Adding Facebook Pixel or Google Analytics

Tracking of a Facebook Pixel or a Google Analytics account only works if you use a special url, to better guarantee the privacy of the visitor: instead of the link https://guts.events/xxxxxx you **MUST** use https://**widget.**guts.events/xxxxxx. This widget link can also be used as a *source url* to embed ticket-shop(s) on your website.

Facebook Pixel

In order to connect your Facebook Pixel with this link, you have to add your Pixel ID as a so-called query parameter to your link, as follows: "***fb=123456789101112***". If your Pixel ID is ***"123456789101112"***, you add this to the query parameter ***"fb="***. The link will become like this:

https://widget.guts.events/xxxxxx?fb=123456789101112

Google Analytics

In order to connect your Google Analytics with this link you have to add your GA ID as a query parameter to your link, as follows: "***ga=UA-123456***". If your Pixel ID is ***"UA-123456"***, you add this to the query parameter ***"ga="***. The link will become like this:

https://widget.guts.events/xxxxxx?ga=UA-123456

Multiple query parameters

Does your url have a query parameter? For example ***?c=00AAEE***, to set the main color of your ticket shop? Then you MUST add additional query parameters by using the "&" symbol, instead of "?". Only the first parameter start with "?".
Example:
https://widget.guts.events/xxxxxx**?**c=00AAEE**&**fb=123456789101112**&**ga=UA-123456

NOTE: DO NOT share the link the browser redirects to if you click on the link in the dashboard!

Adding Google Tag Manager

Introduction

- A Google Tag Manager (GTM) ID can be given to each event/group of events, whereby, if the visitor accepts cookies, various events and data are forwarded to this specific GTM account, up to a successful payment.
- GUTS Tickets supports all official tags & trackers that GTM offers.
- The most commonly used "Sandbox" Community Template scripts are also supported (see below) and support for new Community Template scripts can be provided upon request.
- New triggers and specific data can be supported on request.
- It is your responsibility to set up your Google Tag Manager properly.
- You can create a Google Tag Manager account for free at https://tagmanager.google.com and documentation can be found at https://support.google.com/tagmanager.

Add to an event / url

Tracking via your own GTM only works if you use a special url, to better guarantee the privacy of the visitor: instead of the link https://guts.events/xxxxxx you **MUST** use https://**widget.**guts.events/xxxxxx. This widget link can also be used as a *source url* to embed ticket-shop(s) on your website.

To then connect your GTM to this link, you must add your GTM ID as a so-called query parameter to the link, in the form of ***gtm=GTM-AB123456***. If your GTM ID is "*GTM-AB123456*", add it to the query parameter "*gtm=*". The link will then look like this:

https://widget.guts.events/xxxxxx?gtm=GTM-AB123456

- NOTE: DO NOT share the link the browser redirects to when you click on the link in the dashboard!
- NOTE: You cannot use the above Facebook Pixel/Google Analytics parameters in combination with your own Google Tag Manager. If you want to use GTM, you will also have to add Facebook Pixel and/or Google Analytics to your GTM.

Allowed pixels & scripts

Allowed:

- All templates of Google Tag Manager
- Some Community Templates (sandboxedScripts)

Blocked:

- Custom Scripts (customScripts)
- Community Templates that are not whitelisted

You can use Community Templates, but if it requires an external Javascript file, we may need to whitelist it first. You can whitelist these external scripts by contacting us. We will check the code to see if it is from a trusted source.

Events

The following events are forwarded to Google Tag Manager and can be used as a trigger to activate a template/pixel.

