

---

# **Piwik PRO Marketing Suite Documentation**

*Release 10.0*

**Piwik PRO**

**Jan 31, 2020**



---

## Contents

---

<b>1</b>	<b>Analytics (new)</b>	<b>3</b>
<b>2</b>	<b>Analytics</b>	<b>15</b>
<b>3</b>	<b>Audience Manager</b>	<b>45</b>
<b>4</b>	<b>Consent Manager</b>	<b>55</b>
<b>5</b>	<b>Tag Manager</b>	<b>73</b>
<b>6</b>	<b>Platform</b>	<b>75</b>
<b>7</b>	<b>SDK</b>	<b>83</b>
<b>8</b>	<b>Glossary</b>	<b>107</b>
	<b>Index</b>	<b>109</b>



*PPAS version: 10.0 (see changelog)*



## 1.1 Columns

This article documents core columns available in the *HTTP API*. Additional columns may become available through *Integrations*.

### 1.1.1 Metrics

The table below lists core metrics that may be used in queries. Additional metrics may be created using dimension transformations.

Table 1: Base Metrics

Metric Name	Column ID	Type
Events	events	int
Page views	page_views	int
Unique page views	unique_page_views	int
Entries	entries	int
Exits	exits	int
Bounces	bounces	int
Sessions	sessions	int
Visitors	visitors	int
% of returning visitors	returning_visitors_rate	float
Users	users	int
Visitor IPs	visitor_ips	int
Outlinks	outlinks	int
Unique outlinks	unique_outlinks	int
Downloads	downloads	int
Unique downloads	unique_downloads	int
Searches	searches	int

Continued on next page

Table 1 – continued from previous page

Metric Name	Column ID	Type
Unique searches	unique_searches	int
Custom events	custom_events	int
Unique custom events	unique_custom_events	int
Content impressions	content_impressions	int
Unique content impressions	unique_content_impressions	int
Content interactions	content_interactions	int
Unique content interactions	unique_content_interactions	int
Goal conversions	goal_conversions	int
Ecommerce conversions	ecommerce_conversions	int
Ecommerce abandoned carts	ecommerce_abandoned_carts	int
Unique purchases	unique_purchases	int
Entry rate	entry_rate	float
Exit rate	exit_rate	float
Exit rate events	exit_rate_events	float
Bounce rate	bounce_rate	float
Bounce rate	bounce_rate_events	float
Content interaction rate	content_interaction_rate	float
Goal conversion rate	goal_conversion_rate	float
Ecommerce conversion rate	ecommerce_conversion_rate	float
Events per session	events_per_session	float

## 1.1.2 Dimensions

The table below lists core dimensions that may be used in queries.

Table 2: Base Dimensions

Dimension Name	Column ID	Scope	Type	Nullable	Enum Table
User ID	user_id	session	str	false	
Returning visitor	visitor_returning	session	[int, str]	false	visitor_retu
Session number	visitor_session_number	session	int	false	
Days since last session	visitor_days_since_last_session	session	int	true	
Days since first session	visitor_days_since_first_session	session	int	true	
Days since order	visitor_days_since_order	session	int	true	
Events in session	session_total_events	session	int	false	
Session time	session_total_time	session	int	false	
Page views in session	session_total_page_views	session	int	false	
Outlinks in session	session_total_outlinks	session	int	false	
Downloads in session	session_total_downloads	session	int	false	
Site searches in session	session_total_site_searches	session	int	false	
Custom events in session	session_total_custom_events	session	int	false	
Content impressions in session	session_total_content_impressions	session	int	false	
Content interactions in session	session_total_content_interactions	session	int	false	
Goal conversions in session	session_total_goal_conversions	session	int	false	
Ecommerce conversions in session	session_total_ecommerce_conversions	session	int	false	
Abandoned carts in session	session_total_abandoned_carts	session	int	false	
Unique page views in session	session_unique_page_views	session	int	false	
Unique outlinks in session	session_unique_outlinks	session	int	false	
Unique downloads in session	session_unique_downloads	session	int	false	



Table 2 – continued from previous page

Dimension Name	Column ID	Scope	Type	Nullable	Enum Table
Unique site searches in session	session_unique_searches	session	int	false	
Unique custom events in session	session_unique_custom_events	session	int	false	
Unique content impressions in session	session_unique_content_impressions	session	int	false	
Unique content interactions in session	session_unique_content_interactions	session	int	false	
Shopping stage	session_ecommerce_status	session	[int, str]	false	session_econ
Source	source	session	str	false	
Medium	medium	session	str	false	
Source/Medium	source_medium	session	str	false	
Keyword	keyword	session	str	false	
Channel	referrer_type	session	[int, str]	false	referrer_tyt
Referrer URL	referrer_url	session	str	false	
Campaign name	campaign_name	session	str	false	
Campaign ID	campaign_id	session	str	false	
Campaign content	campaign_content	session	str	false	
Operating system	operating_system	session	[str, str]	true	operating_sy
Operating system version	operating_system_version	session	str	false	
Browser engine	browser_engine	session	str	false	
Browser name	browser_name	session	[str, str]	true	browser_name
Browser version	browser_version	session	str	false	
Browser language	browser_language_iso639	session	[str, str]	true	browser_lang
Device type	device_type	session	[int, str]	true	device_type
Device brand	device_brand	session	[str, str]	true	device_brand
Device model	device_model	session	str	false	
Resolution	resolution	session	str	true	
Resolution width	resolution_width	session	int	true	
Resolution height	resolution_height	session	int	true	
PDF plugin	plugin_pdf	session	int(0,1)	false	
Flash plugin	plugin_flash	session	int(0,1)	false	
Java plugin	plugin_java	session	int(0,1)	false	
Director plugin	plugin_director	session	int(0,1)	false	
QuickTime plugin	plugin_quicktime	session	int(0,1)	false	
RealPlayer plugin	plugin_realplayer	session	int(0,1)	false	
Windows Media Player plugin	plugin_windowsmedia	session	int(0,1)	false	
Gears plugin	plugin_gears	session	int(0,1)	false	
Silverlight plugin	plugin_silverlight	session	int(0,1)	false	
Cookie support	plugin_cookie	session	int(0,1)	false	
Continent	location_continent_iso_code	session	[str, str]	true	location_cor
Country	location_country_name	session	[str, str]	true	ISO 3166-2 code
Subdivision	location_subdivision_1_name	session	[str, str]	true	ISO 3166-2 code
Subdivision 2	location_subdivision_2_name	session	[str, str]	true	ISO 3166-2 code
City	location_city_name	session	[int, str]	true	unique identifiers
Designated market area	location_metro_code	session	[str, str]	true	location_met
Latitude	location_latitude	session	float	true	
Longitude	location_longitude	session	float	true	
Provider	location_provider	session	str	false	
Organization	location_organization	session	str	false	
Session exit URL	session_exit_url	session	str	false	
Session exit title	session_exit_title	session	str	false	
Session entry URL	session_entry_url	session	str	false	

Table 2 – continued from previous page

Dimension Name	Column ID	Scope	Type	Nullable	Enum Table
Session entry title	session_entry_title	session	str	false	
Session second URL	session_second_url	session	str	false	
Session second title	session_second_title	session	str	false	
Event type	event_type	event	[int, str]	false	event_type.
Page URL	event_url	event	str	false	
Page title	event_title	event	str	false	
Outlink URL	outlink_url	event	str	false	
Download URL	download_url	event	str	false	
Search keyword	search_keyword	event	str	false	
Search category	search_category	event	str	false	
Search results count	search_results_count	event	int	true	
Custom event category	custom_event_category	event	str	false	
Custom event action	custom_event_action	event	str	false	
Custom event name	custom_event_name	event	str	false	
Custom event value	custom_event_value	event	float	true	
Content name	content_name	event	str	false	
Content piece	content_piece	event	str	false	
Content target	content_target	event	str	false	
Content interaction	content_interaction	event	str	false	
Previous page view URL	previous_event_url	event	str	false	
Previous page view title	previous_event_title	event	str	false	
Next page view URL	next_event_url	event	str	false	
Next page view title	next_event_title	event	str	false	
Time on page	time_on_page	event	int	true	
Page generation time	page_generation_time	event	float	true	
Goal name	goal_id	event	[int, str]	true	goal IDs from An
Goal revenue	goal_revenue	event	float	true	
Lost revenue	lost_revenue	event	float	true	
Order ID	order_id	event	str	false	
Item count	item_count	event	int	true	
Revenue	revenue	event	float	true	
Revenue (Subtotal)	revenue_subtotal	event	float	true	
Revenue (Tax)	revenue_tax	event	float	true	
Revenue (Shipping)	revenue_shipping	event	float	true	
Revenue (Discount)	revenue_discount	event	float	true	
Time until DOM is ready	timing_dom_interactive	event	int	true	
Time to interact	timing_event_end	event	int	true	
Event custom dimension 1	event_custom_dimension_1	event	str	false	
Event custom dimension 2	event_custom_dimension_2	event	str	false	
Event custom dimension 3	event_custom_dimension_3	event	str	false	
Event custom dimension 4	event_custom_dimension_4	event	str	false	
Event custom dimension 5	event_custom_dimension_5	event	str	false	
Event custom variable key 1	event_custom_variable_key_1	event	str	false	
Event custom variable value 1	event_custom_variable_value_1	event	str	false	
Event custom variable key 2	event_custom_variable_key_2	event	str	false	
Event custom variable value 2	event_custom_variable_value_2	event	str	false	
Event custom variable key 3	event_custom_variable_key_3	event	str	false	
Event custom variable value 3	event_custom_variable_value_3	event	str	false	
Event custom variable key 4	event_custom_variable_key_4	event	str	false	

Table 2 – continued from previous page

Dimension Name	Column ID	Scope	Type	Nullable	Enum Table
Event custom variable value 4	event_custom_variable_value_4	event	str	false	
Event custom variable key 5	event_custom_variable_key_5	event	str	false	
Event custom variable value 5	event_custom_variable_value_5	event	str	false	
Session custom dimension 1	session_custom_dimension_1	session	str	false	
Session custom dimension 2	session_custom_dimension_2	session	str	false	
Session custom dimension 3	session_custom_dimension_3	session	str	false	
Session custom dimension 4	session_custom_dimension_4	session	str	false	
Session custom dimension 5	session_custom_dimension_5	session	str	false	
Session custom variable key 1	session_custom_variable_key_1	session	str	false	
Session custom variable value 1	session_custom_variable_value_1	session	str	false	
Session custom variable key 2	session_custom_variable_key_2	session	str	false	
Session custom variable value 2	session_custom_variable_value_2	session	str	false	
Session custom variable key 3	session_custom_variable_key_3	session	str	false	
Session custom variable value 3	session_custom_variable_value_3	session	str	false	
Session custom variable key 4	session_custom_variable_key_4	session	str	false	
Session custom variable value 4	session_custom_variable_value_4	session	str	false	
Session custom variable key 5	session_custom_variable_key_5	session	str	false	
Session custom variable value 5	session_custom_variable_value_5	session	str	false	
Time of redirections	redirections_time	event	int	true	
Domain Lookup Time	domain_lookup_time	event	int	true	
Server Connection Time	server_connection_time	event	int	true	
Server Response Time	server_response_time	event	int	true	
Page Rendering Time	page_rendering_time	event	int	true	
IPv4 address	ipv4_address	session	ipv4	true	
Website Name	website_name	session	[str, str]	false	website UUID

**Note:** Please note that the number of available custom slots (dimensions, variables) depends on your organisation's configuration.

### 1.1.3 Transformations

The tables below list all transformations that may be used to transform dimensions to metrics or different dimensions.

Table 3: Dimension To Metric Transformations

Transformation Name	Transformation ID	Source Types	Result Type
Unique Count	unique_count	str	int
Min	min	float, int	(as source)
Max	max	float, int	(as source)
Average	average	bool, float, int	float
Median	median	float, int	(as source)
Sum	sum	float, int	(as source)

Table 4: Dimension To Dimension Transformations

Transformation Name	Transformation ID	Source Types	Result Type
Date To Day	to_date	date, datetime	date
Date To Start Of Hour	to_start_of_hour	datetime	datetime
Date To Start Of Week	to_start_of_week	date, datetime	date
Date To Start Of Month	to_start_of_month	date, datetime	date
Date To Start Of Quarter	to_start_of_quarter	date, datetime	date
Date To Start Of Year	to_start_of_year	date, datetime	date
Date To Hour Of Day	to_hour_of_day	datetime	int
Date To Day Of Week	to_day_of_week	date, datetime	int
Date To Month Number	to_month_number	date, datetime	int
URL To Path	to_path	str	str
URL To Domain	to_domain	str	str
URL Strip Query String	strip_qs	str	str

## 1.2 Integrations

Documents in this section describe the structure of data provided by third-party integrations.

### 1.2.1 Google Search Console

The *HTTP API* supports querying Google Search Console data just like the internal analytics data.

---

**Note:** You must configure the Google Search Console integration before any data from it will become available. This can be done in the **Settings / Integrations** application's section.

---

#### Metrics

The table below lists metrics provided by Google Search Console integration.

Table 5: Google Search Console Metrics

Metric Name	Column ID	Type
Clicks (search engine)	search_engine_clicks	int
Impressions (search engine)	search_engine_impressions	int
CTR (search engine)	search_engine_ctr	float
Average position (search engine)	search_engine_average_position	float

#### Dimensions

The table below lists dimensions provided by Google Search Console integration.

Table 6: Google Search Console Dimensions

Dimension Name	Column ID	Type	Nul- lable	Enum Table
Source/Medium	source_medium	str	False	
Channel	referrer_type	[int, str]	False	referrer_type.json
Referrer URL	referrer_url	str	False	
Device type	device_type	[int, str]	True	device_type.json
Continent	loca- tion_continent_iso_code	[str, str]	True	location_continent_iso_code. json
Country	location_country_name	[str, str]	True	ISO 3166-2 codes (e.g. "PL")
Session entry URL	session_entry_url	str	False	
Timestamp	timestamp	date	False	
Search engine key- word	search_engine_keyword	str	False	

## Mixed Queries

It is possible to request both internal analytics and Google Search Console metrics in a single query (for example: "Sessions" and "Clicks (search engine)"), however **only the common dimensions listed below** may be used in such queries.

Table 7: Common Dimensions

Dimension Name	Column ID	Type	Nul- lable	Enum Table
Source/Medium	source_medium	str	False	
Channel	referrer_type	[int, str]	False	referrer_type.json
Referrer URL	referrer_url	str	False	
Device type	device_type	[int, str]	True	device_type.json
Continent	loca- tion_continent_iso_code	[str, str]	True	location_continent_iso_code. json
Country	location_country_name	[str, str]	True	ISO 3166-2 codes (e.g. "PL")
Session entry URL	session_entry_url	str	False	
Timestamp	timestamp	date	False	

**Warning:** Using dimensions that are not explicitly listed in the table above in such queries (either as query columns or as filters) will result in a **Bad Request** response.

## 1.3 HTTP API

## 1.4 Metrics Mapping

Names of metrics used in API are different in Analytics classic and Analytics new. If you're migrating to the Analytics new API then below metrics mapping table will be handfull to you. You can find there a list of metrics used in Analytics classic and their corresponding names in the Analytics new.

### 1.4.1 Simple Metrics

Metric name	Legacy API	New API
Events	nb_actions	{"column_id": "events"}
Sessions	nb_visits	{"column_id": "sessions"}
Visitors	nb_uniq_visitors	{"column_id": "visitors"}
Users	nb_users	{"column_id": "users"}
Page views	nb_pageviews nb_hits	{"column_id": "page_views"}
Unique page views	nb_uniq_pageviews	{"column_id": "unique_page_views"}
Outlinks	nb_outlinks	{"column_id": "outlinks"}
Unique outlinks	nb_uniq_outlinks	{"column_id": "unique_outlinks"}
Downloads	nb_downloads	{"column_id": "downloads"}
Unique downloads	nb_uniq_downloads	{"column_id": "unique_downloads"}
Searches	-	{"column_id": "searches"}
Unique searches	nb_searches	{"column_id": "unique_searches"}
Custom events	nb_events	{"column_id": "custom_events"}
Unique custom events	-	{"column_id": "unique_custom_events"}
Content impressions	nb_impressions	{"column_id": "content_impressions"}
Unique content impressions	-	{"column_id": "unique_content_impressions"}
Content interactions	nb_interactions	{"column_id": "content_interactions"}
Unique content interactions	-	{"column_id": "unique_content_interactions"}
Content interaction rate	interaction_rate	{"column_id": "content_interaction_rate"}

Continued on next page

Table 8 – continued from previous page

Metric name	Legacy API	New API
Goal conversions	nb_conversions	{"column_id": "goal_conversions"}
Ecommerce conversions	<div style="border: 1px solid black; padding: 5px;"> <b>Warning:</b> ecommerce conversion was reported as goal conversion for goal_id 0                 </div>	{"column_id": "ecommerce_conversions"}
Goal conversions (specific goal)		{"column_id": "goal_conversions", "goal_id": 1}
Ecommerce abandoned carts	-	{"column_id": "ecommerce_abandoned_carts"}
Goal conversion rate	conversion_rate	{"column_id": "goal_conversion_rate"}
Ecommerce conversion rate	-	{"column_id": "ecommerce_conversion_rate"}
Entries	entry_nb_visits	{"column_id": "entries"}
Entry rate	-	{"column_id": "entry_rate"}
Exits	exit_nb_visits	{"column_id": "exits"}
Exit rate	exit_rate	{"column_id": "exit_rate"}
Exit rate (events)	<div style="border: 1px solid black; padding: 5px;"> <b>Warning:</b> definition switches depending on report                 </div>	{"column_id": "exit_rate_events"}
Bounces		bounce_count
Bounce rate	bounce_rate	{"column_id": "bounce_rate"}
Bounce rate (events)	<div style="border: 1px solid black; padding: 5px;"> <b>Warning:</b> definition switches depending on report                 </div>	{"column_id": "bounce_rate_events"}
% of returning visitors		-
Visitor IPs	-	{"column_id": "visitor_ips"}

Continued on next page

Table 8 – continued from previous page

Metric name	Legacy API	New API
Events per session	nb_actions_per_visit	{"column_id": "events_per_session"}
<div style="border: 1px solid black; padding: 5px; display: inline-block;"> <b>Warning:</b>                      does not include all event types                 </div>		
Unique purchases	-	{"column_id": "unique_purchases"}

### 1.4.2 Calculated Metrics

**Warning:** This table does not include every single possible combination of a dimension and a transformation, just some common examples.

