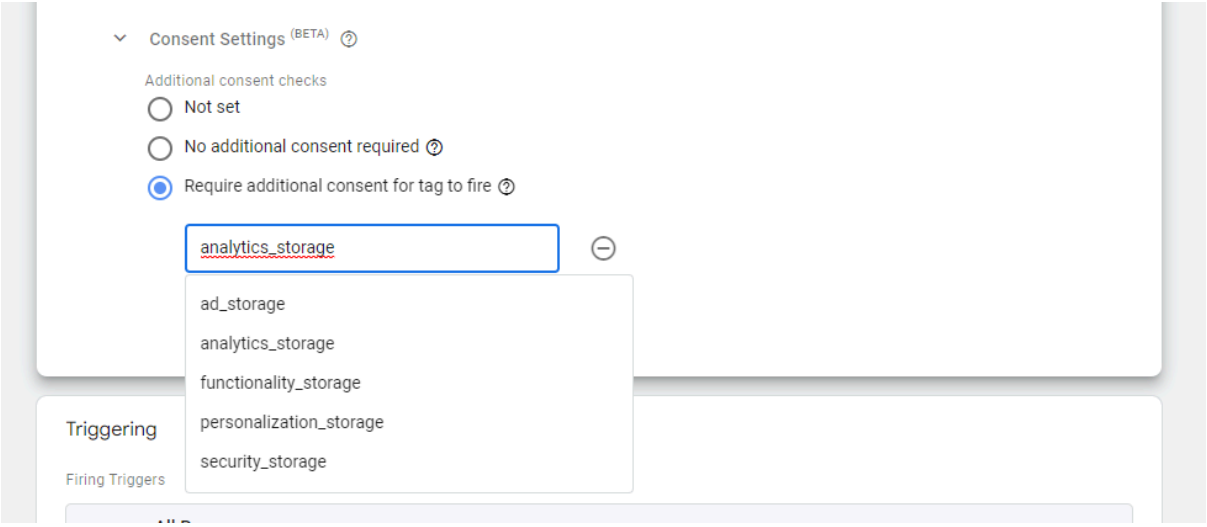
2 - Select the tag you wish to restrict by clicking on the name:



3 - Edit the Tag Configuration and toggle open Advanced Settings, then Consent Settings and enable “Require additional consent for tag to fire”:



4 - Select one or more consent types as appropriate to match the categorisation of your resource:



Category Designation	Title	Consent Mode
Category 1	Necessary	security_storage
Category 2	Site experience	functionality_storage OR personalization_storage
Category 3	Performance & operation	analytics_storage
Category 4	Marketing, anonymous cross site tracking	ad_storage
	Although a selectable mode, when Category 4 is selected the following consent modes will also be granted: ad_personalization ad_user_data	

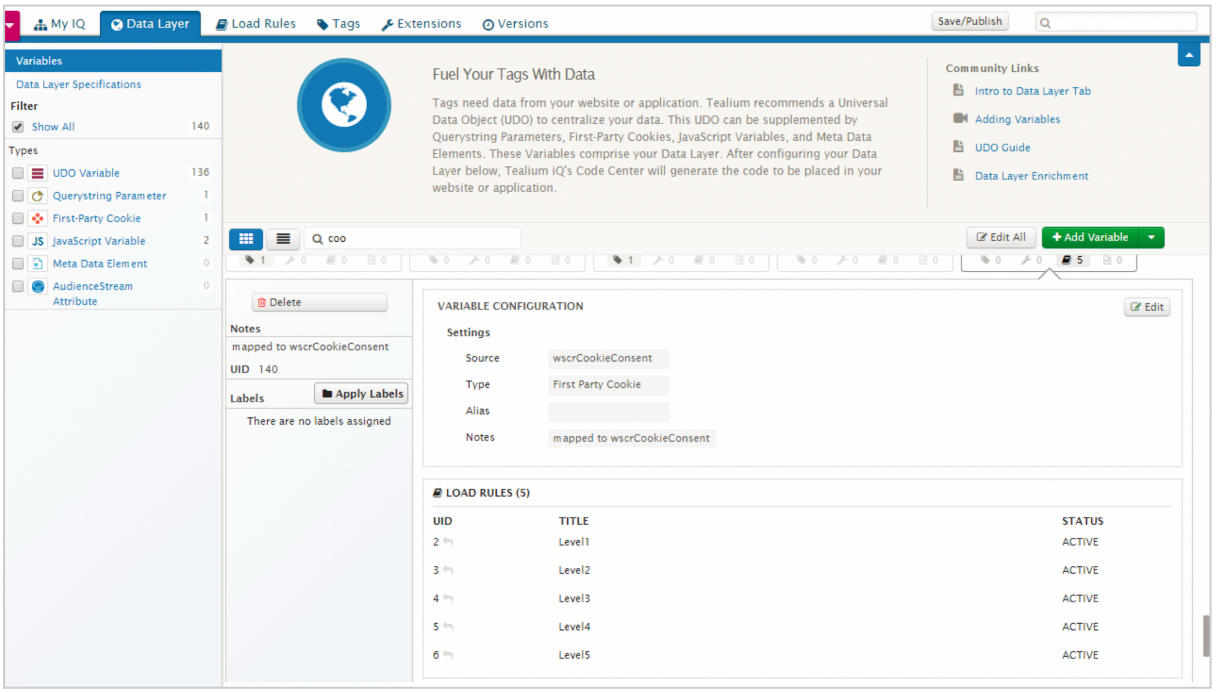
5 - Edit the Triggering section, removing All Pages (if listed), and select the Trigger you created earlier ("DCR Event - consent")