Section	Event	Description	Data Layer Variables
*	*Init*	Visitor has accepted cookies and a link with GTM has been made	dataLayer.push({event:"CookiesConsentAccept"})
*	*PageView*	Visitor visits a (new) page	dataLayer.push({ event: "PageView", pagePath: "/", pageRoute: "*{Sectie: Queue/Shop/Account/Order}*", gateSlug: "*{gateslug}*", shopSlug: "*{shopslug}*", eventName: “{the name of an event}”, eventSubname: “{subtitle of an event}” })
Queue	*QueueNotStarted*	Visitor visits queue that has not yet been opened	dataLayer.push({event: "QueueNotStarted"})
Queue	*QueueOpen*	Visitor visits queue that is open	dataLayer.push({event: "QueueOpen"})
Queue	*QueueEnter*	Visitor registers in queue	dataLayer.push({event: "QueueEnter"})
Queue	*QueueWaiting*	Visitor is in the queue	dataLayer.push({event: "QueueWaiting"})
Queue	*QueuePaused*	Queue is paused	dataLayer.push({event: "QueuePaused"})
Queue	*QueueSoldout*	All tickets for which Visitor is in the queue are sold out	dataLayer.push({event: "QueueSoldout"})
Queue	*QueueTurn*	Visitor in the queue is next	dataLayer.push({event: "QueueTurn"})
Shop	*ShowEventInfo*	Visitor clicks on "more info" about the event	dataLayer.push({event: "ShowEventInfo"})

Shop	*ShowFloorplan*	Visitor clicks on interactive floorplan	dataLayer.push({event: "ShowFloorplan"})

Shop	*AddProduct*	Visitor adds a product to the shopping cart	dataLayer.push({
			    event: “AddProduct”,
			    name: “{name of product}”,
			    category: “{name of category}”,
			    amount: {number of products added},
			    ecommerce: {
			        currencyCode: '{currency code}',
			        add: {
			          products: [{
			            name: '{name of product}',
			            id: '{id of product}',
			            price: '{name of product}',
			            category: {name of category},
			            quantity: {number of products added}
			           }]
			        }
			      }
			})

Shop	*RemoveProduct*	Visitor removes a product from the shopping cart	dataLayer.push({
			    event: “RemoveProduct”,
			    name: “{name of product}”,
			    category: “{name of category}”,
			    amount: {number of products added},
			    ecommerce: {
			        remove: {
			          products: [{
			            name: '{name of product}',
			            id: '{id of product}',
			            price: '{name of product}',
			            category: {name of category},
			            quantity: {number of products removed}
			          }]
			        }
			    }
			})

Shop	*Checkout*	Visitor clicks on "order"	dataLayer.push({
			  event: "Checkout",
			  currency: "*EUR*",
			  tickets: *1*,
			  ticketsAmount: *21.21*,
			  upsells: *0*,
			  upsellAmount: *0*,
			  total: *1*,
			  totalAmount: *21.21,*

			products: [
			{
			name: “*{name of product}*”,
			price: *{price of a product}*,
			quantity: *{number of products},*
			type: *{‘ticket’ or ‘upsell’}*
			}
			],
			ecommerce: {
			currencyCode: 'EUR',
			checkout: {
			products: [{
			name: '{name of product}',
			id: '{id of product}',
			price: '{name of product}',
			category: '{name of category}',
			quantity: {number of products removed}
			}]
			}
			}
			})

Account	*Login*	Visitor logs in	dataLayer.push({ event: "Login", age: *null*, gender: *null*, city: "", country: "" email: “”, first_name: “”, last_name: “”, mobile_number: “”}) })
Account	*Signup*	New visitor creates account	dataLayer.push({event: "Signup"})
Order	*OptIn*	Visitor accepts the privacy opt-in of the organization	dataLayer.push({event: "OptIn"})
Order	*Payment*	Visitor chooses payment method	dataLayer.push({event: "Payment"})
Order	*Purchase*	Payment successful	dataLayer.push({ event: "Purchase", currency: “EUR”, tickets: {number of total tickets}, ticketsAmount: {amount of tickets}, upsells: {number of upsell products},

			upselAmount: {amount of upsell product},
			total: {total product: tickets and upsell},
			totalAmount: {total amount all products: tickets and upsell}",
			order_id: {id of order},
			status: "{if order is paid or not}*",
			daysInAdvance: {days before event},
			products: [
			{
			name: "*{name of product}*",
			price: *{price of a product}*,
			quantity: *{number of products}*,
			type: *{'ticket' or 'upsell'}*
			}
			],
			ecommerce: {
			purchase': {
			actionField: {
			id: {id of order},
			revenue: {total amount all products: tickets and upsell},
			},
			products: [{
			name: '{name of product}',
			id: '{id of product}',
			price: '{name of product}',
			category: {name of category},
			quantity: {number of products removed}
			}]
			}
			}
			})
			
			** pending-timeout means we can't confirm if the payment was actually successful yet since some payment methods might take some time to confirm.*

Pageviews

The *PageView* steps for the funnel are (ideally) as follows, once the visitor opens the event widget/url(s):

1. https://widget.guts.events/{gate-id}/
   - If queue is on: Visitor enters phone number and email address and joins queue.
2. https://widget.guts.events/{gate-id}/
   - If queue is on: Visitor's turn after a few minutes and then chooses the desired event.

