ALLAN THRAEN | 3 years ago | PDF |

Optimizely (Episerver) Integrations Addon Development Website Improvements Tips and Tricks

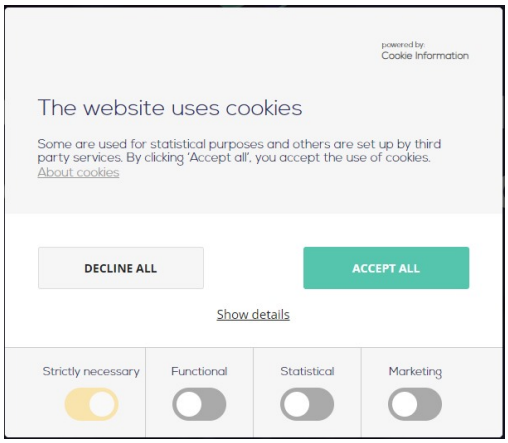
ADHERING TO CONSENT WITH COOKIE INFORMATION FOR EPISERVER

In the EU the past year has added even more rules and regulations to which cookies can be set, which data can be collected and which consents are needed for it. While it may not be tricky to add a basic consent box, adding one that adhere to all the proper legislation and then follow the consents given can be a bit more challenging. In this post I take a deep dive into how Cookie Information's solution together with their Connector for Episerver can make it easier - and faster to accomplish.

Cookies are not just a festive treat, or a useful web-technolgy, they can also be the cause for many a legal headache.

Nowadays in the EU, it's not enough to simply inform visitors that you use cookies - no, you have to get an explicit permission from them (based on a well informed decision) for each category of cookies used on your web site. In fact, in the interpretation in at least some countries you cannot even nudge them towards "Accept All". Furthermore they should be able to change their consents easily and at any time. This poses some extraordinary challenges to those responsible for web site compliance and website development.

Luckily we have vendors like Cookie Information (and others) that make it fairly easy to get these consents in a nice and standardized way (known as 'Consent management'). They even crawl your website to detect which cookies it's trying to set, so they can be listed in the proper categories and explained in order for the visitors to make an informed decision.



So far, so good. But, what I've noticed that a lot of web sites they just add the consent box and then tend to forget that there is a lot more to it than just pasting in a line of javascript. Adding the consent box and signing up with a consent vendor is only half a solution. The other half - one might say 'the tricky half' - is to actually **adhering to the consents** given by the visitors, collected by the consent box!

Why is this so important?

When you are a company with a website, you should obviously follow the rules. And over the last few years the data/privacy rules are being more and more enforced by government bodies in the EU countries - and the many companies have already gotten multi-million euro fines for violating the rules.

Where do cookies come from?

This is tricky because cookies are quite essential for many things - since a web visit technically breaks down to a lot of independent HTTP calls, cookies is often the chosen technology to ensure that there is something resembling a coherent dialogue going on. A fun mental exercise is trying to imagine going to a real life physical store and have a conversation with a sales guy, but the sales guy has a complete short time memory loss and can only answer your latest question without remembering ever having seen you before. But I digress.

Cookies are typically set in many different places - and how you can disable them depends on where, how and by what they are set.

These are some of the typical places I've noticed them:

- Google Tag Manager. Different tags embed various 3rd party code - like Google Analytics tracking, facebook pixel, and so on.
- Through javascript or iframes, referenced in razor Views. For example embedded youtube videos, ShareThis panels with share buttons, linkedin integrations and remarketing services. Could also be Marketing Automation tracking.
- In Controllers and other code-behind. These are often functional - for example to store things about the current visitor selections and preferences.
- In Episerver and Addons. This could be A/B testing cookies, Form tracking, personalization support

and so on. Or Episerver recommendations, Profiles and tracking and so on.
- In scripts referenced or injected by server-side code outside views. For example Episervers Analytics integration.
- In custom snippets inserted by Editors in HTML fields or custom blocks. Often a piece of clever "embed" code generated by a 3rd party service to be copied in.