6 - Save

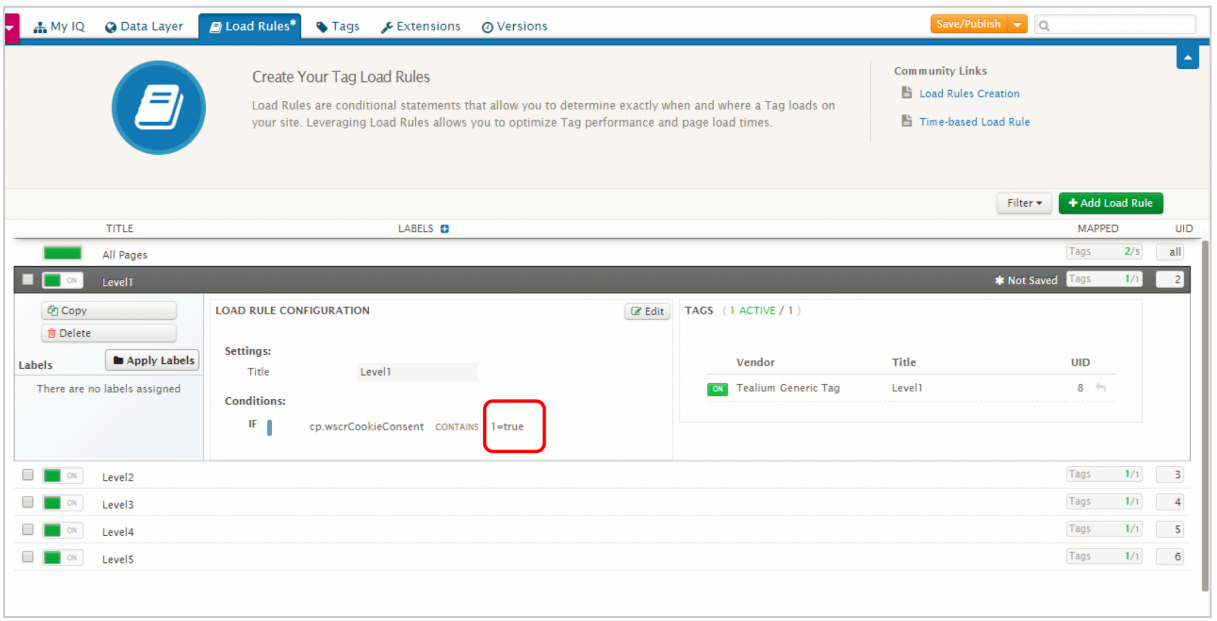
Tealium Tag Manager

The following guide creates a variable and load rule to which can be applied as required.

1 - Create a Cookie Data Layer Variable mapped to the Digital Control Room cookie stored:



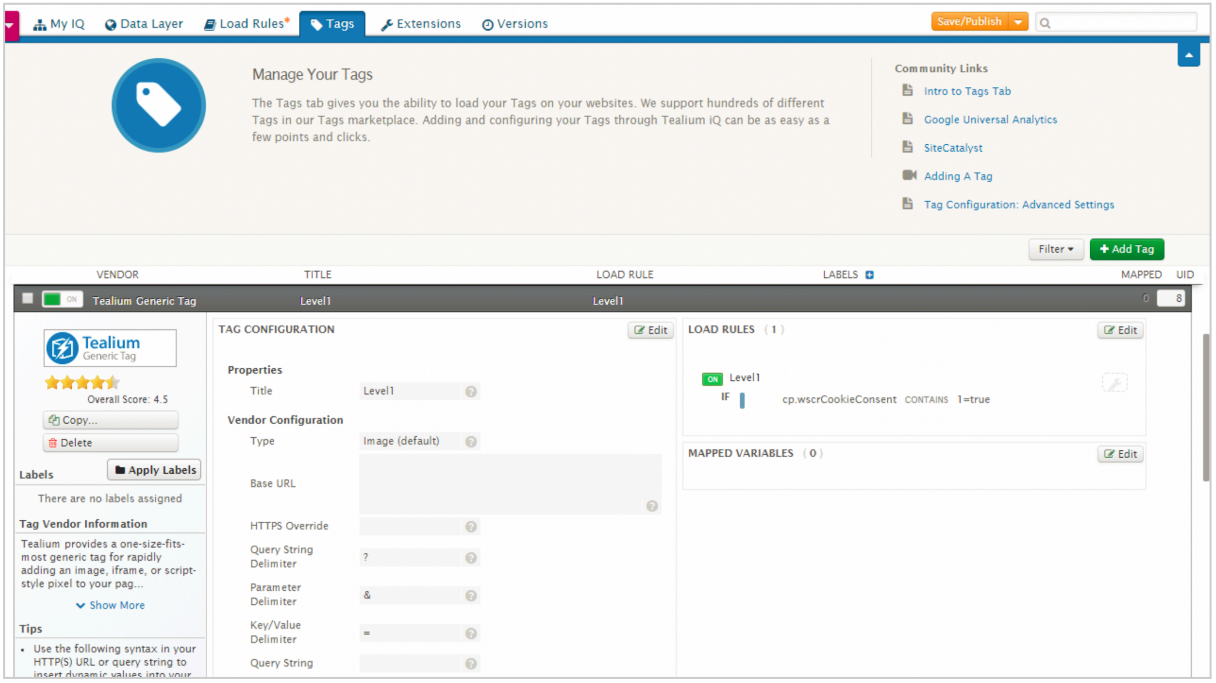
2 - Define Load Rules to only allow tags to be used when the Cookie Consent is defined as true.



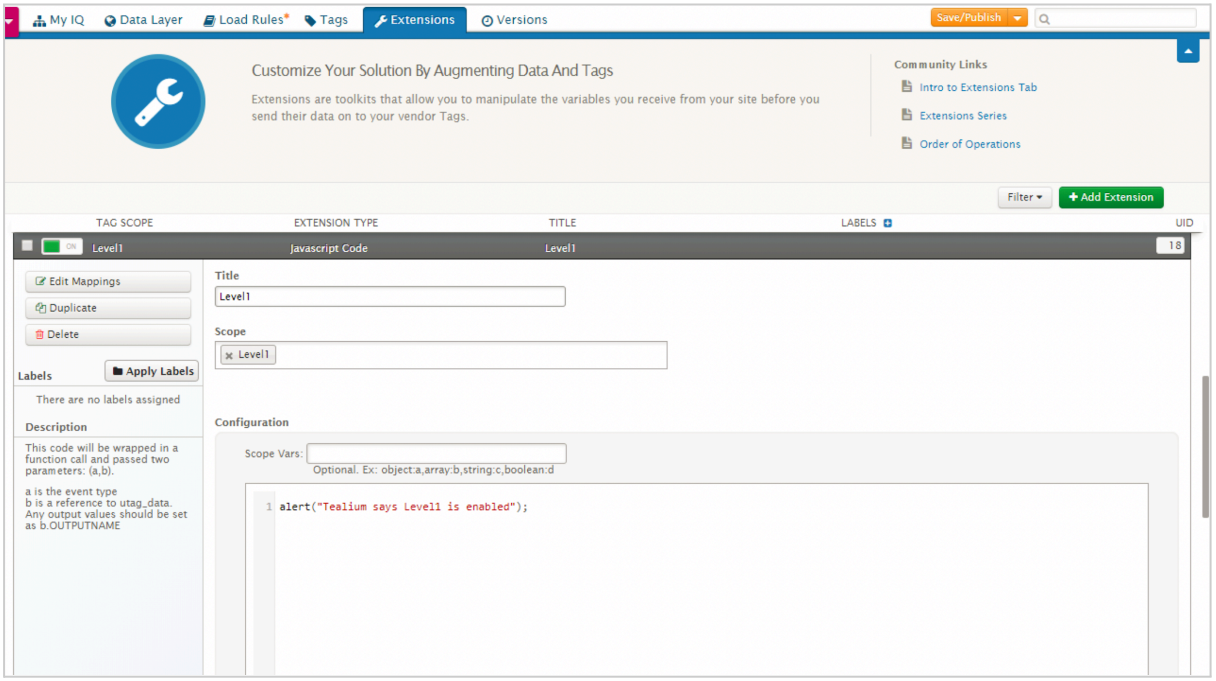
The condition to check depends on the category you wish to enforce:

Category Designation	Title	State	Value to check
Category 1	Necessary	Opted in	`1=true`
		Opted out	`N/A`
Category 2	Site experience	Opted in	`2=true`
		Opted out	`2=false`
Category 3	Performance & operation	Opted in	`3=true`
		Opted out	`3=false`
Category 4	Marketing, anonymous cross site tracking	Opted in	`4=true`
		Opted out	`4=false`
Category 5	Marketing, targeted advertising	Opted in	`5=true`
		Opted out	`5=false`

3 - Define the Tags to load based on the Load Rules and the types of cookies that are used by the tags
Example: use a generic tag and an Extension to indicate if the tag was loaded based on the Load Rules



4 -FOR TESTING ONLY - Extension that indicates whether the Tag was loaded. This is associated with the “Level 1” Tag.