Metric name	Legacy API	New API
Sum of goal revenue	revenue ecommerce revenue was reported as goal revenue for goal_id	{"column_id": "goal_revenue", "transformation_id": "sum"}
Sum of ecommerce revenue	0	{"column_id": "revenue", "transformation_id": "sum"}
Sum of goal revenue (specific goal)	goal_<idGoal>_revenue	{"column_id": "goal_revenue", "transformation_id": "sum", "goal_id": 1}
Average generation time	avg_time_generation	{"column_id": "page_generation_time", "transformation_id": "average"}
Max generation time	max_time_generation	{"column_id": "page_generation_time", "transformation_id": "max"}
Average time on page	avg_time_on_page	{"column_id": "time_on_page", "transformation_id": "average"}
Sum of time on page	sum_time_spent	{"column_id": "time_on_page", "transformation_id": "sum"}
Sum of session time	sum_visit_length	{"column_id": "session_total_time", "transformation_id": "sum"}
Average session time	avg_time_on_site	{"column_id": "session_total_time", "transformation_id": "average"}
Max events in session	max_actions	{"column_id": "session_total_events", "transformation_id": "max"}
Sum of custom events value	sum_event_value	{"column_id": "custom_event_value", "transformation_id": "sum"}
Average custom events value	avg_event_value	{"column_id": "custom_event_value", "transformation_id": "average"}



### 1.4.3 Not available

Name	Legacy API	Closest equivalent in Analytics new
Number of sessions that converted a goal	nb_visits_converted	Sessions metric with filter goal_conversions > 0
Number of custom events which had a value set	nb_events_with_value	Custom events metric with filter custom event value > 0
Number of hits that included generation time information	nb_hits_with_time_generation	Page views metric with filter page_generation_time > 0
Number of unique visitors that started their visit on this page	entry_nb_uniq_visitors	-
Number of page views for sessions that started on this page	entry_nb_actions	Entries metric (all entries are page views now)
Time spent, in seconds, by sessions that started on this page	entry_sum_visit_length	-
Number of sessions that started on this page, and bounced	entry_bounce_count	Bounces metric
Number of unique visitors that ended their visit on this page	exit_nb_uniq_visitors	-
Sum of daily unique visitors over days in the period	sum_daily_nb_uniq_visitors	No longer relevant, unique visitors are calculated across any period
Sum of daily unique visitors that started their visit on this page	sum_daily_entry_nb_uniq_visitors sum_daily_exit_nb_uniq_visitors	
Number of times this action was done after a site search	nb_hits_following_search	-



### 2.1 JavaScript tracking API

The following API allows the user to:

- track page views
- track visits on multiple domains and subdomains
- track e-commerce events (successful orders, cart changes, product and category views)
- track content impressions
- manage custom variables to use them later
- track clicked links to external domains and download files

#### 2.1.1 Installing Tracking code

There are two ways of installing a tracking code:

##### Installing tracking code via Tag Manager

This is the easiest and recommended way of tracking code installation. When Tag Manager is added to the site - it automatically publishes tracking code (using “Piwik PRO Analytics template”).

If you do not have Tag Manager on your website yet, follow this procedure to install it: #. Sign in to your PPAS with your admin or Super User account. #. Click on the menu button on the top left. #. Click on the “Websites” position. #. Choose the website for which you want to implement a tracking code. #. Select the “Installation” tab. #. The Tag Manager code snippet for your website is displayed under the “Website code for asynchronous tags” or “Website code for synchronous tags”.

## Installing tracking code via code snippet.

### Installing tracking code via code snippet

Installation via snippet should only be carried out if the Tag Manager is not available or when options of “Piwik PRO Analytics template” do not let you configure your use case.

---

**Note:** We highly recommend using the template from the Tag Manager to set up tracking for the Analytics module (including customizations).

---

---

**Note:** Basic configuration will setup a single domain configuration. For other options, see: [Alternative multi-domain configurations](#).

---

This code should be added in the head section of the page just before the closing `</head>` tag. Additionally, the snippet must be configured in the following way:

- String `XXX-XXX-XXX-XXX-XXX` should be replaced with *app ID* (e.g. `efcd98a5-335b-48b0-ab17-bf43f1c542be`).
- String `https://your-instance-name.piwik.pro/` should be replaced with your PPAS instance address.

```
<!-- Piwik -->
<script type="text/javascript">
  var _paq = _paq || [];
  _paq.push(["trackPageView"]);
  _paq.push(["enableLinkTracking"]);
  (function() {
    var u="https://your-instance-name.piwik.pro/";
    _paq.push(["setTrackerUrl", u+"ppms.php"]);
    _paq.push(["setSiteId", "XXX-XXX-XXX-XXX-XXX"]);
    var d=document, g=d.createElement("script"), s=d.getElementsByTagName("script")
    ↪[0];
    g.type="text/javascript"; g.async=true; g.defer=true; g.src=u+"ppms.js"; s.
    ↪parentNode.insertBefore(g,s);
  })();
</script>
```

Deprecated since version 5.5.1: Older installations using `piwik.php` and `piwik.js` filenames are deprecated.

This code initializes the Analytics tracker in following ways:

1. Initializes the global `_paq.push` command queue that schedules commands to be run when the Analytics tracker library is loaded.
2. Schedules basic configuration of Analytics tracker using `_paq.push`.
3. Creates a `<script>` tag that asynchronously loads the Analytics tracker library.

When loading, the snippet is added on the page. The Analytics tracker will start tracking *user* actions starting with page view.

### Alternative multi-domain configurations

## Tracking domains and all subdomains

To track all data between domain and all its subdomains, we must use cookies configured with the following snippet:

```
_paq.push(["setTrackerUrl", u+"ppms.php"]);
_paq.push(["setSiteId", "XXX-XXX-XXX-XXX-XXX"]);

// Share the tracking cookie across example.com, www.example.com, subdomain.example.
↪com, ...
_paq.push(["setCookieDomain", "*.example.com"]);

// Tell Piwik the website domain so that clicks on these domains are not tracked as
↪"Outlinks"
_paq.push(["setDomains", "*.example.com"]);

_paq.push(["trackPageView"]);
```

Deprecated since version 5.5.1: Older installations using `piwik.php` and `piwik.js` filenames are deprecated.

## Tracking multiple domains as one site

To set up tracking between multiple domains, you must use multiple functions `setDomains` to set a list of domains and `enableCrossDomainLinking` to enable cross domain linking:

```
_paq.push(["setDomains", domains]);
```

### domains

**Required** array Domains array, with wildcards

```
_paq.push(["enableCrossDomainLinking"]);
```

## Tracking subdirectories of domain as separate websites

To differentiate parts of a website as another site, you must configure tracker this way:

```
_paq.push(["setSiteId", "App1"]);
_paq.push(["setTrackerUrl", u+"ppms.php"]);
_paq.push(["trackPageView"]);
```

Afterwards, you can change configuration for selected paths and track them as another site:

```
_paq.push(["setSiteId", "App2"]);

_paq.push(["setCookiePath", "/data/something_useful"]);

_paq.push(["setDomains", "example.com/data/something_useful"]);

_paq.push(["setTrackerUrl", u+"ppms.php"]);
_paq.push(["trackPageView"]);
```

In this way, all actions tracked on `/data/something_useful` will be tracked for App2 instead of App1.

If you wish to track a group of pages as separate site, you can use the wildcard in the `setDomains` function.

Deprecated since version 5.5.1: Older installations using `piwik.php` and `piwik.js` filenames are deprecated.

## Navigation timing page performance metrics

To set up page performance metrics gathering use the `setTimingDataSamplingOnPageLoad` function:

```
_paq.push(["setTimingDataSamplingOnPageLoad", updateTimingDataOnPageLoadSampling]);
```

### `updateTimingDataOnPageLoadSampling`

**Required** `integer` Value between 1 and 100 describing the percentage for data sampling

```
_paq.push(["setTimingDataSamplingOnPageLoad", 33]);
```

## 2.1.2 Command queue

Loading snippet creates the following API function:

`_paq.push` (*command*)

JavaScript API interface.

### Arguments

- **command** (*Array<string>*) – Array containing command *name* followed by its arguments. The number of arguments and their function depend on command.

**Return type** `undefined`

## 2.1.3 Commands

### Trigger tracking on demand

### Trigger custom event

Trigger (custom) events bound to user actions:

```
_paq.push(["trackEvent", category, action, name, value, dimensions]);
```

#### **category**

**Required** `string` Event category.

#### **action**

**Required** `string` Event action.

#### **name**

**Optional** `string` Event name.

#### **value**

**Optional** `number` Event value.

#### **dimensions**

**Optional** `object` *Custom dimensions* which should be tracked using this action. It can set multiple dimensions at once. Dimensions are defined as object properties using the `dimension{ID}` notation.

Example:

```
{
  dimension1: "example value",
  dimension2: "example value"
}
```

Example of usage (tracking when the user clicks on the cancel button with exit intent):

```
_paq.push(["trackEvent", "Exit intent", "Click on button", "Cancel"]);
```

## Track goal conversion

Allows the manual tracking of goal conversion. Used in *Goals - Days to Conversion* report. Command:

```
_paq.push(["trackGoal", goal_name, goal_value, dimensions]);
```

### goal\_name

**Required** string Goal Name

### goal\_value

**Optional** number Tracked conversion value.

### dimensions

**Optional** object *Custom dimensions* which should be tracked using this action. Dimensions are defined as object properties using the `dimension{ID}` notation.

Example:

```
{
  dimension1: "example value",
  dimension2: "example value"
}
```

Example of usage:

```
_paq.push(["trackGoal" 1, 15]);
```

## Ecommerce tracking

### Adding Ecommerce item

To add an e-commerce item (for example to track changes in the user's cart using `trackEcommerceCartUpdate`), use the `addEcommerceItem` function:

```
_paq.push(["addEcommerceItem", productSKU, productName, productCategory, productPrice,
↪ productQuantity]);
```

---

**Note:** This function does not send any data to the *Analytics*. It only prepares E-commerce cart/order state to be sent with *trackEcommerceOrder* or *trackEcommerceCartUpdate*.

---

### productSKU

**Required** string Product stock-keeping unit.

**productName**

**Optional** string Product name.

**productCategory**

**Optional** array/string Product category, can be written as Array with up to 5 elements.

**productPrice**

**Optional** number with product price.

**productQuantity**

**Optional** number with product quantity.

**Warning:** Product SKU, names and categories should be URL encoded.

**Warning:** The state of the cart is not maintained across the visit. You must add all products after each page view.

Example of usage:

```
_paq.push(["addEcommerceItem", "craft-311", "Unicorn Iron on Patch", "Crafts & Sewing", "499", "3"]);
```

## Tracking Ecommerce order

To successfully track the e-commerce order(s) (on the checkout page, for example) use the `trackEcommerceOrder` function:

```
_paq.push(["trackEcommerceOrder", orderId, orderGrandTotal, orderSubTotal, orderTax, orderShipping, orderDiscount]);
```

**orderId**

**Required** string Unique order ID.

**orderGrandTotal**

**Required** number Order Revenue grand total - tax, shipping and discount included.

**orderSubTotal**

**Optional** number Order sub total - without shipping.

**orderTax**

**Optional** number Order tax amount.

**orderShipping**

**Optional** number Order shipping costs.

**orderDiscount**

**Optional** number Order discount amount.

Example of usage:

```
_paq.push(["trackEcommerceOrder", "3352", 499, 399, 0, 100]);
```



## Updating cart

To update the user cart (when the user adds new products or removes them from cart) use the `trackEcommerceCartUpdate` function:

```
_paq.push(["trackEcommerceCartUpdate", cartAmount]);
```

### **cartAmount**

**Required** number Cart amount (sum of products).

**Warning:** Before updating the tracking cart, be sure to add all products in the cart by using `addEcommerceItem` first (including the ones that were previously in the cart). Then, use this function.

Example of usage:

```
_paq.push(["trackEcommerceCartUpdate", 250]);
```

## Tracking product / category view

If you wish to track when the user enters the product site or is browsing products category, use the `setEcommerceView` function:

```
_paq.push(["setEcommerceView", productSKU, productName, productCategory, ↵
↵productPrice]);
```

### **productSKU**

**Required** string/boolean Product stock-keeping unit. False for tracking category.

### **productName**

**Optional** string/boolean Product name. False for tracking category.

### **productCategory**

**Optional** array/string Product category, can be written as Array with up to 5 elements.

### **productPrice**

**Optional** number Product price.

**Warning:** Product SKU, names and categories should be URL encoded.

Example of usage:

```
_paq.push(["setEcommerceView", "craft-311", "Unicorn Iron on Patch", "Crafts & Sewing
↵", "499"]);
```

## Custom Variables

Deprecated since version 5.5: We strongly advise using custom dimensions.

## Adding / Editing Custom Variable

To set a custom variable that can be used later, use the `setCustomVariable` function:

```
_paq.push(["setCustomVariable", index, name, value, scope]);
```

### index

**Required** number Index from 1 to 5 where the variable is stored

### name

**Required** string Name of the variable

### value

**Optional** string Value of the variable limited to 200 characters.

### scope

**Optional** string Scope of the variable, "visit" or "page". The default value is "visit"

---

**Note:** A Custom Variable with the scope set on "visit" will be saved for visit, you don't need to save it for every page.

---

**Warning:** Index is separate for each variable scope.

Example of usage:

```
_paq.push(["setCustomVariable", 1, "AspectRatio", "16:9", "visit"]);
```

## Removing Custom Variable

To remove the custom variable, you can use the `deleteCustomVariable` function:

```
_paq.push(["deleteCustomVariable", index, scope]);
```

### index

**Required** number Index from 1 to 5 where the variable is stored

### scope

**Optional** string Scope of the variable, "visit" or "page". The default value is "visit"

Example of usage:

```
_paq.push(["deleteCustomVariable", 1, "visit"]);
```

## Accessing Custom Variable

You can access custom variables by providing a function that will use the `getCustomVariable` function:

```
_paq.push([ function() {  
    var customVariable = this.getCustomVariable(index, scope );  
}]);
```

`getCustomVariable` (*index* [, *scope* ])

### Arguments

- **index** (*number*) – **Required** Number from 1 to 5 where variable is stored
- **scope** (*string*) – **Optional** Scope of the variable, “visit” or “page”. Default value is “visit”

Example of usage:

```
_paq.push([ function() {
  var customVariable = this.getCustomVariable(1, "visit" );
  console.log(customVariable);
}]);
```

## Custom Dimensions

### Tracking Custom Dimension

If you wish to set a custom dimension to use it in tracking functions, use the `setCustomDimension` function:

```
_paq.push(["setCustomDimension", customDimensionID, customDimensionValue]);
```

#### **customDimensionID**

**Required** number ID of dimension

#### **customDimensionValue**

**Required** string Value of Custom Dimension - limited to 255 characters.

**Warning:** When you set a Custom Dimension, that value will be used in all tracking requests within a page load.

**Warning:** This function does not send any data to the *Analytics*. It sets a Custom Dimension to be sent with following events (e.g. page view, ecommerce events, outlink or download events).

Example of usage:

```
_paq.push(["setCustomDimension", 3, "loginStatus"]);
```

### Retrieving Custom Dimension

You can access custom dimension by providing a function that will use the `getCustomDimension` function:

```
_paq.push([ function() {
  var customDimension = this.getCustomDimension(index);
}]);
```

#### **getCustomDimension** (*index*)

##### Arguments

- **index** (*number*) – **Required** Index of custom dimension

Example of usage:

```
_paq.push([ function() {  
    var customDimension = this.getCustomDimension(1);  
    console.log(customDimension);  
}]);
```

## Content Tracking

Content Tracking tracks how many times specific elements were rendered/visible. It can be used to measure if the ad placement was visible or if the user saw the end of an article.

To track content, it has to have the `data-track-content` attribute or `piwikTrackContent` CSS class attached to it.

### Tracking all content impressions within a page

To track all content impression, you can use the `trackAllContentImpressions` function. If this function is invoked multiple times, it will not send duplicated data unless the `trackPageView` was used between invocations:

```
_paq.push(["trackAllContentImpressions"]);
```

### Tracking all visible content impressions

To track all visible content impressions you can use the `trackVisibleContentImpressions` function.

Code:

```
_paq.push(["trackVisibleContentImpressions", checkOnScroll, watchInterval]);
```

#### **checkOnScroll**

**Optional** `boolean` If `true`, it will check new visible content impressions on the scroll event. Default: `true`.

---

**Note:** It will not detect content blocks placed in a scrollable element.

---

#### **watchInterval**

**Optional** `number` Interval, in milliseconds between checking for new visible content. Periodic checks can be disabled for performance reasons by setting `0`. Default value: `750`.