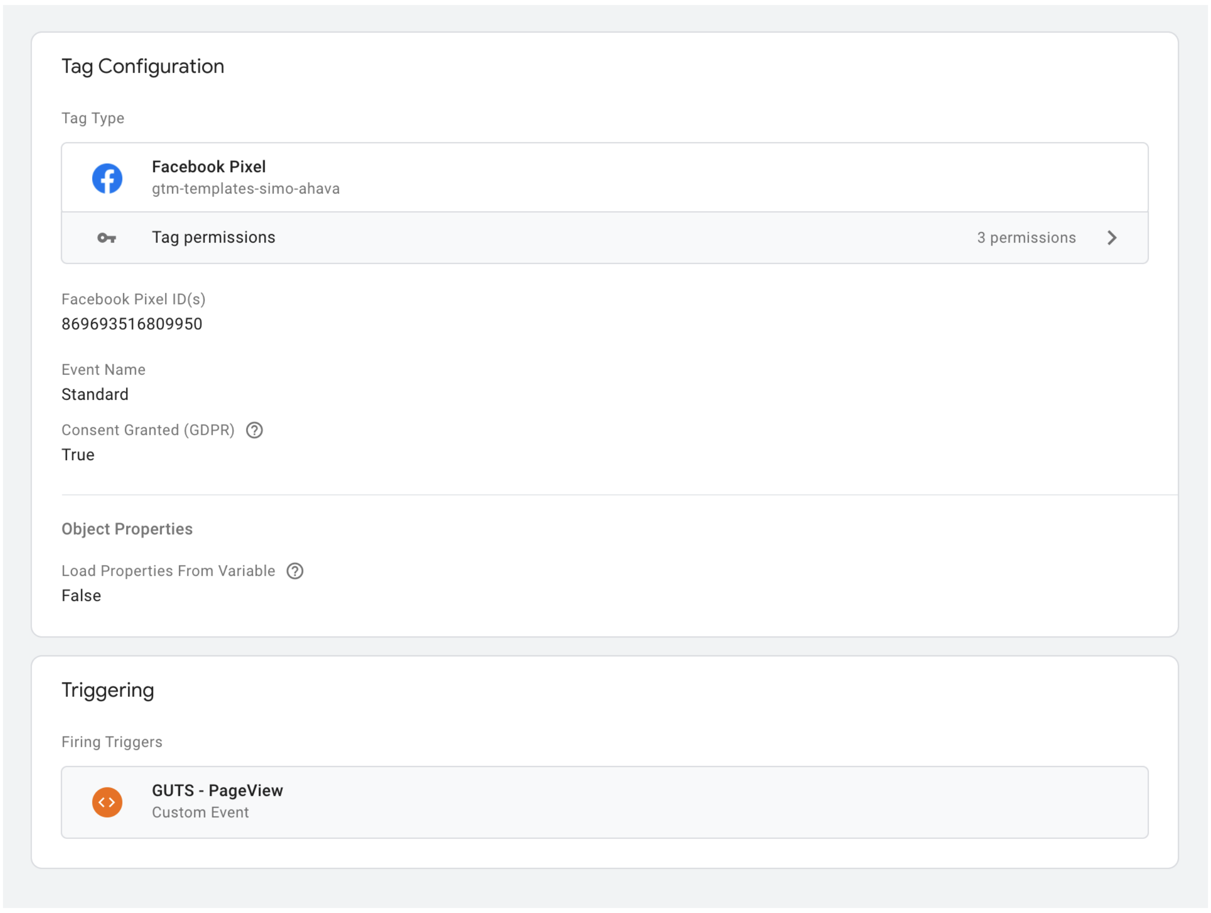
3. https://widget.guts.events/{gate-id}/
    - If no queue: Visitor has their turn immediately and sees a list of events, and then chooses desired event/if 1 event: visitor automatically goes to shop of event.
4. https://widget.guts.events/{gate-id}/{shop-id}/
    - Visitor visits the event page and can choose between a map or a list of tickets ()
5. https://widget.guts.events/{gate-id}/{shop-id}/checkout/
    - Visitor selects the number of tickets and clicks "Order".
6. https://widget.guts.events/account/
    - Visitor checks the entered mobile number and requests a verification code.
7. https://widget.guts.events/account/
    - Visitor receives a verification code by SMS and fills it in.
8. https://widget.guts.events/profile/
    - As a new customer, a visitor is shown a form, which must be completed.
9. https://widget.guts.events/{gate-id}/{shop-id}/shop/orders/{order-id}/
    - Visitor gets an overview of the order and clicks "Pay".
10. https://widget.guts.events/{gate-id}/{shop-id}/shop/orders/{order-id}/pay/
    - Visitor receives an overview of payment methods and selects one.
11. Visitor is redirected to chosen payment methods and settles (external page or app)
12. https://widget.guts.events/{gate-id}/{shop-id}/shop/orders/{order-id}/status/
    - Visitor will see a status page if the payment is successful - The "Purchase" event is therefore forwarded separately so that you know that the payment has actually been successful.