Server Side Consent

HTTP cookies set by the webserver must also honour your site visitor's consent preferences.

The technique of checking consent server-side can be useful within a template to prevent the <script> tags from being inserted in the template in the first place - as a replacement or complement to the template changes described above.

If the site visitor has set a preference it is stored within a cookie named "`wscrCookieConsent`" on the local domain. The value of the cookie is structured as a URL query of key-value pairs:

```
1=true&2=false&3=true&4=false&5=true&visitor=bcf476ec-43e0-11e8-842f-0ed5f89f718b
```

This can be parsed safely in all programming languages. For example the following code samples indicate that the site visitor has:

- opted in to levels 1, 3, 5
- opted out of levels 2 and 4
- logged their consent with DCR under the reference: `'bcf476ec-43e0-11e8-842f-0ed5f89f718b'`
- Note that the cookie may contain other key value pairs in the future

Python

```
try:
    from urllib.parse import parse_qsl  # python 3
except ImportError:
    from urlparse import parse_qsl  # python 2

# replace this with code to extract the actual cookie value
# from the request object
cookie_value =
'1=true&2=false&3=true&4=false&5=true&visitor=bcf476ec-43e0-11e8-842f-0ed5f89f718b'

preferences = parse_qsl(cookie_value)
print(preferences)
```

```
[('1', 'true'), ('2', 'false'), ('3', 'true'), ('4', 'false'), ('5', 'true'), ('visitor', 'bcf476ec-43e0-11e8-842f-0ed5f89f718b')]
```

Go

```
package main

import "fmt"
import "net/url"

func main() {
	// replace this with code to extract the actual cookie value from the request object
	m, err := url.ParseQuery(`1=true&2=false&3=true&4=false&5=true&visitor=bcf476ec-43e0-11e8-842f-0ed5f89f718b`)
	if err != nil {
		// handle err
	}
	fmt.Println(m)
}
```

```
map[visitor:[bcf476ec-43e0-11e8-842f-0ed5f89f718b] 1:[true] 2:[false] 3:[true] 4:[false] 5:[true]]
```

PHP

```
// replace this with code to extract the actual cookie value
// from the request object
$cookie_value = '1=true&2=false&3=true&4=false&5=true&visitor=bcf476ec-43e0-11e8-842f-0ed5f89f718b';

parse_str($cookie_value, $get_array);
print_r($get_array);
```

```
Array ( [1] => true [2] => false [3] => true [4] => false [5] => true [visitor] => bcf476ec-43e0-11e8-842f-0ed5f89f718b )
```

HTML Templates

Modify all <script> tags to disable them from running until the site visitor has opted-in:

1. Locate all <script> tags in your templates, typically look something like this:

```
<script async
src="https://www.googletagmanager.com/gtag/js?id=UA-0000000-1"></scrip
t>
<script>
window.dataLayer = window.dataLayer || [];
function gtag(){dataLayer.push(arguments);}
gtag('js', new Date());
gtag('config', 'UA-0000000-1');
</script>
```

2. Determine the category of cookies set by these scripts and select one or more from the Cookie categories table (page 2).

   Example: for Google Analytics, cookies are set under category 3 (Performance & operation).

3. Make the following changes to the script tags to prevent the scripts running until category 3 is opted-in:

```
<script async type="text/plain" data-consent-category="3"
src="https://www.googletagmanager.com/gtag/js?id=UA-0000000-1"></scrip
t>
<script type="text/plain" data-consent-category="3">
window.dataLayer = window.dataLayer || [];
function gtag(){dataLayer.push(arguments);}
gtag('js', new Date());
gtag('config', 'UA-0000000-1');
</script>
```

   If, in a different example, you wanted to use multiple categories then enter each one separated by a comma:

```
data-consent-category="3,4,5"
```

   This example indicates a script that will only run if the site visitor has opted in to categories 3, 4 **and** 5

   For scripts setting only necessary cookies you can leave the script tag as is, or set `data-consent-category="1"`

If other tags set cookies, for example iframes, images, etc then DCR supports a simple generic mechanism to enable compliance on all tags.