*Keep in mind, that legally you cannot track in Google Analytics or show an embedded Youtube video **before** you have gotten positive consent from the visitor to those specific cookies.*

The solution

To make it easier for developers building websites on Episerver, we @ codeart have collaborated with the team at Cookie Information, and together we've built an Episerver connector for their service with the primary purpose to make it easier to follow the consents given by your visitors.

First of all, it comes with an easy way to map cookies to the different cookie categories, typically: necessary, functional, statistic, marketing and other. Once they are mapped, the connector includes a full toolbox of components you can use to control the cookies for each of the above mentioned scenarios. This includes a HTTP Module that 'washes' the response of cookies set in the backend without consent, extension methods and helpers to check in Razor or controller code which consent is given as well as visitor group criteria to even allow editors to handle lack of consent directly where they embed components that set cookies.

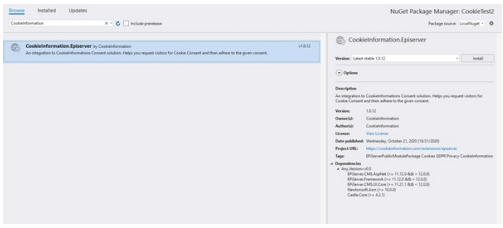
To make things easier, the connector also includes the ability to auto-register the script for the consent box itself (with the correct language setting), and even a short-cut menu item to take you to Cookie Informations dashboard.

Installing the Integration

Installing the integration is a straight forward nuget package install.

The package Cookieinformation.Episerver can both be found in the Episerver feed, as well as in the regular nuget.org feed. Once installed, there is a few configuration settings you'll have to adjust in your app settings - and naturally, you also need to make sure you have an account setup with Cookie Information (they do offer a free trial).

If you are working on a development environment you'll also want to make sure your development domain is registrered there as well (note, they do not support localhost, so configure a local domain in your hosts file).



Using the integration

By default the integration will inject the Consent box on all pages, adjusted to the content language, but if you prefer to insert it yourself that can easily be changed in configuration.

The configuration (AppSettings) is also where you map the cookies to categories.

And if you look through your Episerver edit mode, you'll find a new menu item to easily access the Cookie information dashboard.

In the dashboard you can configure your custom cookies as well, and manage the text and UI for your consent box, as well as get detailed insights into cookies, privacy risks and consents given.

Now, you are ready to start adhering to consents.

Adhering to consents

As mentioned above, cookies can be set in a lot of different ways, and here is some suggested strategies to handle them with the Cookie Information connector:

Google Tag Manager Tags

It's pretty straightforward to handle Tag Manager tags that embed code which can set cookies. Simply follow the steps described here to change the triggers, or check a custom variable for the correct consent.

Razor Views

There are several ways to deal with code embedded through a razor view that sets cookies. Cookie Information comes with a full SDK that's already preloaded, where you can basically add a "data-category-consent" attribute to a pixel-tag or an iframe to prevent it loading unless that consent is given, described here. But the connector also provides server-side helper methods you can wrap your razor code in, which can be really helpful - for example to ensure that provide a good alternative to those visitor that do not consent.

Here are a few examples:

```
@if (VisitorConsentService.Current.ConsentsToCategory(CookieInformationCategories.MARKETING))
 {
                                                // Script that will set a marketing cookie
                                                console.log('There is consent to Marketing cookies'
     <script type="text/javascript">
 }
```

The code above checks if the visitor consents to a given cookie category, but you can also check against a specific cookie:

```
    @if (VisitorConsentService.Current.ConsentsToCookie("MySpecialCookie"))
    {

                            //Code that sets a cookie called "MySpecialCookie" which is mapped to the
                             console.log('There is consent to MySpecialCookie');
        <script>
    }
```