Pixel integration example: Facebook Pixel

Adding a tracker through our Google Tag Manager integration works a bit different then just adding it directly to a website like you normally would. Out of the box, no triggers or data will be sent to your trackers. You have to set up a pageview pixel and an ecommerce pixel.

For the basic functionalities of Facebook Pixel pgaeview tracking, at least the *PageView* trigger will have to be added, so that every new page that the visitor uses will be forwarded to Facebook (the standard “All Pages” trigger from GTM will not work!):

1. Add the *Facebook Pixel* as a template to your GTM Account. You can find this in the Community Gallery.
2. Add a new Trigger: a *Custom Event* with the event: *PageView*. You can name this anything you’d like, but it is important that the “Custom Event” maps with one of the triggers we provide (in the table above).
3. Add the Facebook Pixel as a new Tag, and enter your Facebook Pixel ID. As Event Name add *PageView*.
4. Add to the created Tag the previously created trigger with event: *PageView*.
5. Publish your changes and from now on all page visits of visitors to your event(s) will be forwarded to Facebook (provided these visitors have accepted cookies).


Tag Configuration

Tag Type

Facebook Pixel
gtm-templates-simo-ahava

Tag permissions — 3 permissions

Facebook Pixel ID(s)
869693516809950

Event Name
Standard

Consent Granted (GDPR)
True

Object Properties

Load Properties From Variable
False

Triggering

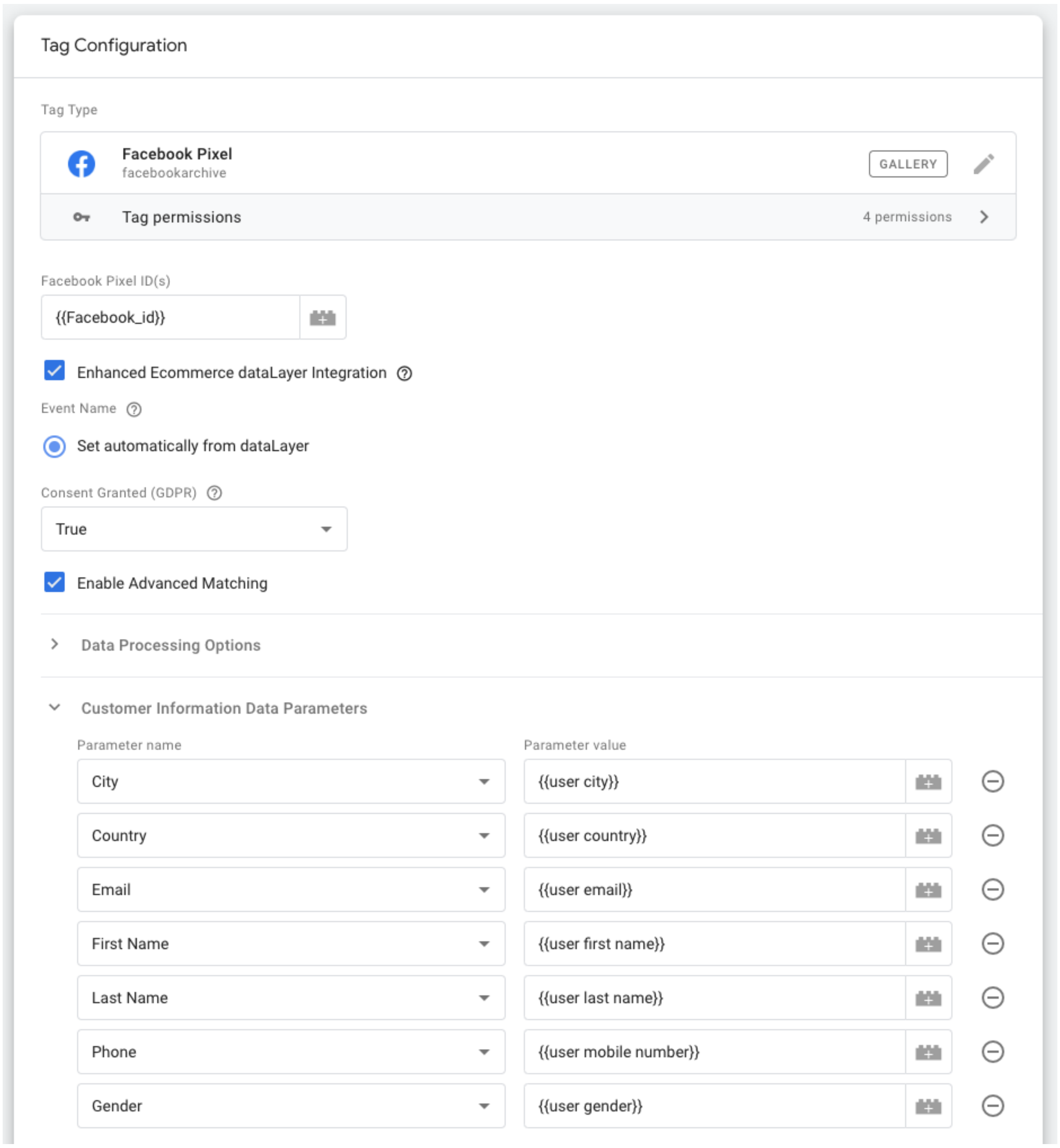
Firing Triggers

GUTS - PageView
Custom Event