Simply place the attribute you wish to disable (eg src) under the prefix "`data-consent-`", for example:

Change:

```
<img src="http://example.com/sets_cookies.png" alt="Image that sets cookies">
```

To:

```
<img data-consent-src="http://example.com/sets_cookies.png" alt="Image that sets cookies" data-consent-category="4,5">
```

If the site visitor has opted in to categories 4 and 5 the consent solution will dynamically set the src to "http://example.com/sets_cookies.png", otherwise it will hide the image tag.

This feature works for all attributes.

Force opting in to categories via a link

You may wish to consider assisting the site visitor with opting in to one or more categories in order to enable a feature they wish to use. For example a third party support chat application or a hosted video. The text of the link must make it clear to the user what they are doing.

Example:

```
<a href="#" class="CookieReportsLink" data-consent-enable="3,4">Accept our marketing cookies and enable this service</a>
```

The class is determined by your existing footer link setup. When "data-consent-enable" is set the default action of the link will not run (for example the banner will not be shown). The value of "data-consent-enable" can be a single category ("2") or an array as shown above.

Events

The DCR panel code triggers events which page authors can listen for, interpret and then make dynamic changes to the page.

This can be used instead of deactivating script tags (as described above) and instead injecting them to the page once the site visitor has opted-in.

Event:	`wscr.init`
Arguments	`event.detail` is an object with the following keys: ● `event.detail.consent` - an object of category numbers and a boolean indicating current consent, e.g.: `{1: true, 2: true, 3: true, 4: false, 5: false}` ● `event.detail.reason` - a string, value will be "init"
Trigger:	DCR panel code loaded, before any changes are made to the DOM. This is useful if you wish to modify the page before we make any changes.

Event:	`wscr.consent`

Arguments	`event.detail` is an object with the following keys:
	● `event.detail.consent` - an object of categories and a boolean representing current opt-in consent, e.g. :
	`{1: true, 2: true, 3: true, 4: false, 5: false}`
	● `event.detail.consent_string` - a serialized string representing current opt-in consent, for example:
	`1=true&2=true&3=true&4=false&5=false`
	● `event.detail.reason` - a string indicating why the event was processed, the value is one of:
	○ `init`
	Page loaded, set up initial state based on site visitor's existing preference if set.
	○ `banner`
	Source of event is from the DCR banner
	○ `accordion`
	Source of event is from the DCR accordion
	○ `page`
	Source of event is from the page
	● `event.detail.trigger` - a string indicating the user action taken, the value is one of:
	○ `ok`
	The OK button was clicked
	○ `consent-switch`
	A consent switch was clicked
	○ `link`
	The on page link
	○ `auto`
	Consent was automatically given. Perhaps due to a regional preference
	○ `shared`

	Consent was saved or loaded from a shared grouping ○ `sharedStatus` Indicate state of shared consent, one of: “used” (shared consent was set and represented here), “empty” (shared consent has not been set and default is represented here), “expired” (shared consent was set but has expired, and the default is represented here) When trigger = “consent-switch” additional keys are provided: ● `event.detail.display` - a string indicating the display type the user interacted with, the value is one of: ○ `inject` The inject accordion ○ `popup` The popup accordion ● `event.detail.level` - an int indicating the category number changed. This can be combined with event.detail.consent to take different action on consent given vs withdrawn When trigger = “shared” additional keys are provided: ● `event.detail.shared` - a string indicating the action taken, the value is one of: ○ `load` The preference was loaded remote and used on the site ○ `save` The preference was saved remotely
Trigger:	Triggered after changes are made to the DOM that enable or disable features based on a new opt-in or opt-out selection. Also triggered on page load (after `wscr.init`) to enable any features previously opted in.

Event:	`wscr.consent.log`
Arguments	`event.detail` is an object with the same keys as wscr.consent plus: ● `event.detail.log` - an object containing: ○ `status` A string reporting the status of the log request, will be one of "success", "error", "timeout", "abort", or "parsererror" ○ `data` (if successful) An object containing: ■ `visitor`: The Site visitor reference UUID
Trigger:	Triggered after a response received from the remote consent log. Note that consent logging is an optional feature.

Example listeners

These examples are provided as a starting point.

Log the “detail” object:

```
document.addEventListener("wscr.init", function(event) {
    console.log(event.detail);
});

document.addEventListener("wscr.consent", function(event) {
    console.log(event.detail);
});
```

Show / Hide the container of a third party plugin:

```
document.addEventListener("wscr.consent", function(event) {
    var div = document.getElementById("social-fallback");
    if (event.detail.consent[4]) {
        div.style.display = "block";
    } else {
        div.style.display = "none";
    }
});
```

Insert third party script tag when category 5 opted in:

```
document.addEventListener("wscr.consent", function(event) {
    if (event.detail.consent[5]) {
        var script = document.createElement('script');
        script.src = "//www.example.com/script.js";
        var head = document.getElementsByTagName("head");
        head[0].appendChild(script);
    }
});
```