**Warning:** Both options cannot be changed after the initial setup.

Example of usage:

```
_paq.push(["trackVisibleContentImpressions", true]);
```

Example of usage:

```
_paq.push(["trackVisibleContentImpressions", false, 500]);
```

## Tracking only content impressions for specific page part

To track impressions on part of a webpage that will be populated after a page load, you can use the `trackContentImpressionsWithinNode`:

```
_paq.push(["trackContentImpressionsWithinNode", domNode]);
```

### **domNode**

**Required** `domNode` DOM element that will have impression DOM elements with `data-track-content` attribute

It can be used with `trackVisibleContentImpressions` to track only visible content impressions

Example of usage:

```
var element = document.querySelector("#impressionContainer");
_paq.push(["trackContentImpressionsWithinNode", element]);
```

## Track interactions manually with auto detection

If you wish to trigger an interaction manually (for example on click event), you can do it using `trackContentInteractionNode`, just add this code in the action you want to track:

```
_paq.push(["trackContentInteractionNode", domNode, contentInteraction]);
```

### **domNode**

**Required** `domNode` Node marked as content block or containing content blocks. If no content block is found - nothing will be tracked.

### **contentInteraction**

**Optional** string Name of interaction (e.g. "click"). Default value: "Unknown".

Example of use

```
<button onClick="function() {_paq.push(['trackContentInteractionNode', this, 'clicked
↪']);}">Click me!</button>
```

## Track impression manually

If you wish to trigger tracking impressions entirely manually, you can use the `trackContentImpression`

```
_paq.push(["trackContentImpression", contentName, contentPiece, contentTarget]);
```

### **contentName**

**Required** string Name of Content Impression

### **contentPiece**

**Required** string Name of Content Impression Piece

### **contentTarget**

**Required** string URL of Content Impression Target

Example of use:

```
_paq.push(["trackContentImpression", "trackingWhitepaper", "document", "http://
↪cooltracker.tr/whitepaper"]);
```

## Track user interaction manually

If you wish to trigger tracking interactions entirely manually, you can use the `trackContentInteraction`. Use it as a function inside listener on event:

```
_paq.push(["trackContentInteraction", contentInteraction, contentName, contentPiece, ↵  
↵contentTarget]);
```

### **contentInteraction**

**Required** string Name of interaction (e.g. "click").

### **contentName**

**Required** string Name of Content Impression

### **contentPiece**

**Required** string Name of Content Impression Piece

### **contentTarget**

**Required** string URL of Content Impression Target

Example of use:

```
_paq.push(["trackContentImpression", "clicked", "trackingWhitepaper", "document", ↵  
↵"http://cooltracker.tr/whitepaper"]);
```

**Warning:** Use this function in conjunction with `trackContentImpression`, as it can only be mapped with an impression by linking `contentName`. It does not map automatically as `trackContentInteractionNode`.

## Download and Outlink Tracking

### Tracking Outlink

To enable the Download & Outlink tracking, run:

```
_paq.push(["enableLinkTracking"]);
```

just after the first `trackPageView` or `trackEvent`

---

**Note:** All Outlinks are tracked automatically. As `enableLinkTracking` is part of the default snippet.

---

### Ignoring alias domains

To ignore internal outlinks from alias domains, use the `setDomains` function to define internal domains and subdomains, you can use the wildcard:

```
_paq(["setDomains", domains]);
```

### **domains**

**Required** array Domains written as strings, \* are accepted.

Example of usage:

```
_paq(["setDomains", ["*.example.com", "*.example.co.uk"]]);
```

## Force Tracking using CSS class

To track clicking a link as an outlink using the CSS class, simply add the `piwik_link` class to the link element. It will then be considered as an outlink, even if it points to the same domain.

This class name can be changed, use `setLinkClasses` to define which CSS class should be tracked:

```
_paq.push(["setLinkClasses", className]);
```

### **className**

**Required** string CSS class that should be tracked instead of `piwik_link`

Example of usage:

```
_paq(["setLinkClasses", "track-this-link"]);
```

## Force Tracking using JS function

If you wish to use JS to force the outlink to be tracked, you can add the `trackLink` function on element `onClick` attribute:

```
_paq.push(["trackLink", linkAddress, "link", dimensions]);
```

### **linkAddress**

**Required** string Address that link points to.

### **dimensions**

**Optional** object *Custom dimension* that should be tracked with this action. Can be multiple dimensions. Written as object property using `dimension{ID}` notation.

Example:

```
{
  dimension1: "example value",
  dimension2: "example value"
}
```

Example of usage

```
<button onClick="function(){_paq.push(['trackLink', 'http://www.example.com/example',
↪'link']);}">
  Click me!
</button>
```

## Tracking Downloads

### Default extensions recognized as download

The following extensions are tracked as download by default:

7z	aac	arc	arj	apk	asf	asx	avi	bin	bz	bz2	csv	deb	dmg	doc
exe	flv	gif	gz	gzip	hqx	jar	jpg	jpeg	js	mp2	mp3	mp4	mpg	mov
movie	msi	msp	odb	odf	odg	odp	ods	odt	ogg	ogv	pdf	phps	png	ppt
qt	qtm	ra	ram	rar	rpm	sea	sit	tar	tbz	tbz2	tgz	torrent	txt	wav
wma	wmv	wpd	xls	xml	z	zip								

## Adding extension to default extensions

You can add an extension to the default extensions list using the `addDownloadExtensions` function:

```
_paq.push(["addDownloadExtensions", extensions]);
```

### extensions

**Required** `string|Array<string>` Extensions separated by `|` for example `"7z|apk|mp4"` can also be written as an Array, for example: `["7z", "apk", "mp4"]`

Example of usage:

```
_paq.push(["addDownloadExtensions", "mhj|docx"]);
```

## Replacing default extensions list

Default extensions list can be overwritten using the `setDownloadExtensions` function:

```
_paq.push(["setDownloadExtensions", extensions]);
```

### extensions

**Required** `string|Array<string>` Extensions separated by `|` for example `"7z|apk|mp4"` can also be written as an Array, for example: `["7z", "apk", "mp4"]`

Example of usage:

```
_paq.push(["setDownloadExtensions", "7z|apk|mp4"]);
```

## Force Tracking download using CSS class

To track clicking a link as a download using css class simply add the `piwik_download` class to link element.

This class name can be changed, use `setDownloadClasses` to define which CSS class should be tracked:

```
_paq.push(["setDownloadClasses", className]);
```

### className

**Required** `string` CSS class that should be tracked instead of `piwik_download`

Example of usage:

```
_paq(["setDownloadClasses", "track-this-link-for-download"]);
```



## Force Tracking download using JS function

If you wish to use JS to force tracking download, you can add `trackLink` function on element `onClick` attribute:

```
_paq.push(["trackLink", linkAddress, "download", dimensions]);
```

### **linkAddress**

**Required** string Address that link points to.

### **dimensions**

**Optional** object *Custom dimension* that should be tracked with this action. Can be multiple dimensions. Written as object property using `dimension{ID}` notation.

Example:

```
{
  dimension1: "example value",
  dimension2: "example value"
}
```

Example of usage

```
<button onClick="function(){_paq.push(['trackLink', 'http://www.example.com/example.
↪xrt', 'download']);}">
  Click me!
</button>
```

## Setting link tracking delay

After each outbound/download link click, there is a small delay introduced, after which the browser navigates to the new URL. This ensures there is enough time to track link interactions. That delay is set by default to 500ms. To modify it you can use the `setLinkTrackingTimer` function:

```
_paq.push(["setLinkTrackingTimer", time]);
```

### **time**

**Required** number Time in ms between user action (click) and changing a website (for outlink) or downloading a file.

## Disabling tracking

You can disable download and outlink tracking for links using CSS classes, simply add `piwik_ignore` css class.

To disable using CSS class you can use `setIgnoreClasses` function:

```
_paq.push(["setIgnoreClasses", className]);
```

### **className**

**Required** string|Array<string> Css class name that will be ignored, can be written as Array with CSS classes.

## User ID Management

User ID enables merging user data that is collected between many devices and browsers.

### Setting UserId

You must provide unique user-id for every user. To set user ID for tracked data use `setUserId` function:

```
_paq.push(["setUserId", userID]);
```

#### **userID**

**Required** `string` Unique, non-empty permanent ID of the user in application.

### Reset UserId

When `UserId` becomes unavailable anymore (eg. user logged out) you may clean the value out with `resetUserId` function:

```
_paq.push(["resetUserId"]);
```

### Miscellaneous

#### Custom page name

We are using the current page URL as the page title. To change this use the `setDocumentTitle` function:

```
_paq.push(["setDocumentTitle", title]);
```

#### **title**

**Required** `string` Title to show instead of URL

Example of usage:

```
_paq.push(["setDocumentTitle", document.title]);
```

#### Measuring user time spent on web page

When the user will enter a single page during a visit, we will assume that his total time spent on the website was 0 ms. To measure that time more accurately you can use the `enableHeartBeatTimer` function:

```
_paq.push(["enableHeartBeatTimer", beat]);
```

#### **beat**

**Required** `number` Time in seconds between cyclical heartbeat requests, default 30

Example of usage:

```
_paq.push(["enableHeartBeatTimer", 50]);
```

#### Tracking internal searches

To track search requests on your site use the `trackSiteSearch` function:

```
_paq.push(["trackSiteSearch", keyword, category, searchCount, dimensions]);
```

**keyword**

**Optional** string Keyword that was searched

**category**

**Optional** string Category selected in search engine - you can set it to false when not used.

**searchCount**

**Optional** number Results on the results page - you can set it to false when not used.

**dimensions**

**Optional** object *Custom dimension* that should be tracked with this action. Can be multiple dimensions. Written as object property using `dimension{ID}` notation.

Example:

```
{
  dimension1: "example value",
  dimension2: "example value"
}
```

Example of usage:

```
_paq.push(["trackSiteSearch", "test", false, 20]);
```

## Tracking user anonymously

To track visitor anonymously (without consent) use the `setUserIsAnonymous` function:

```
_paq.push(["setUserIsAnonymous", isAnonymous]);
```

**isAnonymous**

**Required** boolean Flag that sets anonymous tracking on and off

Example of usage:

```
_paq.push(["setUserIsAnonymous", true]);
```

To disable tracking user anonymously (after visitor gave consent) use `deanonymizeUser` function:

```
_paq.push(["deanonymizeUser"]);
```

## Gathering navigation timing page performance metrics

To set up page performance metrics gathering use the `setTimingDataSamplingOnPageLoad` function:

```
_paq.push(["setTimingDataSamplingOnPageLoad", updateTimingDataOnPageLoadSampling]);
```

**updateTimingDataOnPageLoadSampling**

**Required** integer Value between 1 and 100 describing the percentage for data sampling

It sets another request triggered on `load`, after `trackPageView` setting timing data. Normally we try to use `trackPageView` as soon as possible, not to lose any actions but since it's usually before the full page was loaded then we don't have complete data about timing. You may either trigger `trackPageView` after `onLoad` event or enable this option so the followup request containing all timing values is sent after `onLoad`.

Argument to this function represents data sampling percentage (with possible integer values between 0 and 100).

Example of usage:

```
_paq.push(["setTimingDataSamplingOnPageLoad", 0]); // disables timing data collection
_paq.push(["setTimingDataSamplingOnPageLoad", 5]); // this is the default setting,
↳ uses 5% data sampling
_paq.push(["setTimingDataSamplingOnPageLoad", 30]); // enables 30% data sampling
↳ (only around 30% of all tracked actions will collect timing data if possible)
_paq.push(["setTimingDataSamplingOnPageLoad", 100]); // enables 100% data sampling
↳ (which means that all tracked actions will collect timing data if possible)
```

IMPORTANT: in order for this setting to make effect `setTimingDataSamplingOnPageLoad()` should be used before the `trackPageView()` function

IMPORTANT: if enabled, timing data is collected only when page view lasted longer than the time it takes the page to load no partial information is stored, all metrics or nothing

## 2.2 Tracker Object Functions

This document describes all the functions available for the Tracker object and how to create its instances. This enables users to track multiple Trackers at once.

### 2.2.1 Accessing Tracker Object

To access Tracker object instance you must use the `Piwik.getTracker` function

`Piwik.getTracker` (*trackerUrl*, *siteId*)  
Getter for Analytics Tracker instance.

#### Arguments

- **trackerUrl** (*string*) – Required URL for Tracker
- **siteId** (*string*) – Required Site ID that will be linked to tracked data.

**Returns** Analytics Tracker instance

To access internal instance of the Tracker used for asynchronous tracking you must use the `Piwik.getAsyncTracker` function

`Piwik.getAsyncTracker` (*trackerUrl*, *siteId*)  
Getter for Analytics Tracker instance.

#### Arguments

- **trackerUrl** (*string*) – Required URL for Tracker
- **siteId** (*string*) – Required Site Id that will be linked to tracked data.

**Returns** Analytics Tracker instance

### 2.2.2 Tracking functions

`trackPageView` (*[customPageTitle]*)  
Tracks a visit on the page that the function was run on.

#### Arguments

- **customPageTitle** (*string*) – **Optional** Custom page title, for example `document . title`

**trackEvent** (*category*, *action* [, *name*, *value* ])

Tracks events that should not trigger on page loading, but only when user performs an action

#### Arguments

- **category** (*string*) – **Required** Category of event.
- **action** (*string*) – **Required** Event action, for example "link click".
- **name** (*string*) – **Optional** Event name, for example "Cancel button".
- **value** (*string*) – **Optional** Event value.

**trackGoal** (*idGoal* [, *customRevenue*, *customData* ])

Manually tracks goal (conversion).

#### Arguments

- **idGoal** (*int/string*) – **Required** Id of goal.
- **customRevenue** (*int/float*) – **Optional** Revenue value
- **customData** (*mixed*) – **Optional** Object with *Custom dimensions*.

**trackSiteSearch** (*keyword* [, *category*, *resultCount* ])

The function that tracks internal site searches.

#### Arguments

- **keyword** (*string*) – **Required** String containing keyword that was searched.
- **category** (*string/boolean*) – **Optional** String with category selected in search engine, can set it to false when not used.
- **searchCount** (*number/boolean*) – **Optional** Number of results on the results page, can be set to false when not used.

**enableHeartBeatTimer** (*delay*)

When the user will enter a single page during a visit, we will assume that his total time spent on the website was 0 ms. To measure that time more accurately you can use the `enableHeartBeatTimer` function:

```
:param number delay: **Required** Time in seconds between cyclical heartbeat_
↪requests, default ``30``
```

**enableCrossDomainLinking** ()

The function that will enable cross domain linking. That way visitors across domains will be linked.

**setCrossDomainLinkingTimeout** (*timeout*)

The function will change default time in which two visits across domains will be linked.

#### Arguments

- **timeout** (*number*) – **Required** Time in seconds in which two visits across domains will be linked. Default is 180.

## 2.2.3 Ecommerce tracking

**addEcommerceItem** (*productSKU* [, *productName*, *productCategory*, *price*, *quantity* ])

The function that adds ecommerce item, can be used when adding and removing items from cart.

#### Arguments

- **productSKU** (*string*) – **Required** String with product stock-keeping unit.
- **productName** (*string*) – **Optional** String with product name.
- **productCategory** (*Array<string>*) – **Optional** Product category, can be written as Array with up to 5 elements.
- **price** (*string*) – **Optional** String with product price.
- **quantity** (*string*) – **Optional** String with product quantity.

**trackEcommerceOrder** (*orderId*, *orderGrandTotal* [, *orderSubTotal*, *orderTax*, *orderShipping*, *orderDiscount* ])

The function that tracks Ecommerce order, also tracks all items previously added.

#### Arguments

- **orderId** (*string*) – **Required** Unique order ID.
- **orderGrandTotal** (*number*) – **Required** Order Revenue grand total - tax, shipping and discount included.
- **orderSubTotal** (*number*) – **Optional** Order sub total - without shipping.
- **orderTax** (*number*) – **Optional** Order tax amount.
- **orderShipping** (*number*) – **Optional** Order shipping costs.
- **orderDiscount** (*number*) – **Optional** Order discount amount.

**trackEcommerceCartUpdate** (*grandTotal*)

The function that tracks the shopping cart value. Use this each time there is a change in cart as the last function after adding cart items.

#### Arguments

- **grandTotal** (*number*) – **Required** Order Revenue grand total - tax, shipping and discount included.

**setEcommerceView** (*productSKU* [, *productName*, *categoryName*, *productPrice* ])

The function to track product or category page view, must be followed by the `trackPageView` function.

#### Arguments

- **productSKU** (*string*) – **Required** String with product stock-keeping unit.
- **productName** (*string*) – **Optional** String with product name.
- **productCategory** (*Array<string>*) – **Optional** Product category, can be written as Array with up to 5 elements.
- **price** (*string*) – **Optional** String with product price.

## 2.2.4 Custom variables

**setCustomVariable** (*index*, *name*, *value*, *scope*)

The function that sets a custom variable to be used later.

#### Arguments

- **index** (*string*) – **Required** Number from 1 to 5 where variable is stored.
- **name** (*string*) – **Required** Name of the variable.
- **value** (*string*) – **Required** Value of the variable.

- **scope** (*string*) – **Required** Scope of the variable, "visit" or "page".