In some cases it makes a lot of sense to wrap it in javascript like this, because then your visitors won't have to wait for a server-roundtrip after they give their consents to see the consequences.

```
    @using (Html.CreateConditionalCookieScript(CookieInformationCategories.STATISTIC, "MyInit()"))
    {
        <script>
            function MyInit() {
                //Code that sets statistical cookies
                console.log('There is now consent for Statistical cookies');
            }
        </script>
    }
```

Controllers and code-behind

The same helper methods can of course be used in your controllers - or anywhere else in your code, assuming you are setting cookies there, or perhaps injecting references to external code that might set 3rd party cookies.

```
public class StartPageController : PageControllerBase<StartPage>
    {
        private readonly IVisitorConsentService visitorConsentService;
        public StartPageController(IVisitorConsentService consentService)
                                                                        {
            this.visitorConsentService = consentService;
        }

        public ActionResult Index(StartPage currentPage)
                                                    {
            var model = PageViewModel.Create(currentPage);

            //Set cookie
            if (visitorConsentService.ConsentsToCategory(CookieInformationCategories.FUNCTIONAL))
            {
                Response.Cookies.Add(new System.Web.HttpCookie("MyServerSideCookie", "Cookie Value"));
            }


            return View(model);
        }
    }
```

Episerver and add-on cookies....and catch-all

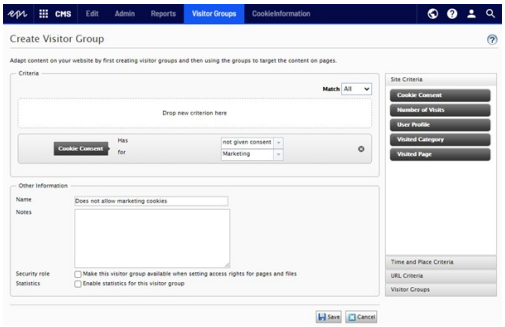
This is probably one of the main features of this connector. Episerver and a couple of popular add-ons, sets plenty of cookies for various purposes (most of them are already mapped out-of-the-box in the integration, btw). But since they are not set in a place where you can control them, they can be tricky to remove in the case of missing consent.

To solve this, the connector comes with an HTTP module that attaches itself to the end of the request/response flow. Just before the HTTP response is about to be returned to the browser, it will intercept them and quickly check if any cookies are being set that the visitor has not consented to. And if there is, it will remove them! It's that simple!

In fact, although it's off by default, you can in configuration enable that it should clean up old cookies set before the consents were changed last time!

Editor injected code that causes cookies

The last typical scenario is when you have editors that add interactive content from another website. Typically through an "html-block" (yes, sadly I often see them) - or similar. It could also be that the editor wants to insert a video, but wants to ensure a good alternative in case of a missing cookie consent. For these scenarios we included a Visitor Group Criterion in Episerver that makes it easy to define visitor groups based on their consent status to various categories, and then simply let editors use basic personalization - either for blocks, or inside an XHTML property - to define what can be shown - and which alternative should be shown to those with a different consent status. It's that simple, but of course might take a bit of training.



Proposed strategy

Ensuring proper consent is handled throughout a large sites with a lot of history can be a big task. I usually break it down into smaller bites:

1. Install connector, map cookies (you can see a list of all crawled cookies in the Cookie Information dashboard) and adjust consent box text/layout.
   This will give you the consent box on your site and the automatic removal of 1st party server-side cookies set in the response.
2. Go through your Tag manager and change triggers. Also pretty easily achieved.
3. Wait for a new crawl from Cookie Information, then export a list of cookies found and where they were seen and go through your razor code. When you find what is setting the cookies handle it.
4. Repeat #3 until satisfied.

I hope you enjoyed this post. Feel free to drop a comment below with questions or ideas.

Optimizely (Episerver) Integrations Addon Development Website Improvements Tips and Tricks

RECENT POSTS

CodeArt ApS
Teknikerbyen 5, 2830 Virum, Denmark
Email: info@codeart.dk
Phone: +45 26 13 66 96
CVR: 39680688