Reload page after category 5 opted in via clicking “ok” on banner or accordion popup:

```
document.addEventListener("wscr.consent", function(event) {
    if (event.detail.trigger == "ok" &&
event.detail.consent[5]) {
        window.location.reload(true);
    }
});
```

If logging is enabled our log action will be cancelled if the page is navigated away, instead you can listen for the logging event:

```
document.addEventListener("wscr.consent.log", function(event)
{
    if (event.detail.trigger == "ok" &&
event.detail.consent[5]) {
        window.location.reload(true);
    }
});
```

Reload page after category 3, 4 or 5 opted in via inject accordion:

```
document.addEventListener("wscr.consent", function(event) {
    if (event.detail.reason == "accordion" &&
event.detail.display == "inject" && event.detail.trigger ==
"consent-switch" && event.detail.category >= 3) {
        window.location.reload(true);
    }
});
```

Cross-domain consent

Clients may use an optional feature to share consent preference across a group of their domains. Also known as cross-domain consent, this enables a visitor (using a compatible browser) to set their consent preference once and have this automatically applied whenever they visit another site within the group of domains.

We recommend the use of the Events consent management technique. Specifically listening for the Event with the joint properties of trigger=”shared” and reason=”page”. An example JavaScript listener is provided below. The same concept can be used within Google Tag Manager’s DataLayer, Tealium and other tag managers.

We recommend against the use of the “init” Event, as this is fired before the cross-domain preference lookup is made. Also, depending on configuration, the initial value of the “wscrCookieConsent” cookie may also have been set before the cross-domain preference lookup is made.

Cross-domain (shared) consent listener:

```
document.addEventListener("wscr.consent", function(event) {
    if (event.detail.trigger == "shared" &&
event.detail.reason == "page") {
        console.log(event.detail);
    }
});
```

The full payloads and options are described in more detail in the Events section. The contents of event.detail (as logged in the above listener) is as follows:

When no cross-domain (shared) consent is set:

```
{
    "trigger": "shared",
    "shared": "load",
    "sharedStatus": "empty",
    "reason": "page",
    "consent": {
        "1": true,
        "2": true,
        "3": true,
        "4": true,
        "5": true
    },
```

```
    "consent_string":
"1=true&2=true&3=true&4=true&5=true&version=20210101-001"
}
```


Note that “sharedStatus” is set to “empty” to identify the lack of a match. The value of the consent string matches the default for the policy, in this case with everything defaulted to opted-in.

When a cross-domain (shared) consent is present:


```
{
    "trigger": "shared",
    "shared": "load",
    "sharedStatus": "used",
    "reason": "page",
    "consent": {
        "1": true,
        "2": true,
        "3": true,
        "4": false,
        "5": false
    },
    "consent_string":
"1=true&2=true&3=false&4=false&5=false&version=20210101-001"
}
```


Note that “sharedStatus” is set to “used” to identify the use of the cross-domain consent. The value of the consent string is copied from the cross-domain preference, in this case categories 4 and 5 are opted out.