**deleteCustomVariable** (*index, scope*)

The function that will delete a custom variable.

#### Arguments

- **index** (*string*) – **Required** Number from 1 to 5 where variable is stored.
- **scope** (*string*) – **Required** Scope of the variable, "visit" or "page".

**getCustomVariable** (*index, scope*)

The function that will return the value of custom variable.

#### Arguments

- **index** (*string*) – **Required** Number from 1 to 5 where variable is stored.
- **scope** (*string*) – **Required** Scope of the variable, "visit" or "page".

**storeCustomVariablesInCookie** ()

The function will enable storing "visit" type custom variables in a first party cookie. That will enable getting them via the `getCustomVariable` function.

## 2.2.5 Custom dimensions

**setCustomDimension** (*customDimensionId, customDimensionValue*)

The function that sets a custom dimension to be used later.

#### Arguments

- **customDimensionId** (*string*) – **Required** Id of custom dimension.
- **customDimensionValue** (*string*) – **Required** Value of custom dimension.

**deleteCustomDimension** (*customDimensionId*)

The function that will delete a custom dimension.

#### Arguments

- **customDimensionId** (*string*) – **Required** Id of custom dimension.

**getCustomDimension** (*customDimensionId*)

The function that will return the value of custom dimension.

#### Arguments

- **customDimensionId** (*string*) – **Required** Id of custom dimension.

## 2.2.6 Content Tracking

### Impressions

**trackAllContentImpressions** ()

The function that will scan DOM for content blocks and tracks impressions after all page will load.

**trackVisibleContentImpressions** (*[checkOnScroll, watchInterval]*)

The function that will scan DOM for all visible content blocks and will track these impressions.

#### Arguments

- **checkOnScroll** (*boolean*) – **Optional** Enables tracking content blocks that will be visible after scroll event.

- **watchInterval** (*number*) – **Optional** Interval, in milliseconds between checking for new visible content. Periodic checks can be disabled for performance reasons by setting 0. Default value: 750.

**trackContentImpressionsWithinNode** (*domNode*)

The function that will scan domNode (with its children) for all content blocks and will track impressions.

**Arguments**

- **domNode** (*domNode*) – **Required** DOM node with content blocks (with data-track-content attribute) inside.

**trackContentImpression** (*contentName, contentPiece, contentTarget*)

The function that manually tracks content impression.

**Arguments**

- **contentName** (*string*) – **Required** String containing name of Content Impression.
- **contentPiece** (*string*) – **Required** String containing name of Content Impression Piece.
- **contentTarget** (*string*) – **Required** String containing URL of Content Impression Target.

**logAllContentBlocksOnPage** ()

The function that will print all content blocks in the console for debugging purposes.

## Interactions

**trackContentInteractionNode** (*domNode*[, *contentInteraction* ])

The function that tracks interaction within domNode. This can be used as a function inside the onClick attribute.

**Arguments**

- **domNode** (*domNode*) – **Required** Node marked as content block or containing content blocks. If no content block will be found - nothing will be tracked.
- **contentInteraction** (*string*) – **Optional** Name of interaction (e.g. "click"). Default value: "Unknown".

**trackContentInteraction** (*contentInteraction, contentName, contentPiece, contentTarget*)

The function that tracks content interaction using the given data.

**Arguments**

- **contentInteraction** (*string*) – **Required** Name of interaction (e.g. "click").
- **contentName** (*string*) – **Required** Name of Content Impression.
- **contentPiece** (*string*) – **Required** Name of Content Impression Piece.
- **contentTarget** (*string*) – **Required** URL of Content Impression Target.

## 2.2.7 Download and Outlink Tracking

**trackLink** (*url, linkType*[, *customData, callback* ])

The function that will manually track downloads or outlinks, depending on type.

**Arguments**

- **url** (*string*) – **Required** Address that link points to.



- **linkType** (*string*) – **Required** Type of link, if is set to "link" it will track an outlook, if it is set to "download" it will track a download.
- **customData** (*object*) – **Optional** Object containing *Custom dimension* that should be linked to tracked link.
- **callback** (*function*) – **Optional** The function that should be triggered after tracking link.

## Tracking Outlink

### **enableLinkTracking** (*enable*)

The function that will register all links as trackable (left and middle mouse buttons are being treated the same, right mouse button is treated as “open in a new tab”).

#### Arguments

- **enable** (*boolean*) – **Required** Set it to true to track links, false to disable tracking.

### **setLinkClasses** (*classes*)

The function that sets classes to be treated as outlinks (piwik-link is the default one).

#### Arguments

- **classes** (*array/string*) – **Required** String containing CSS class, can be written as array of strings.

## Tracking Downloads

### **setDownloadClasses** (*classes*)

The function that sets classes to be treated as downloads (piwik\_download is the default one).

#### Arguments

- **classes** (*array/string*) – **Required** String containing CSS class, can be written as array of strings.

### **setDownloadExtensions** (*extensions*)

The function that will set a list of file extensions that will automatically be recognized as a download action.

#### Arguments

- **extensions** (*array/string*) – **Required** List of extensions to be set. Can be written as string: "zip|rar" or an array: ["zip", "rar"]

### **addDownloadExtensions** (*extensions*)

The function that will add extensions to a list of known extensions to be automatically recognized as a download action.

#### Arguments

- **extensions** (*array/string*) – **Required** List of extensions to be set. Can be written as string: "zip|rar" or an array: ["zip", "rar"]

### **removeDownloadExtensions** (*extensions*)

The function that will remove extensions from a list of known extensions to be automatically recognized as a download action.

#### Arguments

- **extensions** (*array/string*) – **Required** List of extensions to be set. Can be written as string: "zip|rar" or an array: ["zip", "rar"]

**setLinkTrackingTimer** (*time*)

The function that will set delay between tracking and download;

**Arguments**

- **time** (*number*) – **Required** Delay between tracking and download, written in milliseconds.

**getLinkTrackingTimer** ()

The function that will return delay between tracking and download.

## Disabling tracking

**setIgnoreClasses** (*classes*)

The function that will set classes to be ignored in tracking download and outlinks.

**Arguments**

- **classes** (*array/string*) – **Required** String containing CSS class, can be written as array of strings.

## 2.2.8 User ID and Visitor ID Management

### User ID

**getUserId** ()

The function that will return user ID.

**setUserId** (*userId*)

The function that will set user ID to this user.

**Arguments**

- **userId** (*string*) – **Required** Unique, non-empty string preserved for each user.

**regetUserId** ()

The function that will reset user ID value.

### Visitor ID

**getVisitorId** ()

The function that will return 16 characters ID for the visitor.

**getVisitorInfo** ()

The function that will return visitor information in an array:

- new visitor flag indicating new (1) or returning (0) visitor
- visitor ID (UUID)
- first visit timestamp (Unix epoch time)
- previous visit count (0 for first visit)
- current visit timestamp (Unix epoch time)
- last visit timestamp (Unix epoch time or ' ' if N/A)
- last e-commerce order timestamp (Unix epoch time or ' ' if N/A)

## 2.2.9 Tracking cookies management

Cookies that are used by analytics are first party cookies.

### **disableCookies** ()

The function that will disable all first party cookies. Existing ones will be deleted in the next page view.

### **deleteCookies** ()

The function that will delete existing tracking cookies after the next page view.

### **hasCookies** ()

The function that will return `true` if cookies are enabled in this browser.

### **setCookieNamePrefix** (*prefix*)

The function that will set the prefix for analytics tracking cookies. Default is `"_pk_"`

#### Arguments

- **prefix** (*string*) – **Required** String that will replace default analytics tracking cookies prefix.

### **setCookieDomain** (*domain*)

The function that will set the domain for the analytics tracking cookies.

#### Arguments

- **domain** (*string*) – **Required** Domain that will be set as cookie domain. For enabling subdomain you can use wildcard sign or dot.

### **setCookiePath** (*path*)

The function that will set the analytics tracking cookies path.

#### Arguments

- **path** (*string*) – **Required** Path that will be set, default is `"/"`

### **setSecureCookie** (*bool*)

The function that will toggle the Secure cookie flag on all first party cookies (if you are using HTTPS).

#### Arguments

- **bool** (*boolean*) – **Required** If set to true it will add Secure cookie flag.

### **setVisitorCookieTimeout** (*seconds*)

The function that will set the expiration time of visitor cookies.

#### Arguments

- **seconds** (*number*) – **Required** Seconds after which the cookie will expire. Default is 13 months.

### **setReferralCookieTimeout** (*seconds*)

The function that will set the expiration time of referral cookies.

#### Arguments

- **seconds** (*number*) – **Required** Seconds after which the cookie will expire. Default is 6 months.

### **setSessionCookieTimeout** (*seconds*)

The function that will set the expiration time of session cookies.

#### Arguments

- **seconds** (*number*) – **Required** Seconds after which the cookie will expire. Default is 30 minutes.

## 2.2.10 Tracker Configuration

### `setDocumentTitle` (*[title]*)

The function that will set the document title that is being sent with tracking data.

#### Arguments

- **title** (*string*) – **Optional** String that will override default `document.title`

### `setDomains` (*domains*)

The function that will set an array of domains to be treated as local. Sub-domain wildcards are supported (e.g. `*.example.com`).

#### Arguments

- **domains** (*array<string>*) – **Required** Array of hostnames written as strings.

### `setCustomUrl` (*customUrl*)

The function that will override tracked page URL. Tracker will use current page URL if custom URL was not set.

#### Arguments

- **customUrl** (*string*) – **Required** Value that will override default URL.

### `setReferrerUrl` (*url*)

The function that will override the detected HTTP referrer.

#### Arguments

- **url** (*string*) – **Required** Value that will override HTTP referrer.

### `setApiUrl` (*url*)

The function that will set the Analytic's HTTP API URL endpoint. Usually the root directory of analytics.

#### Arguments

- **url** (*string*) – **Required** Path to Analytic's HTTP API URL

### `getPiwikUrl` ()

The function that will return the Analytic's server URL.

### `getCurrentUrl` ()

The function that will return the current URL of the page. The custom URL will be returned if set.

### `discardHashTag` (*enableFilter*)

The function that will set tracker to include or remove **'URL fragment identifier'**<[https://en.wikipedia.org/wiki/Fragment\\_identifier](https://en.wikipedia.org/wiki/Fragment_identifier)>' from tracked URLs.

#### Arguments

- **enableFilter** (*boolean*) – **Required** If set to true, URL fragment identifier will be removed from tracked URLs.

### `setGenerationTimeMs` (*generationTime*)

The function that overrides DOM Timing API provided time needed to download the page.

#### Arguments

- **generationTime** (*number*) – **Required** Time that will take to download page, in milliseconds.

### `appendToTrackingUrl` (*appendToUrl*)

The function that will append a custom string to the tracking URL.

#### Arguments

- **appendToUrl** (*string*) – **Required** String that will be added to the tracking URL.

**setDoNotTrack** (*enable*)

The function that will disable tracking users who set the Do Not Track setting.

#### Arguments

- **enable** (*boolean*) – **Required** When set to true tracking wont occur.

**killFrame** ()

The function that will block a site from being iframed.

**redirectFile** (*url*)

The function that will force the browser to load URL if the tracked web page was saved as a file.

#### Arguments

- **url** (*string*) – **Required** URL that should be loaded.

**setHeartBeatTimer** (*minimumVisitLength, heartBeatDelay*)

The function that sets how long the page has been viewed for if the minimumVisitLength is attained.

#### Arguments

- **minimumVisitLength** (*number*) – **Required** Minimum visit length in seconds.
- **heartBeatDelay** (*number*) – **Required** Update sever time threshold.

**setCampaignNameKey** (*name*)

The function that will set campaign name parameters.

#### Arguments

- **name** (*string*) – **Required** Campaign name.

**setCampaignKeywordKey** (*keyword*)

The function that will set campaign keyword parameters.

#### Arguments

- **keyword** (*array<string>*) – **Required** Keyword parameters.

## 2.2.11 Anonymization

**setUserIsAnonymous** (*isAnonymous*)

The function that will set user anonymous tracking.

#### Arguments

- **isAnonymous** (*boolean*) – **Required** Flag that sets anonymous tracking on and off.

**deanonymizeUser** ()

The function that will disable user anonymous tracking and send deanonymization request.

## 2.2.12 Advanced Usage

**addListener** (*domElement*)

The function will add a click listener to link element.

#### Arguments

- **domElement** (*DOMElement*) – **Required** Element that click will trigger logging the click automatically.

**setRequestMethod** (*method*)

The function that will set the request method.

**Arguments**

- **method** (*string*) – **Required** Method that will be used in requests. "GET" or "POST" are supported. The default is "GET"

**setCustomRequestProcessing** (*function*)

The function that will process the request content. The function will be called once the request (query parameters string) has been prepared, and before the request content is sent.

**setRequestContentType** (*contentType*)

The function that will set tracking requests Content-Type header. Used when tracking uses the "POST" method (set by `setRequestMethod`).

**Arguments**

- **contentType** (*string*) – **Required** Content-Type value to be set.

## 2.3 HTTP API

Tracking HTTP API allows sending to analytics information about Visitors page views, events and visits.

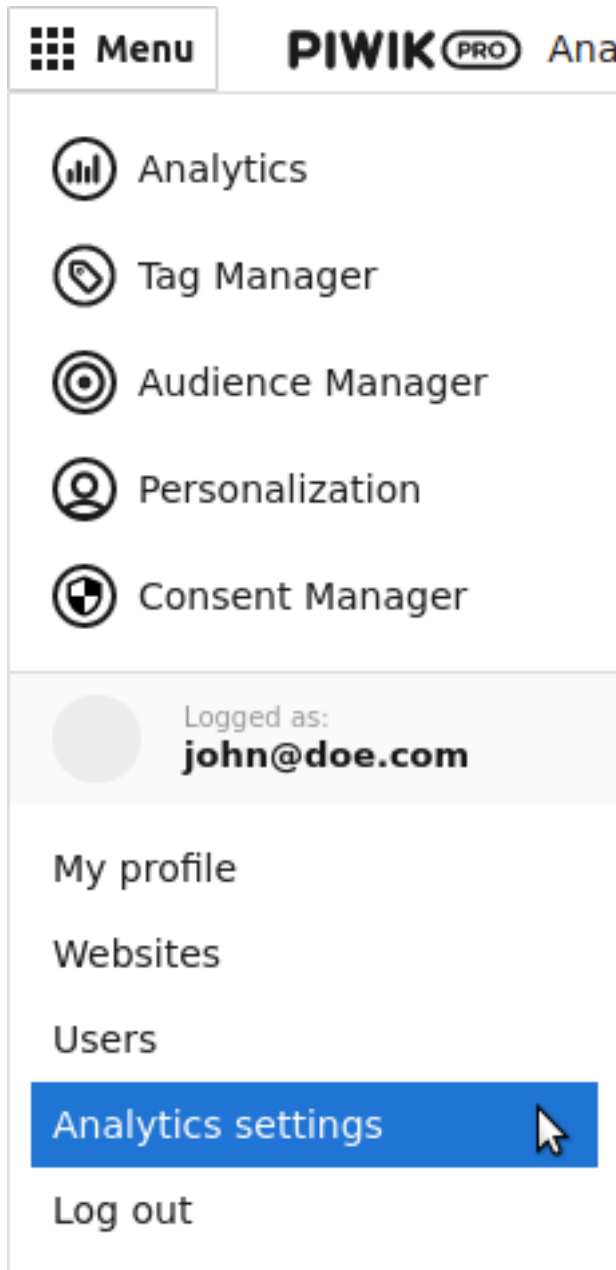
Deprecated since version 5.5.1: Endpoint `/piwik.php` is moved to `/ppms.php`. The old endpoint still works, but its support will be disabled at some point.

## 2.4 Reporting API

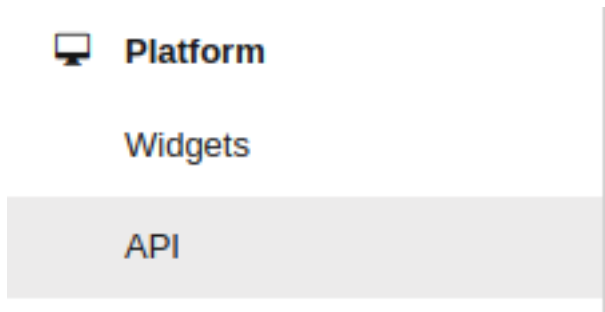
This API gives access to your analytics reports.

A short description of all available reports is available in *Analytics*. Follow these instructions to access it:

1. Login into your PPAS instance
2. Go to *Menu > Analytics settings*.



3. Select in the left menu *Platform > API*.







## 3.1 JavaScript API

This API provides access to information about *users* such as ID of *audience* they are part of and their *attributes*. It also allows you to update their *attributes*.

### 3.1.1 Loading snippet

Add the following snippet on your page to start using this API. It should be added just before the first API usage.

Changed in version 10.0: Loading snippet changed to allow multiple initializations. Now separate scripts can initiate and use this API without interference.

Configuration:

- String `XXX-XXX-XXX-XXX-XXX` should be replaced with *app ID* (e.g. `efcd98a5-335b-48b0-ab17-bf43f1c542be`).
- String `https://your-instance-name.piwik.pro/` should be replaced with your PPAS instance address. (please note that it's used in 2 places in the snippet).