Facebook Pixel with Ecommerce dataLayer Integration

For every event that contains an ecommerce object the Facebook Pixel can be used with the Enhanced Ecommerce dataLayer integration.

1. Add the *Facebook Pixel* as a template to your GTM Account. You can find this in the Community Gallery.
2. Add new Triggers: Custom *Events* with the event: *AddProduct, RemoveProduct. Login, Purchase*. You can name this anything you'd like, but it is important that the "Custom Event" maps with one of the triggers we provide (in the table above).
3. Add the Facebook Pixel as a new Tag and enter your Facebook Pixel ID. Enable "Enhanced Ecommerce dataLayer Integration" and "set automatically from dataLayer". Also "Enable Advanced Matching" can be used. See user parameter values in de tabel above.
4. Add the created Triggers previously (AddProduct, RemoveProduct, Login and Purchase) to the Tag.
5. Publish your changes and from now on all ecommerce events from a visitor will be forwarded to Facebook (provided these visitors have accepted cookies).



Tag Configuration

Tag Type

Facebook Pixel
facebookarchive

GALLERY

Tag permissions — 4 permissions

Facebook Pixel ID(s)

{{Facebook_id}}

✓ Enhanced Ecommerce dataLayer Integration

Event Name

● Set automatically from dataLayer

Consent Granted (GDPR)

True

✓ Enable Advanced Matching

Data Processing Options

Customer Information Data Parameters

Parameter name	Parameter value
City	{{user city}}
Country	{{user country}}
Email	{{user email}}
First Name	{{user first name}}
Last Name	{{user last name}}
Phone	{{user mobile number}}
Gender	{{user gender}}