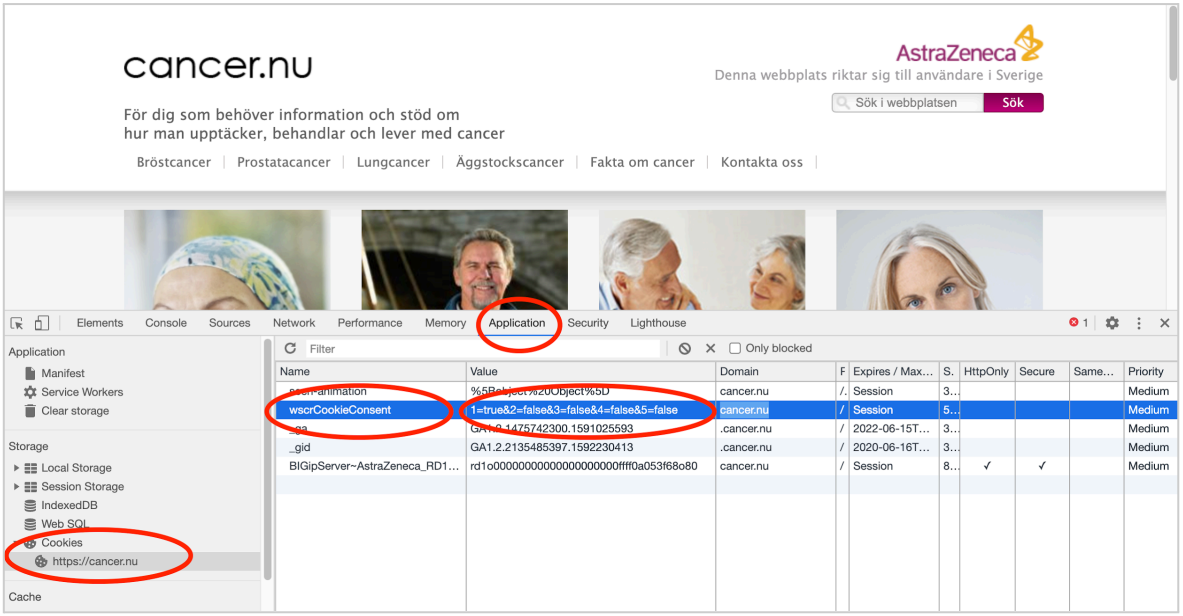
Testing the deployment

To enable Consent for development and testing purposes you have two options:

1. Manipulate the cookie to test different scenarios
2. Toggle consent switches on via a local storage configuration

1 - Manipulate the cookie to test different scenarios

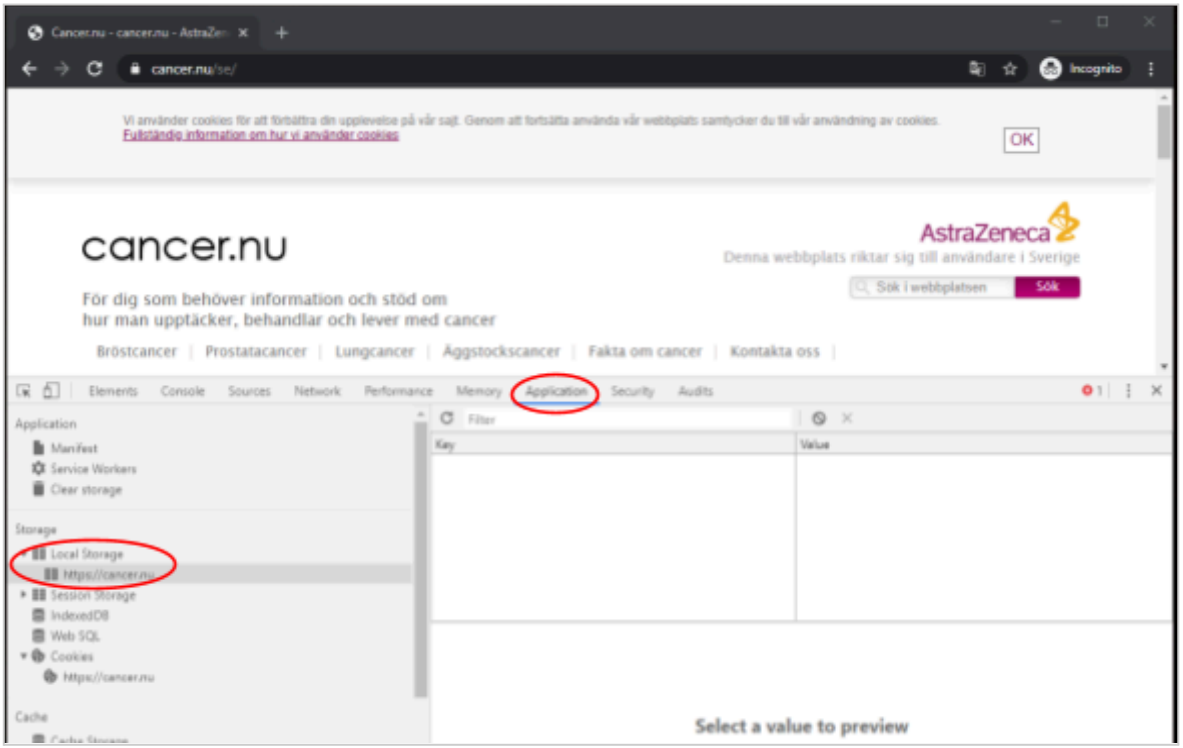
1. Navigate to the site in Chrome (either the front page or the cookie information page) and press F12 to display the Developer tools.
2. Go to “Application” > “Cookies” > The address of your site:
3. Select your site on the left and then double click the area under “Name” and add “wscrCookieConsent”
4. Then, next to the cookie you just created select the empty space under the column “Value” and add enter one of the following values depending on the scenario you wish to test:
    a. *Necessary only*: 1=true&2=false&3=false&4=false&5=false
    b. *Allow all cookies*: 1=true&2=true&3=true&4=true&5=true
    c. To test the enabling of a single category or a mixture of categories set the desired number(s) to “N=true” (remember to always set category 1=true)
    d. Delete the cookie to experience the site as a new visitor.