Code:

```
<script>
  (function(a,d,g,h,b,c,e){a[b]=a[b]||{};a[b][c]=a[b][c]||{};if(!a[b][c][e])
  ↪{a[b][c][e]=function(){(a[b][c][e].q=a[b][c][e].q||[]).push(arguments)};var f=d.
  ↪createElement(g);d=d.getElementsByTagName(g)[0];f.async=1;f.src=h;d.parentNode.
  ↪insertBefore(f,d)}})
  (window,document,"script","https://your-instance-name.piwik.pro/audiences/static/
  ↪widget/audience-manager.api.min.js","ppms","am","api");

  ppms.am.api("create","XXX-XXX-XXX-XXX-XXX","your-instance-name.piwik.pro");
</script>
```

This code initializes the API interface in the following ways:

1. Creates a `<script>` tag that asynchronously loads the Audience Manager API library.
2. Initializes the global `ppms.am.api` command queue that schedules commands to be run when the API library is loaded.
3. Schedules `create` command on `ppms.am.api` to initialize the API object with a basic PPAS configuration.

You can use the API command queue (`ppms.am.api`) immediately after step 3.

### 3.1.2 Command queue

Executing the snippet creates the following global function:

`ppms.am.api` (*command*, ...*args*)

Audience Manager API command queue.

#### Arguments

- **command** (*string*) – Command name.
- **args** – Command arguments. The number of arguments and their function depend on command.

**Returns** Commands are expected to be run asynchronously and return no value.

**Return type** undefined

### 3.1.3 Commands

All commands work in context of the current *user*. Additionally they require communication with a PPAS server and are asynchronous. Callback functions are used to provide response value or information about errors.

#### Get list of audiences user belongs to

Fetches a list of *audience* IDs the *user* belongs to.

Code:

```
ppms.am.api("getAudiences", onFulfilled, onRejected);
```

**onFulfilled** (*audience\_list*)

The fulfilment handler callback (called with result).

#### Arguments

- **audience\_list** (*Array<string>*) – **Required** Array of *audience* IDs the *user* belongs to.

Example:

```
["e8c6e873-955c-4771-9fd5-92c94577e9d9", "756e5920-422f-4d13-b73a-917f696ca288"]
```

**onRejected** (*error\_code*)

The rejection handler callback (called with error code).

#### Arguments

- **error\_code** (*string*) – **Required** Error code.

Example:

```
"server_error"
```

### Check user membership in the audience

Checks if the *user* belongs to the *audience*.

Code:

```
ppms.am.api("checkAudience", audience_id, onFulfilled, onRejected);
```

#### **audience\_id**

**Required** *string* ID of the checked *audience*.

Example:

```
"52073260-5861-4a56-be5e-6628794722ee"
```

#### **onFulfilled** (*in\_audience*)

The fulfilment handler callback (called with result).

##### Arguments

- **in\_audience** (*boolean*) – **Required** *True* when *user* is part of the *audience*, *false* otherwise.

Example:

```
true
```

#### **onRejected** (*error\_code*)

The rejection handler callback (called with error code).

##### Arguments

- **error\_code** (*string*) – **Required** Error code.

Example:

```
"server_error"
```

### Get user attributes

Fetches the *user* profile *attributes*. The *user* have to be identified by *analytics ID*.

---

**Note:** In order to secure the *PII* data, no *attribute* is returned by default. You need to put each *attribute* you want to access on *attribute whitelist* before it is returned by this command. In order to do that, go to *Audience Manager > Attributes* tab and *enable attribute* for the public API access. It is your responsibility to make sure no *user PII* data will be available via API.

---

Code:

```
ppms.am.api("getAttributes", onFulfilled, onRejected);
```

**onFulfilled** (*attributes*)

The fulfilment handler callback (called with result).

**Arguments**

- **attributes** (*Object*<string, *Object*<string, string>>) – **Required** Object containing *user attributes* divided by source.
  - *analytics* - *Object*<string, string> Contains *analytics attributes* about the *user* (e.g. browser name, browser version, country).
  - *attributes* - *Object*<string, string> Contains *custom attributes* about the *user* (e.g. first name, last name, email).

Example:

```
{
  "analytics": {
    "browser_name": "chrome",
    "country": "us"
  },
  "attributes": {
    "first_name": "James",
    "last_name": "Bond"
  }
}
```

**onRejected** (*error\_code*)

The rejection handler callback (called with error code).

**Arguments**

- **error\_code** (*string*) – **Required** Error code.

Example:

```
"server_error"
```

## Update user attributes

Creates or updates *user custom attributes*.

---

**Note:** Any *attribute* can be updated this way whether it is on the *attribute whitelist* or not.

---

Code:

```
ppms.am.api("updateAttributes", attributes, options);
```

**attributes**

**Required** *Object*<string, (string|number|object)> *Object* containing *attributes* to update:

- key (string) - *attribute* name
- value (string|number|object) - Value of the *attribute*. System will process it differently depending on its type:
  - string - overwrite the *attribute* value with the new value. If the *attribute* was not used before - creates new text *attribute*.

- number - overwrite the *attribute* value with the new value. If the *attribute* was not used before - creates new numeric *attribute*.
- object - ModificationAction using following format: {action: string, value: (string|number)}. It allows to manipulate *attribute* value using one of the following ModificationAction action values:
  - \* "set" - overwrite *attribute* value using the ModificationAction value. Works identically to the shorter versions using string or number types.
  - \* "add" - add the ModificationAction value (or 1, if not specified) to the *attribute* value.

**Note:**

- Works only on numeric *attributes*.
- ModificationAction value can be any number (including negative and fractional numbers).
- If the *attribute* was not used before - creates new numeric *attribute* and sets its value to 0 before performing action.

Example:

```
{
  "favourite_color": "black",
  "drink": "Martini",
  "code_number": 7,
  "kill_count": {
    "action": "add",
    "value": 3,
  }
}
```

**options**

**Optional** object Object that can specify additional *user identifiers* and callback functions.

Example:

```
{
  "user_id": user_id,
  "device_id": device_id,
  "email": email,
  "onFulfilled": onFulfilled,
  "onRejected": onRejected
}
```

**user\_id**

**Optional** string If the *application* lets *user* sign in - it is possible to pass a unique permanent *user ID* using this parameter. This will let the Audience Manager better identify *users* across devices (laptop, phone) and sessions.

Example:

```
"jbond"
```

**device\_id**

**Optional** string If the *application* has access to *device ID* - it is possible to pass this value using this parameter. This will let the Audience Manager better identify *users* across sessions.

Example:

```
"1234567890ABCDEF"
```

#### **email**

**Optional** *string* If the *application* identifies *user* via his email - it is possible to pass this value using this parameter. This will let the Audience Manager better identify *users* across devices (laptop, phone) and sessions.

Example:

```
"j.bond@mi6.gov.uk"
```

#### **onFulfilled()**

**Optional** The fulfilment handler callback (called with result).

#### **onRejected(*error\_code*)**

**Optional** The rejection handler callback (called with error code).

#### **Arguments**

- **error\_code** (*string*) – **Required** Error code.

Example:

```
"server_error"
```

## 3.2 Form Tracker

Form Tracker gathers data submitted via forms on your page and sends it to the Audience Manager *user* profile as *attributes*.

---

**Note:** Creates or updates *user custom attributes* for each tracker field in the form. The *attribute* name is generated from input tag (HTML tag's name attribute or description from its label). Inputs without a name are ignored.

---

### 3.2.1 Supported browsers

All modern browsers: Chrome, Firefox, Safari, Edge. Internet Explorer from version 8 and above.

### 3.2.2 Privacy by design

PPAS follows “Privacy by design” approach to system engineering.

**Warning:** Form tracker is trying to send its requests using secure **HTTPS** protocol, but **legacy IE browsers** (version 8 and 9) don't have capability to send **CORS** requests using different protocol then the one used by origin page. That means that forms tracked on those browsers will use less secure **HTTP** protocol on pages served via **HTTP** protocol.

## Private information

Form Tracker is trying to automatically detect fields containing *user's* private information and ignores them regardless of the configuration. The following data is never sent to the Audience Manager:

- Value from input with `password` or `hidden` type.
- Credit card number (heuristic detection).
- Credit card validation code (heuristic detection).

---

**Note:** Heuristic detection makes best effort to automatically detect and ignore the aforementioned fields, but it does not guarantee success. Additionally, payment forms usually contain more fields with private information (e.g. address) so it is recommended to configure such forms using fields filter.

---

## Configuration

Changed in version 10.0: Loading snippet changed to allow multiple initializations. Tracker will now try to merge configuration of tracked forms as long as `options` will allow it (will be identical).

Changed in version 6.3: Tracked forms are configured using whitelist approach. All forms that should be tracked have to be added to the list, any unrecognized form will be ignored by the tracker. This approach changed from previous blacklist approach where forms had to be included on the list before tracker started ignoring them.

### 3.2.3 Installation

This section describes how to install the Form Tracker client code on your page.

#### Using Tag Manager

The Form Tracker tag template is the recommended way to install Form Tracker using PPAS stack.

#### Manual installation

Add the following snippet on your page to start using Form Tracker.

This code should be added near the top of the `<head>` tag and before any other script or CSS tags. Additionally the snippet has to be configured this way:

- String `XXX-XXX-XXX-XXX-XXX` should be replaced with *app ID* (e.g. `efcd98a5-335b-48b0-ab17-bf43f1c542be`).
- String `https://your-instance-name.piwik.pro//` should be replaced with your PPAS instance address (please note that it's used in 3 places in the snippet).

Changed in version 10.0.

```
<script>
  (function(a,d,g,h,b,c,e){a[b]=a[b]||{};a[b][c]=a[b][c]||{};if(!a[b][c][e])
  ↪{a[b][c][e]=function(){(a[b][c][e].q=a[b][c][e].q||[]).push(arguments)};var f=d.
  ↪createElement(g);d=d.getElementsByTagName(g)[0];f.async=1;f.src=h;d.parentNode.
  ↪insertBefore(f,d)}})
  (window,document,"script","https://your-instance-name.piwik.pro/audiences/static/
  ↪widget/audience-manager.form.min.js","ppms","am","form");
```

(continues on next page)

(continued from previous page)

```
ppms.am.form("create", "XXX-XXX-XXX-XXX-XXX", "your-instance-name.piwik.pro", ↵
↵forms_config, options);
</script>
```

New in version 6.3.

### forms\_config

**Required** Object<string, (boolean|{type: string, fields: Array<string>})>  
Configuration of tracked forms. Default configuration requires that all tracked forms are specified in this object as keys. Each key is another form ID.

Value of each key can be specified in 2 ways:

- `true` - All fields in form using this ID will be tracked (this behavior can be changed using *trackingType* option).
- Object - Specifies which fields will be included or excluded from the form.

#### type

**Required** "whitelist"|"blacklist" Defines type of form fields filter.

#### fields

**Required** Array<string> Lists field names used by the filter. Default configuration identifies fields by input name attribute, but *useLabels* option can change this behavior.

Example:

```
{
  "tracked_form": true,
  "form_with_whitelisted_fields": {
    type: "whitelist",
    fields: ["included_field_1", "included_field_2"],
  },
  "form_with_blacklisted_fields": {
    type: "blacklist",
    fields: ["excluded_field_1", "excluded_field_2"],
  },
}
```

New in version 6.3.

### options

**Optional** object Options that change behavior of the tracker.

#### useLabels

**Optional** boolean Defines how tracker identifies form fields. When enabled tracker tries to find label of form field and use its text as identifier. If input doesn't have a label, tracker falls back to default identifier (HTML name attribute of the field). Default value: `false`.

Example:

```
false
```

Deprecated since version 6.3.

### trackingType

**Optional** "whitelist"|"blacklist" Defines what is default strategy of form configuration. Default value: "whitelist".

- "whitelist" - All form IDs that are not set in `forms_config` are ignored by the tracker.



- "blacklist" - All form IDs that are set in `forms_config` and use `true` value are ignored by the tracker. Forms defining filtered fields are tracked according to specified fields filter. All other forms are tracked as a whole.

---

**Note:** This option is intended for backward compatibility and is planned to be removed in the future.

---

Example:

```
{
  useLabels: true,
}
```

This code initializes the Form Tracker interface in the following ways:

1. Creates a `<script>` tag that asynchronously loads Audience Manager Form Tracker library.
2. Initializes global `ppms.am.form` command queue that schedules commands to be run when Form Tracker library is loaded.
3. Schedules creation of Form Tracker instance (using `ppms.am.form` function).

### 3.3 Public HTTP API

### 3.4 Authorized HTTP API



## 4.1 JavaScript API

### 4.1.1 Introduction

Consent Manager provides JavaScript API that allows the user to:

- Get compliance types
- Get new compliance types
- Set initial compliance settings
- Set compliance settings
- Get compliance settings
- Send data subject request

JavaScript API is implemented by providing global JavaScript objects queue responsible for executing command:

`ppms.cm.api (command, ...args)`

New in version 6.2: Replaces `dataLayer.push ()`

#### Arguments

- **command** (*string*) – Command name.
- **args** – Command arguments. The number of arguments and their function depend on command.

**Returns** Commands are expected to be run asynchronously and return no value.

**Return type** undefined

`dataLayer.push ({event: command, ...args})`

Deprecated since version 6.2: This interface is only for backward compatibility. You can read more about this particular case below. We recommend using `ppms.cm.api ()` instead.

### Arguments

- **command** (*string*) – Command name.
- **args** – Command arguments. The number of arguments and their function depend on command.

**Returns** Commands are expected to be run asynchronously and return no value.

**Return type** undefined

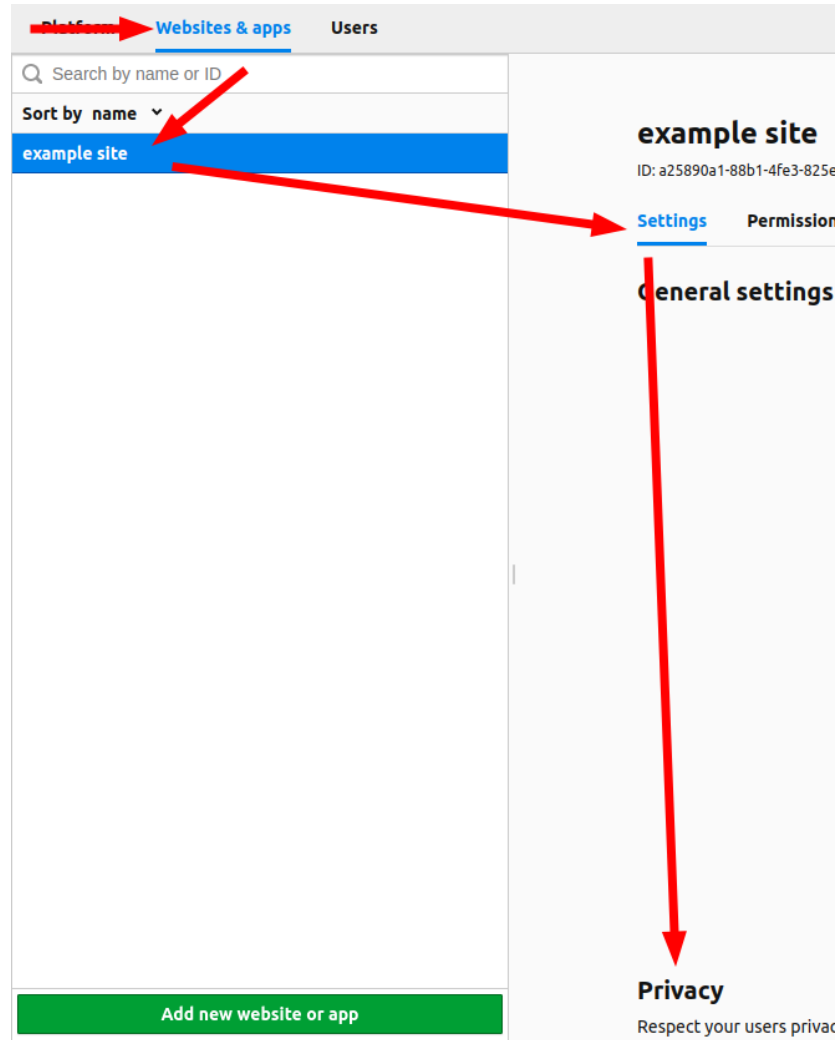
## 4.1.2 Custom consent form

API enables you to build custom consent form in place of default one. To turn off default consent form follow steps:

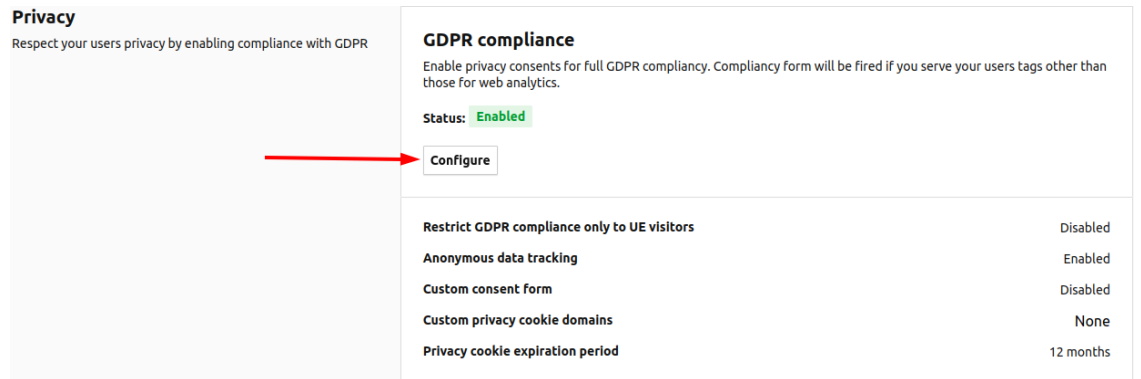
1. Login to your PPAS instance

The screenshot shows the Piwik PRO interface. At the top right, the text "PIWIK PRO Consent Manager" is displayed. Below this, a "Menu" button is visible. The menu is open, showing a list of navigation items: "Analytics" (with a bar chart icon), "Analytics" (with a line graph icon and a blue "NEW" badge), "Tag Manager" (with a tag icon), "Audience Manager" (with a target icon), "Personalization" (with a person icon), and "Consent Manager" (with a shield icon). Below these items, a user profile section shows a green circle with the letter "J" and the text "Logged in as: john@doe.com". Underneath the user profile, the menu items "User panel", "Administration" (highlighted with a blue background), "Analytics settings", "Help center", and "Log out" are listed.

2. Go to Menu > Administration



3. Go to Websites & apps and find Privacy section



4. Click Configure

### GDPR compliance

Enable privacy consents for full GDPR compliancy. Compliancy form will be fired if you serve your users tags other than those for web analytics.

**Restrict GDPR compliance only to UE visitors**

**Anonymous data tracking**

**Custom consent form**

**Custom privacy cookie domains**

Press "Tab" key to confirm

Enter the list of domains. You can use wildcards, e.g. \*.example.com

**Privacy cookie expiration period** 12

Ask visitors for consent again after this time period

**i** Read more: [Turn on/off GDPR compliance on the website](#)

**OK**

Cancel

#### 5. Check Custom consent form check box

Here is basic flow description to achieve a result similar to default consent form:

- get compliance settings (if you want to know if user is visiting site for the first time)
- get new compliance types
- get checked compliance types (if you want to show already checked consents on your consent form)
- show custom consent form and set initial compliance settings (setting initial compliance settings is required for Consent Manager Insights)
- set compliance types

### 4.1.3 Installation

Consent Manager is fully integrated with Tag Manager. If you have already installed asynchronous snippet and you are using API only from Tag Manager tags, you are able use JavaScript API without any pitfalls.

The one thing that should be considered before using API is where you call commands. Lets assume that your goal is to perform API method outside Tag Manager tags like in example below:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>

  <!-- Start Piwik PRO Tag Manager code -->
  <script>
    // Tag Manager async code snippet
  </script>
  <!-- End Piwik PRO Tag Manager code -->

  <script>
    // API call outside Tag Manager injected manually
    ppms.cm.api(command, ...args);
  </script>

  <body>
    Rest content of document
  </body>
</html>
```

When you need to execute API in such manner, you should take care about Tag Manager snippet version. Because `ppms.cm.api()` global object is initialized in snippet and/or in Tag Manager container, if you are using old version of Tag Manager snippet (PPAS version < 6.2), `ppms.cm.api()` might be undefined until container will be loaded. If you want to use your own scripts outside Tag Manager, you need to update the snippet to use `ppms.cm.api()`, or use `dataLayer.push()` interface if replacing snippet is not possible:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>

  <!-- Start Piwik PRO Tag Manager code -->
  <script>
    // Tag Manager async code snippet
  </script>
  <!-- End Piwik PRO Tag Manager code -->

  <script>
    // API call outside Tag Manager injected manually
    dataLayer.push({event: command, ...args});
  </script>

  <body>
    Rest content of document
  </body>
</html>
```



## 4.1.4 Commands

All commands work in context of the current visitor and website. Additionally they sometimes require communication with a PPAS server and are asynchronous. Callback functions are used to provide response value or information about errors. `onSuccess(...args)` callback is always required. `onFailure(exception)` callback is optional and if is specified, any error object occurred will be passed as a argument. If not specified, error is reported directly on console output.

### Get compliance types

Fetches a list of consent types.

Code:

```
ppms.cm.api('getComplianceTypes', onFulfilled, onRejected);
dataLayer.push({'event': 'ppms.cm:getComplianceTypes', parameters: [onFulfilled,
↪onRejected]});
```

`dataLayer.push()` interface is only for backward compatibility and you can read more about this particular case below. We recommend `ppms.cm.api()`.

**onFulfilled** (*types*)

**required** The fulfilment handler callback (called with result).

#### Arguments

- **types** (*Array<string>*) – **Required** Array of consent types

Example:

```
["remarketing", "analytics"]
```

**onRejected** (*error*)

The rejection handler callback (called with error code). If not specified, exception will be thrown in main stacktrace.

#### Arguments

- **error** (*string/object*) – **Required** Error code or exception.

### Get new compliance types

Fetches a list of new consent types which were appearing after given consents.

Code:

```
ppms.cm.api('getNewComplianceTypes', onFulfilled, onRejected);
dataLayer.push({'event': 'ppms.cm:getNewComplianceTypes', parameters: [onFulfilled,
↪onRejected]});
```

`dataLayer.push()` interface is only for backward compatibility and you can read more about this particular case below. We recommend `ppms.cm.api()`.

**onFulfilled** (*types*)

**required** The fulfilment handler callback (called with result).

#### Arguments

- **types** (*Array<string>*) – **Required** Array of consent types

Example:

```
["remarketing", "analytics"]
```

**onRejected** (*error*)

The rejection handler callback (called with error code).

#### Arguments

- **error** (*string/object*) – **Required** Error code or exception.

## Set initial compliance settings

Sets initial compliance settings. This API command might be useful to note that user has seen a popup with consents but didn't make a decision (popup was closed). After successful, Consent Manager internally sends only to stats endpoint an information that user has seen consents. Result from `getNewComplianceTypes` method can be passed directly.

Code:

```
ppms.cm.api('setInitialComplianceSettings', settings, onFulfilled, onRejected);
dataLayer.push({'event': 'ppms.cm:setInitialComplianceSettings', parameters:
↳ [settings, onFulfilled, onRejected]});
```

`dataLayer.push()` interface is only for backward compatibility and you can read more about this particular case below. We recommend `ppms.cm.api()`.

### settings

**required** The consent settings object.

Example:

```
{consents: ['analytics']}
```

or

Example:

```
['analytics']
```

**onFulfilled** ()

**required** The fulfilment handler callback. This function is **required**.

**onRejected** (*error*)

The rejection handler callback (called with error code). If not specified, exception will be thrown in main stacktrace.

#### Arguments

- **error** (*string/object*) – **Required** Error code or exception.

## Set compliance settings

Set compliance settings base on user decision. This API command might be useful when user interact with custom, extended UI that reacts on user approve/reject action. After successful, Consent Manager internally send consent settings to tracking server and force page view on tags. Additionally information to statistics is sent about user decisions.

Code:

```
ppms.cm.api('setComplianceSettings', settings, onFulfilled, onRejected);
dataLayer.push({'event': 'ppms.cm:setComplianceSettings', parameters: [settings,
↪onFulfilled, onRejected]});
```

`dataLayer.push()` interface is only for backward compatibility and you can read more about this particular case below. We recommend `ppms.cm.api()`.

### settings

**required** The consent settings object.

Example:

```
{consents: {analytics: {status: 1}}}
```

Where `consent.analytics` is consent type and status indicate:

- 0 - user has rejected the consent
- 1 - user has approved the consent

### onFulfilled()

**required** The fulfilment handler callback. This function is **required**.

### onRejected(error)

The rejection handler callback (called with error code). If not specified, exception will be thrown in main stacktrace.

### Arguments

- **error** (*string/object*) – **Required** Error code or exception.

## Get compliance settings

Return current privacy settings. Might be useful for initializing custom decision view. When there is no decisions, just returns empty object. This state can be used to detect first time user interaction with consent mechanism.

Code:

```
ppms.cm.api('getComplianceSettings', onFulfilled, onRejected);
dataLayer.push({'event': 'ppms.cm:getComplianceSettings', parameters: [onFulfilled,
↪onRejected]});
```

`dataLayer.push()` interface is only for backward compatibility and you can read more about this particular case below. We recommend `ppms.cm.api()`.

### settings

**required** The consent settings object.

Example:

```
{consents: {analytics: {status: -1, updatedAt: '2018-07-
↪03T12:18:19.957Z'}}}
```

Where `consent.analytics` is consent type and status indicate:

- -1 - user has not interacted, e.g. has closed a consent popup without any decision
- 0 - user reject consent

- 1 - user approve consent

**onFulfilled** (*settings*)

**required** The fulfilment handler callback (called with result).

**onRejected** (*error*)

The rejection handler callback (called with error code). If not specified, exception will be thrown in main stacktrace.

#### Arguments

- **error** (*string/object*) – **Required** Error code or exception.

## Send data subject request

Command send data subject request to Consent Manager collector.

Code:

```
ppms.cm.api('sendDataRequest', request, onFulfilled, onRejected);
dataLayer.push({'event': 'ppms.cm:sendDataRequest', parameters: [request, onFulfilled,
↪ onRejected]});
```

*dataLayer.push()* interface is only for backward compatibility and you can read more about this particular case below. We recommend *ppms.cm.api()*.

**request**

**required** The subject data request.

Example:

```
{content: 'user input', email: 'example@example.org', type: 'delete_data'}
```

Where *type* is request type, and can be one of:

- *change\_data* for data alteration request
- *view\_data* for view data request
- *delete\_data* for delete data request

**onFulfilled** ()

**required** The fulfilment handler callback.

**onRejected** (*error*)

The rejection handler callback (called with error code). If not specified, exception will be thrown in main stacktrace.

#### Arguments

- **error** (*string/object*) – **Required** Error code or exception.

## 4.1.5 Example usage

### Example implementation using jQuery

Based on above listed commands there are many possibilities to implement custom consent gathering. Below is listed a simple implementation using jQuery library.

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" href="data:;base64,iVBORw0KGgo=">
  <title>Piwik Pro Tag Manager Custom Consent Implementation</title>

  <!-- Start Piwik PRO Tag Manager custom consent css code -->
  <link rel="stylesheet" href="https://rawgit.com/djanix/jquery.switcher/master/dist/
↪switcher.css"/>
  <style>
    * {
      font-family: BlinkMacSystemFont, -apple-system, Roboto, Oxygen-Sans, Ubuntu, ↪
↪Cantarell, "Helvetica Neue", sans-serif;
    }

    .consent-container {
      background: white;
      display: none;
      bottom: 0;
      position: fixed;
      width: 100%;
      border-top: 1px solid #e0e0e0;
      z-index: 10000;
      color: rgba(0, 0, 0, 0.7);
      box-sizing: border-box;
    }

    .consent-content {
      display: inline-flex;
      width: 100%;
    }

    .consent-left {
      flex: 1 0 0;
      flex-direction: column;
      border-right: 1px solid #e0e0e0;
      font-size: 14px;
      display: flex;
      justify-content: center;
      align-items: center;
      text-align: center;
      padding: 0 30px;
    }

    .consent-right {
      flex: 3 0 0;
      box-sizing: border-box;
      display: flex;
      opacity: 0.9;
      padding: 30px;
      background-color: #f5f5f5;
      line-height: 16px;
    }

    .consent-opt-in-button {

```

(continues on next page)

(continued from previous page)

```
border-color: rgba(197, 103, 57, 1);
background-color: rgba(252, 131, 72, 1);
color: #fff;
min-width: 120px;
font-weight: 600;
font-size: 16px;
line-height: 17px;
text-align: center;
border: 1px solid;
border-radius: 2px;
outline: 0;
cursor: pointer;
padding: 9px 16px 9px 16px;
}

.consent-link-more {
color: #107EF1;
font-size: 14px;
line-height: 16px;
margin-top: 7px;
text-decoration: none;
display: inline-block;
}

.consent-bottom {
max-height: 0;
transition: max-height 0.5s;
}

.consent-items {
box-sizing: border-box;
position: relative;
}

.consent-items-container {
display: flex;
flex-direction: column;
}

.consent-items-text {
margin-left: 10px;
}

.consent-item {
display: flex;
height: 50px;
}

.consent-item-left {
width: 25%;
border-right: 1px solid #e0e0e0;
box-sizing: border-box;
display: flex;
justify-content: space-between;
}
```

(continues on next page)

(continued from previous page)

```
.consent-item-right {
  width: 75%;
  display: flex;
  align-items: center;
}

.consent-item-right-text {
  font-size: 14px;
  margin: 0 30px;
}

.consent-items-description {
  padding: 20px 0;
  max-height: 54px;
  display: inline-flex;
  width: 100%;
  border-top: 1px solid #e0e0e0;
  border-bottom: 1px solid #e0e0e0;
}

.consent-items-footer {
  padding: 20px 0;
  max-height: 54px;
  width: 100%;
  border-top: 1px solid #e0e0e0;
}

label {
  width: 100%;
  padding: 0 30px;
  box-sizing: border-box;
  font-weight: 500;
  line-height: 55px;
  cursor: pointer;
  margin: 0;
}

.consent-switcher {
  margin: 10px 10px 10px 0;
}

.consent-blue {
  background: #107EF1;
  border: 1px solid #107EF1;
}
</style>
<!-- End Piwik PRO Tag Manager custom consent css code -->
</head>
<body>

<!-- PUT HERE CONTAINER JS CODE -->

<!-- Start Piwik PRO Tag Manager custom consent javascript code -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.0.0/jquery.min.js"></
<script>
<script src="https://rawgit.com/djanix/jquery.switcher/master/dist/switcher.js"></
<script>
```

(continues on next page)

(continued from previous page)

```
<div style="display: none; visibility: hidden;" data-template="consentitem">
  <div class="consent-item">
    <div class="consent-item-left">
      <div>
        <label>${name}</label>
      </div>
      <div class="consent-switcher">
        <input class="consent-checkbox" type="checkbox" name="consentValues" value="
↪${key}" />
      </div>
    </div>
    <div class="consent-item-right">
      <div class="consent-item-right-text">
        ${description}
      </div>
    </div>
  </div>
</div>

<script>
  var availableConsents = [
    {
      key: 'analytics',
      name: 'Analytics',
      description: 'We will store data in an aggregated form about visitors and their
↪experiences on our website. We use this data to fix bugs and improve the experience
↪for all visitors.'
    },
    {
      key: 'ab_testing_and_personalization',
      name: 'AB Testing',
      description: 'We will create a cookie in your browser to ensure consistency of
↪our A/B tests. A/B tests are small changes displayed to different groups of
↪visitors. We use the data to create a better experience for all visitors. We will
↪also use this cookie to personalize content for you.'
    },
    {
      key: 'conversion_tracking',
      name: 'Conversion Tracking',
      description: 'We will store data about when you complete certain actions on our
↪website to understand better how you use it. We use this data to improve your
↪experience with our site.'
    },
    {
      key: 'marketing_automation',
      name: 'Marketing Automation',
      description: 'We will store data to create marketing campaigns for certain
↪groups of visitors.'
    },
    {
      key: 'remarketing',
      name: 'Remarketing',
      description: 'We will store data to show you our advertisements (only ours) on
↪other websites relevant to your interests.'
    },
    {
```

(continues on next page)



(continued from previous page)

```

    key: 'user_feedback',
    name: 'User Feedback',
    description: 'We will store data in an aggregated form to analyze the
↳performance of our website\'s user interface. We use this data to improve the site
↳for all visitors.'
  },
  {
    key: 'custom_consent',
    name: 'Custom consent',
    description: 'Adjust this copy to your needs.'
  }
];

var customConsentSolution = {
  isDetailsOpen: false,
  containerSelector: '#consent-container',
  consentBottomSelector: '#consent-bottom',
  consentLinkMoreSelector: '#consent-link-more',
  consentItemFooterSelector: '#consent-items-footer',
  switcherSelector: '.consent-items-container .consent-checkbox',
  optInButton: '.consent-orange',
  sendConsentButtonSelector: '.consent-blue',
  itemsSelector: '.consent-items-container',
  consentTemplate: $('div[data-template="consentitem"]').html().split(/\${(.+?)\}/
↳g),
  switcherElement: null,

  init: function() {
    $(this.consentLinkMoreSelector).click(this.showDetails.bind(this));
    $(this.sendConsentButtonSelector).click(this.sendConsents.bind(this, false));
    $(this.optInButton).click(this.sendConsents.bind(this, true));
    this.loadConsentList();
  },

  show: function() {
    $(self.containerSelector).slideDown(100);
  },

  hide: function() {
    $(self.containerSelector).slideUp(100);
  },

  loadConsentList: function() {
    function success(types) {
      self = customConsentSolution;

      if (types.length > 0) {
        self.show();
      }

      $(self.itemsSelector).append(
        availableConsents
          .filter(function (element) {
            return types.join(',').indexOf(element.key) !== -1;
          })
          .map(function (item) {
            return self.consentTemplate.map(self.replaceTemplate(item)).join('');
          }

```

(continues on next page)

(continued from previous page)

```

    })
    ).ready(function () {
        self.switcherElement = $(self.switcherSelector).switcher();
    });
}

ppms.cm.api('getNewComplianceTypes', function(types) {
    if (types.length) {
        success(types);
        return;
    }

    ppms.cm.api('getComplianceSettings', function (result) {
        var types = [];

        Object.keys(result.consents).forEach(function (key) {
            if (result.consents[key].status === -1) {
                types.push(key);
            }
        });

        success(types);
    }, function(e) {});
}, function(e) {});
},

setInitialConsents: function() {
    ppms.cm.api('getNewComplianceTypes', function(types) {
        ppms.cm.api('setInitialComplianceSettings', types, function() {}, function(e)
→{});
    }, function(e) {});
},

sendConsents: function(all) {
    var queryElements = {
        consents: {}
    };