2 - Toggle consent switches on via a local storage configuration

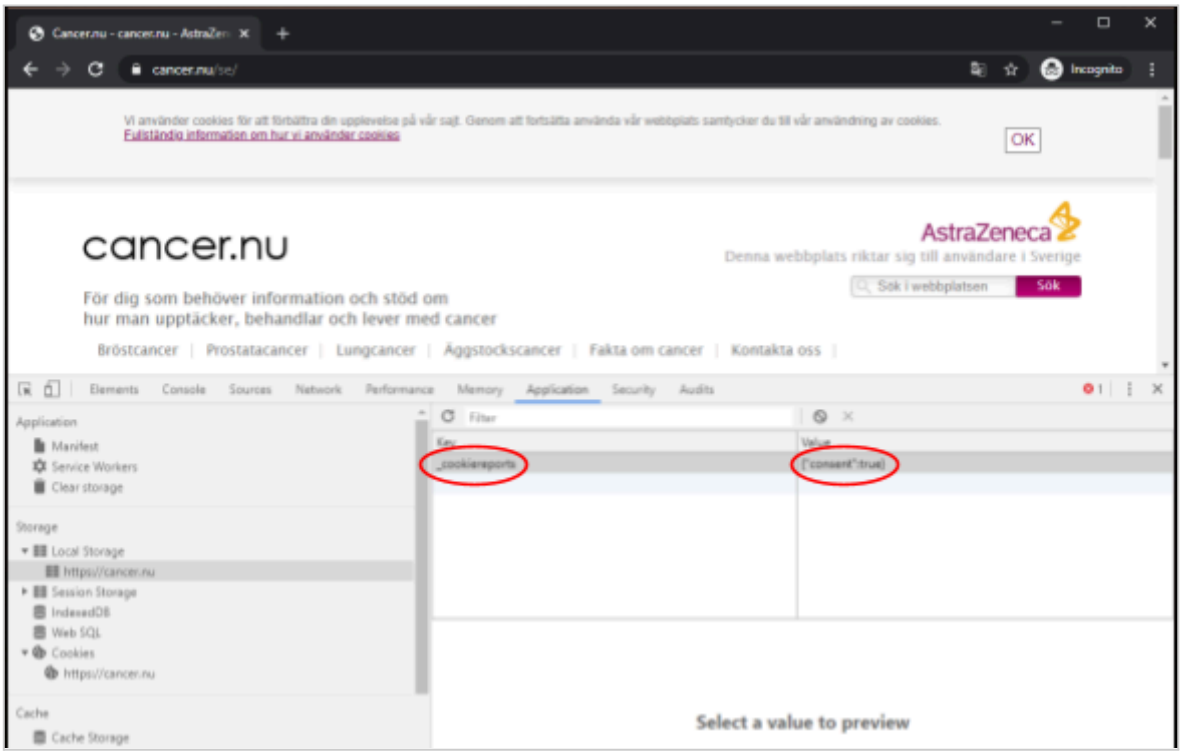
1. Navigate to the site in Chrome (either the front page or the cookie information page) and press F12 to display the Developer tools.

2. Go to “Application” > “Local Storage” > The address of your site:



3. Select your site on the left and then double click the area under “Key” and add: _cookiereports

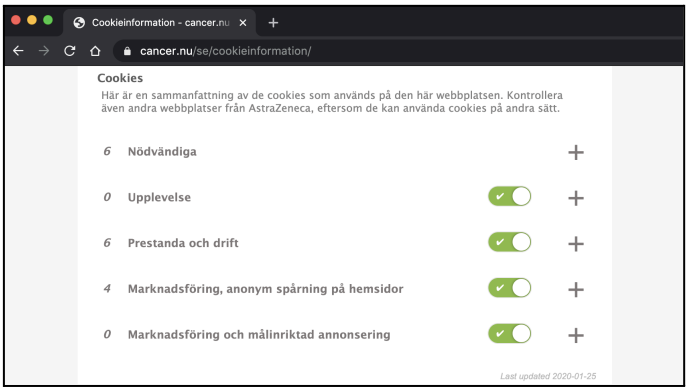
4. Under “Value” add: {"consent":true}



5. Close the console by clicking the “x”, to the right of the console tabs.

6. Refresh the web page

7. Click the link in the banner (or reload the cookie information page) and the existing accordion should now show the consent toggle controls, similar to:



To reset this override and no longer see the additional consent toggle controls, right click on the entry previously added and select “Delete”.

< End of Document >

Digital Control Room Ltd
www.digitalcontrolroom.com
info@digitalcontrolroom.com

Disclaimer

This document is offered as an overview and a starting point only – it should not be used as a single, sole authoritative guide. You should not consider this as legal guidance.

The service provided by Digital Control Room Limited is based on an audit of the available areas of a website at a point in time. Sections of the site that are not open to public access or are not being served (possibly be due to site errors or downtime) may not be covered by our reports.

Where matters of legal compliance are concerned you should always take independent advice from appropriately qualified individuals or firms.