    $.each(this.switcherElement, function(index, value) {
        queryElements.consents[$(value).val()] = {
            status: all ? 1 : +$(value).prop('checked')
        };
    });

    ppms.cm.api('setComplianceSettings', queryElements, function() {
        self = customConsentSolution;
        self.hide();
    }, function (e) {});
},

showDetails: function() {
    var detailsScrollHeight = $(this.consentBottomSelector).prop('scrollHeight');
    var baseScrollHeight = $(this.containerSelector).prop('scrollHeight');
    var consentItemFooterHeight = $(this.consentItemFooterSelector).prop(
→'scrollHeight');

    $(this.consentBottomSelector).css({

```

(continues on next page)

(continued from previous page)

```

    maxHeight: baseScrollHeight + detailsScrollHeight + consentItemFooterHeight +
↪ "px",
    overflowY: 'auto',
    display: 'block'
  });

  this.setInitialConsents();
},

replaceTemplate: function(props) {
  return function(token, i) { return (i % 2) ? props[token] : token; };
}
};

$(document).ready(customConsentSolution.init.bind(customConsentSolution));
</script>
<!-- End Piwik PRO Tag Manager custom consent javascript code -->

<!-- Start Piwik PRO Tag Manager custom consent html code -->
<div class="consent-container" id="consent-container">
  <div class="consent-content">
    <div class="consent-left">
      <button class="consent-opt-in-button consent-orange">Opt-in and let's go!</
↪button>
    </div>
    <div class="consent-right">
      <div>
        <h1>[IMPORTANT] You're invited!...</h1>
        <a class="consent-link-more" id="consent-link-more" href="#">Show more</a>
      </div>
    </div>
  </div>
  <div class="consent-bottom" id="consent-bottom">
    <div class="consent-items">
      <div class="consent-items-description">
        <div class="consent-items-text">
          ...to tell us how you want us to handle your data.
          We'll only use your data for purposes you consent to.
          Change your mind whenever, we'll adapt to your consent preferences and
↪data requests.
        </div>
      </div>
      <div class="consent-items-container"></div>
      <div class="consent-items-footer" id="consent-items-footer">
        <div class="consent-items-text">
          <button class="consent-opt-in-button consent-blue">Save choices</button>
        </div>
      </div>
    </div>
  </div>
</div>
<!-- End Piwik PRO Tag Manager custom consent code -->

</body>
</html>

```

### Example implementation in ConsentManager-CustomForm repository

You can visit [ConsentManager-CustomForm](#) repository on Github, for another example of Consent Manager JavaScript API usage. You will find there an out-of-the-box solution, based on the one deployed on the [piwik.pro](#) website, along with instructions on how to use this particular example on your website.

New in version 10.0.

## 5.1 Custom data layer name

### 5.1.1 Introduction

The data layer is a global JavaScript object, that can be used to pass information from the website to PPAS container. The default value for the data layer name is *dataLayer*, but the value can be customized. The purpose of this article is to describe the steps that need to be performed to set up custom data layer name.

### 5.1.2 Renaming the data layer

If you wish to rename your data layer, you should follow the instructions below:

1. Log into your PPAS instance.
2. Head to *Menu > Administration*.
3. Select the website that you want to set the new data layer name for and then go to the *Installation* page.
4. From here you should copy the code of the snippet, that you want to use - asynchronous or synchronous - and change the *dataLayer* value highlighted on the screens below:
  - a. for the asynchronous snippet

#### Container code for asynchronous tags

Paste the container code for asynchronous tags into your website's code (inside the <body> element on all your pages). We recommend you embed the code right after the opening <body> tag.

```

expires="+e.toUTCString()+document.cookie="a"+"b+d"; path="/"
5 var isStgDebug=
(window.location.href.match("stg_debug")||document.cookie.match("stg_debug"))&&!window.location.href.match("stg_disable_debug");stgCreateCookie("stg_d
ebug",isStgDebug?1:"","isStgDebug?14:-1);
6 var qP=[];dataLayerName!="dataLayer"&&qP.push("data_layer_name="+dataLayerName),isStgDebug&&qP.push("stg_debug");var qPString=qP.length>0?
("?"*qP.join("&")):"";
7 tags.async=!0,tags.src="//client1.piwikpro.test/containers/"+id+".js"+qPString,scripts.parentNode.insertBefore(tags,scripts);
8 !function(a,n,i){a[n]=a[n]||{};for(var c=0;c<i.length;c++)!function(i){a[n][i]=a[n][i]||{},a[n][i].api=a[n][i].api||function(){var a=
[]}.slice.call(arguments,0);"string"==typeof a[0]&&window[dataLayerName].push({event:n+"."+i+": "+a[0],parameters:[]}.slice.call(arguments,1))};(i[c])
(window,"ppas",["ta","cm"]);
9 }) (window, document, 'dataLayer', '553ad63d-0f52-40c3-9dbf-247689159e9b');
10 </script><noscript><iFrame src="//client1.piwikpro.test/containers/553ad63d-0f52-40c3-9dbf-247689159e9b/noscript.html" height="0" width="0"
style="display:none;visibility:hidden"></iFrame></noscript>

```

## b. for the synchronous snippet

**Container code for synchronous tags**

Install synchronous JavaScript code for advanced features, like A/B testing. Paste the container code for synchronous tags into your website's code (in between the <head></head> element of all your pages). Be aware that embedding this code in between the <body> element may cause performance problems!

```

1 <script type="text/javascript">
2 (function(window, document, dataLayerName, id) {
3   function stgCreateCookie(a,b,c){var d="";if(c){var e=new Date;e.setTime(e.getTime()+24*c*60*60*1e3),d="";
4     expires="e.toUTCString()";document.cookie=a+"="+b+d+"; path=/"
5   }
6   var isStgDebug=
7     (window.location.href.match("stg_debug")||document.cookie.match("stg_debug"))&&!window.location.href.match("stg_disable_debug");stgCreateCookie("stg_deb
8     ug",isStgDebug?1:"");
9   var qp={};dataLayerName!="dataLayer"&&qp.push("data_layer_name="+dataLayerName),isStgDebug&&qp.push("stg_debug");var qPString=qp.length?
10    ("?"*qp.join("&")):"";
11    document.write("<script src='//client1.piwikpro.test/containers/"+id+"/sync.js' + qPString + '></' + 'script'");
12  })(window, document, "dataLayer", "553ad63d-0f52-40c3-9dbf-247689159e9b");
13 </script>

```

- At this point, you can embed the snippet on your website. If you are replacing an existing snippet, make sure to take a look at the *Snippet migration* section for instructions on how to ensure full compatibility.

**Note:**

- The snippet code was changed compared to PPAS versions pre 10.0, to accommodate for the possibility of changing the data layer name. The new snippet, released alongside version 10.0 is backward compatible, however, to use the functionality described in this document, whole snippet code needs to be replaced on your website.
- If you are using both synchronous and asynchronous snippets, it is recommended to use the same data layer name in both. Nonadherence to this rule may cause unexpected and unwanted behavior.
- We strongly advise making sure that the chosen name is not used in any other external software present on your website as well as inside your own website's code. Data layer name should be unique and reserved only for use inside the PPAS container. For additional guidelines regarding the data layer naming process, please refer to *Data layer name guidelines*.

### 5.1.3 Snippet migration

Once you replace your original snippet and want to use the new data layer name, there is one more step that needs to be taken care of. If you are using direct data layer pushes in your code (e.g. to set a *Data layer* variable), you need to make sure to replace all references to `dataLayer` with the newly selected name, e.g (assuming the new name of `customDataLayer`):

```
dataLayer.push({event: "test-event"});
```

will become:

```
customDataLayer.push({event: "test-event"});
```

### 5.1.4 Data layer name guidelines

To avoid conflicts with your existing code, that could cause unwanted behavior, you need to make sure that the name selected for the data layer object is unique. To ensure that it is not already used by someone, we suggest you run the following command in the console on your website::

```
var dataLayerName = "customDataLayer";
!window.hasOwnProperty(dataLayerName);
```

where the value of the `dataLayerName` variable is your chosen name (`customDataLayer` in this case). If this operation returns *true* into the console, then you can safely use this value as the name of your data layer.

## 6.1 Authorized API guide

### 6.1.1 Introduction

This page describes how to access Piwik PRO API which uses `client credentials` OAuth grant type for obtaining user token. All data is sent and received as JSON and is compliant with `JSON API` specification.

#### Obtaining token

If you want to access API for the first time you need to generate your API credentials which then allows you to request for a token that is used for authentication during communication with authorized API.

#### Generate API Credentials

- Login to your account using your email and password.
- Go to your profile (Menu then `My profile`).
- On this page click on `API Credentials` tab. This page allows you to manage all your API credentials.
- Click `Generate new credentials` which will result in new popup. Fill in your custom `credentials` name. Name must contains at least 3 characters.
- Copy your newly generated `CLIENT ID` and `CLIENT SECRET` because they **won't be available for you after dismissing this window**.

Those credentials will be valid as long as you will not revoke them in your profile.

## Create access token

Having generated your API Credentials, now you are ready for creating access token that will be used in communication with API.

Piwik PRO API tokens use [JWT](#) format.

Make POST call to `https://<domain>/auth/token` with header `Content-Type: application/json` and payload: `{ "grant_type": "client_credentials", "client_id": "<client_id>", "client_secret": "<client_secret>" }`.

Response example:

```
{ "token_type": "Bearer", "expires_in": 1800, "access_token": "<your_access_token>" }
```

Now, you can use obtained `<your_access_token>` for communication with Piwik PRO API. Field `expires_in` stands for time (in seconds) for token expiration (TTL). Since token is a Bearer type, it must be **included in every API call** within header.

```
Authorization: Bearer <your_access_token>
```

## Deleting API Credentials

Once you want to revoke the possibility of generating API token using given `CLIENT ID` and `CLIENT SECRET`, go to `My profile` and click `Delete` button on selected API credentials.

### 6.1.2 API usage example

Whatever API call you choose, first remember that you must generate *API credentials* for obtaining client id and secret.

#### API usage example with curl

For sake of this examples, `https://<domain>` is a URL of your PPAS instance (e.g. `https://example.piwik.pro`) and our goal is to perform basic operations on an app. We will:

- create an app
- get created app
- update its attributes
- remove the app

#### Generate your access token

Request example:

```
POST /auth/token
```

```
curl -X POST 'https://<domain>/auth/token' -H "Content-Type: application/json" --data '{
  "grant_type": "client_credentials",
  "client_id": "your_generated_client_id",
```

(continues on next page)



(continued from previous page)

```
"client_secret": "your_generated_client_secret"
}'
```

Response example:

```
{
  "token_type": "Bearer",
  "expires_in": 1800,
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.
  ↳eyJJpc3MiOiJkcG1zIiwiaXVkiOiJoiaHR0cHM6XC9cL3Rlc3RpbmcucG13aWsucHJvXC9zZXR5LCJzdWIiOiJkNmNkZGMxMS1iZDA
  ↳Nec2mYFRv6manzXjq0sHQxINZvu-fbDYT8AedVHBKYvulF9hYKaFReY8rNgfsMANw2OX8-
  ↳IKpTrQb1DyRkG4nxpIEbob528_
  ↳lPd7roho5mtK1E8sfs9WZE1piYOwaNDySDEUwUowgj2xBiJqSODjxBI6qVhLkynGEEeNBVh-
  ↳lrUmlcjpYqUc3saHvX72L-rqbIHa_1dzGarR-dcPyns-RpKjZEILzUSYOHdM09KDti-xsG-
  ↳nbKHGdP8fVEEJPypnAfJPOLHQg_j1c5IvJSvTKVF3j4_
  ↳zo6Zw5g8YkaheT9Iwph5BGHFRneXatcmbwKI8JzSDFi6CinzI-okYKRPbg"
}
```

Field `access_token` contains your token which then will be used for all API calls. Once you generated an access token, you can use it during its lifetime (30 minutes by default)

## Create an app

Request example:

```
POST /api/apps/v2
```

```
curl -X POST 'https://<domain>/api/apps/v2' -H "Authorization: Bearer <your_access_
  ↳token>" -H "Content-Type: application/vnd.api+json" --data '{
  "data": {
    "attributes": {
      "timezone": "UTC",
      "name": "AppName",
      "urls": [
        "http://example.com"
      ],
      "currency": "USD"
    },
    "type": "ppms/app"
  }
}'
```

Note, that you have to replace:

- `<domain>` with your PPAS instance URL,
- `<your_access_token>` with your generated access token

Response example:

```
{
  "data": {
    "type": "ppms/app",
    "id": "b30e538d-4b05-4a75-ae25-7eb565901f38",
    "attributes": {
      "name": "AppName",

```

(continues on next page)

(continued from previous page)

```
"addedAt":"2018-09-13T12:16:30+00:00",
"urls":[
  "http://example.com"
],
"timezone":"UTC",
"currency":"USD",
"excludeUnknownUrls":false,
"keepUrlFragment":true,
"eCommerceTracking":false,
"siteSearchTracking":true,
"siteSearchQueryParams":[
  "q",
  "query",
  "s",
  "search",
  "searchword",
  "keyword"
],
"siteSearchCategoryParams":[

],
"delay":500,
"excludedIps":[

],
"excludedUrlParams":[

],
"excludedUserAgents":[

],
"gdpr":true,
"gdprUserModeEnabled":false,
"privacyCookieDomainsEnabled":false,
"privacyCookieExpirationPeriod":31536000,
"privacyCookieDomains":[

],
"organization":"default",
"appType":"web",
"gdprLocationRecognition":false
}
}
```

## Get an app

Now, when app is added, it is possible to get it.

Request example:

```
GET /api/apps/v2/<app_id>
```

```
curl 'https://<domain>/api/apps/v2/b30e538d-4b05-4a75-ae25-7eb565901f38' -H
↳ "Authorization: Bearer <your_access_token>" -H "Content-Type: application/vnd.
↳ api+json"
```

(continues on next page)

(continued from previous page)

Notice: URL contains b30e538d-4b05-4a75-ae25-7eb565901f38. What is it? It is unique ID of an app. If you want to update given resource you must specify which one. How to obtain this ID? You can obtain ID from response's 'data/id' field when you added an app

Response example:

```
{
  "data":{
    "type":"ppms/app",
    "id":"b30e538d-4b05-4a75-ae25-7eb565901f38",
    "attributes":{
      "name":"AppName",
      "addedAt":"2018-09-13T12:16:30+00:00",
      "urls":[
        "http://example.com"
      ],
      "timezone":"UTC",
      "currency":"USD",
      "excludeUnknownUrls":false,
      "keepUrlFragment":true,
      "eCommerceTracking":false,
      "siteSearchTracking":true,
      "siteSearchQueryParams":[
        "q",
        "query",
        "s",
        "search",
        "searchword",
        "keyword"
      ],
      "siteSearchCategoryParams":[

    ],
    "delay":500,
    "excludedIps":[

  ],
  "excludedUrlParams":[

],
"excludedUserAgents":[

],
"gdpr":true,
"gdprUserModeEnabled":false,
"privacyCookieDomainsEnabled":false,
"privacyCookieExpirationPeriod":31536000,
"privacyCookieDomains":[

],
"organization":"default",
"appType":"web",
"gdprLocationRecognition":false
}
}
}
```

### Update app

Consider you added app, but afterwards you want to change its name.

Request example:

```
PATCH /api/apps/v2/<app_id>
```

```
curl -X PATCH 'https://<domain>/api/apps/v2/b30e538d-4b05-4a75-ae25-7eb565901f38' -H
↳"Authorization: Bearer <your_access_token>" -H "Content-Type: application/vnd.
↳api+json" -v --data '{
  "data": {
    "attributes": {
      "name": "NewAppName"
    },
    "type": "ppms/app",
    "id": "b30e538d-4b05-4a75-ae25-7eb565901f38"
  }
}'
```

This request changed app name from AppName to NewAppName.

Notice three things:

- -X PATCH before URL. It means that this request is available using HTTP PATCH method
- you have to specify also data/id - it's a [JSON API](#) requirement
- also data/type is required. For example, when you want to work with app resource, specify it's type as ppms/app
- you can set only parameters you want to update. For more apps attributes go to [App edit reference](#)

API will return 204 No Content status code with an empty response.

### Delete an app

Sometimes resources are not needed anymore, so let's have a look at example on how to delete them.

Request example:

```
DELETE /api/apps/v2/<app_id>
```

```
curl -X DELETE 'https://<domain>/api/apps/v2/b30e538d-4b05-4a75-ae25-7eb565901f38' -H
↳"Authorization: Bearer <your_access_token>" -H "Content-Type: application/vnd.
↳api+json"
```

There is no response example. API will return 204 No Content status code.

### API usage example with Postman

[Postman](#) is a multiplatform GUI application for creating API calls. PPAS allows you to export swagger documentation and easily import it to Postman. Depending of what you want to work with, you can import given swagger docs:

- Access control
- Apps
- Meta Sites

- Users
- User Groups

Simply click in Postman: `import -> Import From Link`. Then all of your paths are imported! You have to override two things:

- replace your domain in url
- add token. Click on `Authorization` tab on chosen API call and then use `Bearer Token` type. Paste your token and now you can call API using `SEND` button.

### 6.1.3 FAQ

Here you can find the most common issues encountered during work with the API

**API returns "application/json" is not a valid JSON API Content-Type header, use "application/vnd.api+json" instead**

Remember, all API calls needs to be created with `Content-Type: application/vnd.api+json` header. If you use `curl` you need to use `-H "Content-Type: application/vnd.api+json"` flag. Postman allows configuring headers with `Header` tab.

**API returns `JWT not found`**

Remember, you need to always use your API token. You need to send it all the time within `Authorization: Bearer <your_access_token>` header. If you use `curl` you need to use `-H "Authorization: Bearer <your_access_token>"` flag. Postman allows configuring tokens in `authorization` tab. Choose type `Bearer Token` and paste it there. Remember to keep this token secure as it allows access to sensitive data!

**API returns `Expired JWT Token`**

Every token that you generated is specified by `TTL` - time to live. By default it's 30 minutes. After token is expired, you need to *generate your access token*

**API returns `access token not authorized`**

This message means, that you sent access token within proper `Authorization: Bearer` field, although it is invalid. Make sure you set proper token.

## **6.2 Access Control API**

## **6.3 Apps API**

## **6.4 Meta Sites API**

## **6.5 Users API**

## **6.6 User Groups API**

## 7.1 Piwik PRO SDK for Android

### 7.1.1 SDK configuration

#### Server

- You need a Piwik PRO account on the cloud or an on-premises setup which your mobile app will communicate with. For details, please visit the [Piwik PRO website](#).
- Create a new website (or app) in the Piwik PRO web interface.
- Copy and note the Website ID from “Settings > Websites” and your server address.

#### Client

##### Including the library

Add dependencies to your app module `build.gradle` file (e.g. `~/git/MyApplication/app/build.gradle`):

```
dependencies {
    repositories {
        jcenter()
    }
    compile 'pro.piwik.sdk:piwik-sdk:VERSION'
}
```

Replace `VERSION` with the latest release name, e.g. `1.0.0`.

## Configuration

In order to set up the Piwik PRO tracker, you have two options:

1. Extend `PiwikApplication` class with your `Android Application` class. It forces implementation of one abstract method. That approach is used in the [Piwik PRO SDK demo app](#) as below:

```
public class YourApplication extends PiwikApplication{
    @Override
    public TrackerConfig onCreateTrackerConfig() {
        return TrackerConfig.createDefault("https://your.piwik.pro.server.com",
↪ "01234567-89ab-cdef-0123-456789abcdef");
    }
}
```

2. Manage the `Tracker` on your own. To configure the `Tracker` you will need a server address and website ID (you can find it in “Settings > Websites”):

```
public class YourApplication extends Application {
    private Tracker tracker;
    public synchronized Tracker getTracker() {
        if (tracker == null) tracker = Piwik.getInstance(this).newTracker(new
↪ TrackerConfig("https://your.piwik.pro.server.com", "01234567-89ab-cdef-0123-
↪ 456789abcdef"));
        return tracker;
    }
}
```

It is not recommended to create multiple `Tracker` instances for the same target as it may lead to over-count of metrics. It is highly recommended to create and manage the tracker in the `Application` class (to make sure there is only one instance of the tracker). The `Tracker` is thread-safe and can be shared across the application.

```
Tracker tracker = ((PiwikApplication) getApplication()).getTracker();
```

The application is ready to use Piwik PRO SDK.

### 7.1.2 Using Piwik PRO SDK

It is recommended to use `TrackerHelper` class. It has methods for all common actions, which can be chained in a way that facilitates the correct order and use. Combine it with IDE autocompletion and using the SDK will be more convenient.

For tracking each event with `TrackHelper`, you will need to pass `Tracker` instance. The way of getting the correct `Tracker` instance depends on the configuration option (see section above):

1. Your `Android Application` class extend `PiwikApplication` class

```
Tracker tracker = ((PiwikApplication) getApplication()).getTracker();
```

2. You manage the `Tracker` yourself

```
Tracker tracker = ((YourApplication) getApplication()).getTracker();
```

In further examples we will assume usage of the first option.



## Tracking screen views

### *Requires Analytics*

During a valid tracking session, you can track screen views which represent the content the user is viewing in the application. To send a visit on the screen, set the screen path and title on the tracker. This path is internally translated by the SDK to an HTTP URL as the Piwik PRO server uses URLs for tracking views. Additionally, Piwik PRO SDK uses prefixes which are inserted in a generated URL for various types of action(s). For tracking screen views it will use a prefix *screen* by default, however, automatic prefixing can be disabled with the `tracker.setPrefixing(false)` option.

```
public class YourActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Tracker tracker = ((PiwikApplication) getApplication()).getTracker();
        TrackHelper.track().screen("your_activity_path").title("Title").with(tracker);
    }
}
```

- A path (required) – each screen should be mapped to the URL path
- A title (optional) – the title of the action being tracked. It is possible to use slashes (/) to set one or several categories for this action.

To automatically use the activity-stack as a path and activity title as a name, use the overloaded screen method:

```
public class YourActivity extends Activity {
    ...
    TrackHelper.track().screen(YourActivity).with(tracker);
    ...
}
```

- An activity (required) – current instance of android `Activity` class.

In order to bind the tracker to your applications, use the `screens` method. This method will automatically track all open application activities (views) keeping the activity-stack as a path and activity title as the name:

```
TrackHelper.track().screens(getApplication()).with(tracker);
```

## Tracking custom events

### *Requires Analytics*

To collect data about the user's interaction with the interactive components of the application, like a button presses or the use of a particular item in the game - use event method.

```
TrackHelper.track().event("category", "action").path("/main/actionScreen").name("label  
↔").value(1000f).with(tracker);
```

The `track` method allows the specification of the following parameters:

- A category (required) – this String defines the event category. You may define event categories based on the class of user actions (e.g. clicks, gestures, voice commands), or you may define them based on the features available in your application (e.g. play, pause, fast forward, etc.).
- An action (required) – this String defines the specific event action within the category specified. In the example, we are effectively saying that the category of the event is user clicks, and the action is a button click.

- A name (optional) – this String defines a label associated with the event. For example, if you have multiple button controls on a screen, you may use the label to specify the specific view control identifier that was clicked.
- A value (optional) – this Float defines a numerical value associated with the event. For example, if you were tracking “Buy” button clicks, you may log the number of items being purchased or their total cost.
- A path (optional) – the path under which this event occurred.

For more resources, please visit:

- [Custom Events Overview](#)
- [Ultimate guide to event tracking.](#)

### Tracking exceptions

*Requires Analytics*

Caught exceptions are errors in your app for which you’ve defined an exception handling code, such as the occasional timeout of a network connection during a request for data. Exceptions are tracked on the server in a similar way as screen views, however, action internally generated for exceptions always use the *fatal* or *caught* prefix, and additionally the *exception* prefix if `tracker.isPrefixing()` this particular option is enabled(true). The URL corresponds to exception stack trace, including the package name, activity path, method name and line number where crash occurred. Bear in mind that Piwik is not a crash tracker therefore use this sparingly.

Measure a caught exception by setting the exception field values on the tracker and sending the hit, as with this example:

```
try {
    // perform action
} catch (Exception ex) {
    TrackHelper.track().exception(ex).description("Content download error").
    ↪fatal(true).with(tracker);
}
```

- An exception (required) – Caught exception instance.
- A description (optional) – additional information about the issue.
- An isFatal (optional) – true if an exception is fatal.

### Tracking social interactions

*Requires Analytics*

Social interactions such as likes, shares and comments in various social networks can be tracked as below. This, again, is tracked in a similar way as with screen views but the *social* prefix is used when the default `tracker.isPrefixing()` option is enabled.

```
TrackHelper.track().socialInteraction("Like", "Facebook").target("Game").
↪with(tracker);
```

- An interaction (required) – defines the social interaction, e.g. “Like”.
- A network (required) – defines social network associated with interaction, e.g. “Facebook”
- A target (optional) – the target for which this interaction occurred, e.g. “My Piwik PRO app”.

The URL corresponds to String, which includes the network, interaction and target parameters separated by slash.

## Tracking downloads and app installs

### Requires Analytics

You can track the installations and downloads initiated by your application. This only triggers an event once per app version unless you force it. It is recommended to track application install in the Android Application class:

```
TrackHelper.track().download().identifier(new DownloadTracker.Extra.  
↳ApkChecksum(this)).with(getTracker());
```

That will use the package name, version and MD5 app checksum as an identifier, e.g. `com.piwikpro.demo:12/7B3DF8ED277BABEA6126C44E9AECEFEA`.

In case you need to specify more parameters, create the instance of the `DownloadTracker` class explicitly:

```
DownloadTracker downloadTracker = new DownloadTracker(getTracker());  
DownloadTracker.Extra extra = new DownloadTracker.Extra.Custom() {  
    @Override  
    public boolean isIntensiveWork() {  
        return false;  
    }  
  
    @Nullable  
    @Override  
    public String buildExtraIdentifier() {  
        return "Demo Android download";  
    }  
};  
  
TrackHelper.track().download(downloadTracker).identifier(extra).force().  
↳version("1.0").with(getTracker());
```

- `isIntensiveWork()` - return true if this should be run async and on a separate thread.
- `buildExtraIdentifier()` - return a String that will be used as extra identifier or null.

On the analytics panel, all downloads can be viewed in the corresponding section.

## Tracking outlinks

### Requires Analytics

For tracking outlinks to external websites or other apps opened from your application use the `outlink` method:

```
TrackHelper.track().outlink(new URL("https://www.google.com")).with(getTracker());
```

- A URL (required) – defines the outlink target. HTTPS, HTTP and FTP are valid.

## Tracking search operations

### Requires Analytics

Tracking search operations allow the measurement of popular keywords used for various search operations performed inside your application. It can be done via the `search` method:

```
TrackHelper.track().search("Space").category("Movies").count(3).with(getTracker());
```

- A keyword (required) – the searched query that was used in the app.

- A category (optional) – specify a search category.
- A count (optional) – we recommend setting the search count to the number of search results displayed on the results page. When keywords are tracked with a count of 0, they will appear in the “No Result Search Keyword” report.

### Tracking content impressions and interactions

#### *Requires Analytics*

You can track an impression of an ad in your application as below.

```
TrackHelper.track().impression("Android content impression").piece("banner").target(
↳ "https://www.dn.se/").with(getTracker());
```

- A contentName (required) – the name of the content, e.g. “Ad Foo Bar”.
- A piece (optional) – the actual content. For instance, the path to an image, video, audio or any text.
- A target (optional) – the target of the content. For instance the URL of a landing page.

### Tracking goals

#### *Requires Analytics*

By default, goals are defined as “matching” parts of the screen path or screen title. If you want to trigger a conversion manually or track some user interaction, call the method `goal`. Read further about what a goal is in [Goal in Piwik PRO](#).

```
TrackHelper.track().goal(1).revenue(revenue).with(tracker)
```

- A goal (required) – a tracking request will trigger a conversion for the goal of the website being tracked with this ID.
- Revenue (optional) – a monetary value that has been generated as revenue by goal conversion.

Create, view or manage goals is available in the Analytics tab, “Goals” left menu, “Manage goals” section.

### Tracking ecommerce transactions

#### *Requires Analytics*

If your organization depends on online sales, you need detailed analysis to transform raw e-commerce stats into actionable insights. Revenue, orders, conversion rates, and a host of other product statistics can be analyzed by integrating Piwik with your e-commerce solution.

SDK provides the `order` method that can be used for tracking the orders (including the order items). Sample usage:

```
Tracker tracker = ((YourApplication) getApplication()).getTracker();
EcommerceItems items = new EcommerceItems();
// EcommerceItems.Item("<sku>").name("<product>").category("<category>").price(<cents>
↳ .quantity(<number>)
items.addItem(new EcommerceItems.Item("0123456789012").name("Polo T-shirt").category(
↳ "Men's T-shirts").price(3000).quantity(2));
items.addItem(new EcommerceItems.Item("0129876543210").name("Leather shoes").category(
↳ "Shoes").price(40000).quantity(1));
```

(continues on next page)

(continued from previous page)

```
TrackHelper.track().order("orderId",124144).subTotal(33110).tax(9890).shipping(1000).
↳discount(0).items(items).with(tracker);
```

- `orderId` (required) – a unique String identifying the order
- `grandTotal` (required) – Total amount of the order, in cents
- `subTotal` (optional) – the subTotal (net price) for the order, in cents
- `tax` (optional) – the tax for the order, in cents
- `shipping` (optional) – the shipping for the order, in cents
- `discount` (optional) – the discount for the order, in cents
- `items` (optional) – the items included in the order, use the `EcommerceItems` class to instantiate items

## Tracking campaigns

### *Requires Analytics*

Tracking [campaigns](#) URLs configured with the online *Campaign URL Builder tool*, allow you to measure how different campaigns (for example with Facebook ads or direct emails) bring traffic to your application. You can track these URLs from the application via the `campaign` method:

```
TrackHelper.track().campaign(new URL("http://example.org/offer.html?pk_campaign=Email-
↳SummerDeals&pk_keyword=LearnMore")).with(getTracker());
```

- A URL (required) – the campaign URL. HTTPS, HTTP and FTP are valid, however, the URL must contain campaign name and keyword parameters.

## Tracking custom variables

### *Requires Analytics*

A [custom variable](#) is a custom name-value pair that you can assign to your users or screen views, and then visualize the reports of how many visits, conversions, etc. for each custom variable. A custom variable is defined by a name — for example, “User status” — and a value – for example, “LoggedIn” or “Anonymous”. It is required for names and values to be encoded in UTF-8.

Each custom variable has a scope. There are two types of custom variables scope - *visit scope* and *screen scope*. The visit scope can be used for any tracking action, and the screen scope can only be applied to tracking screen views.

To set the custom variable of the screen scope, use the `variable` method in the tracking chain:

```
TrackHelper.track()
    .screen("/custom_vars")
    .title("Custom Vars")
    .variable(1, "filter", "price")
    .variable(2, "language", "en")
    .with(getTracker());
```

To use the custom variable of the visit scope, use the `visitVariables` method in the tracking chain:

```
TrackHelper.track()  
    .visitVariables(1, "filter", "price")  
    .visitVariables(2, "language", "en")  
    .event("category", "action")  
    .with(tracker);
```

Please note that for the *Default custom variables* option, use the custom variables of the visit scope with indexes 1-3.

Custom variable is defined by three parameters:

- An index (required) – a given custom variable name must always be stored in the same “index” per session. For example, if you choose to store the variable name = “Gender” in index = 1 and you record another custom variable in index = 1, then the “Gender” variable will be deleted and replaced with a new custom variable stored in index 1.
- A name (required) – this String defines the name of a specific Custom Variable such as “User type” (Limited to 200 characters).
- A value (required) – this String defines the value of a specific Custom Variable such as “Customer” (Limited to 200 characters).

## Tracking custom dimensions

*Requires Analytics*

To track a custom name-value pair assigned to your users or screen views, use [Custom Dimensions](#). Note that the custom value data is not sent by itself, but only with other tracking actions such as screen views, events or other tracking action:

```
TrackHelper.track()  
    .dimension(1, "visit")  
    .dimension(2, "dashboard")  
    .screen("Home screen")  
    .with(tracker);
```

1 and 2 are our dimension slots and `visit`, `dashboard` are the dimension values for the tracked screen view.

```
TrackHelper.track()  
    .dimension(1, "visit")  
    .dimension(2, "billing")  
    .event("category", "action")  
    .with(tracker);
```

1 and 2 are our dimension slots and `visit`, `billing` are the dimension values for the tracked event.

## Tracking user profile attributes

*Requires Audience Manager*

The Audience Manager stores visitors’ profiles which have data from a variety of sources. One of them can be a mobile application. It is possible to enrich the profiles with more attributes by passing any key-value pair e.g. `gender: male`, `favourite food: Italian`, etc. It is recommended to set additional user identifiers such as *email* or *User ID* which will allow the enrichment of existing profiles or merging of profiles rather than creating a new profile. For example, if the user visited the website, performed some actions, filled in a form with his email (his data was tracked and profile created in Audience Manager) and afterwards started using a mobile application, the existing profile will be enriched only if the email was set. Otherwise, a new profile will be created.

For sending profile attributes use `audienceManagerSetProfileAttribute` method:

```
getTracker().setUserMail("john@doe.com");
...
TrackHelper.track().audienceManagerSetProfileAttribute("food", "pizza").add("color",
↳"green").with(getTracker());
```

- A name (required) – defines the profile attribute name (non-null string).
- A value (required) – defines the profile attribute value (non null string).
- An `add` (chain method) – used to specify more attributes to the user within the same event.

Aside from attributes, each event also sends parameters which are retrieved from the tracker instance:

- `WEBSITE_ID` - always sent,
- `USER_ID` - if it is set. [Read more](#) about the User ID,
- `EMAIL` - if it is set. [Read more](#) about the email,
- `VISITOR_ID` - always sent, ID of the mobile application user, generated by SDK
- `DEVICE_ID` - an **Advertising ID** that, by default, is fetched automatically when the tracker instance is created. To turn off automatic fetch, use the `setTrackDeviceId(boolean isTracked)` method:

```
getTracker().setTrackDeviceId(false);
```

Profile attributes for the user that are tracked will be shown on the Audience Manager - Profile Browser tab.

Audience manager events are dispatched together with analytics events. Therefore, settings set in the tracker for analytics events processing (dispatch interval, cache size and age, etc.) will be same for audience manager events. Once the audience manager event is dispatched, it is no longer stored locally.

## Reading user profile attributes

*Requires Audience Manager*

It is possible to read the attributes of a given profile, however, with some limitations. Due to security reasons (to avoid personal data leakage), it is possible to read only attributes that were enabled for API access (whitelisted) in the Attributes section in Audience Manager. To get user profile attributes use the `audienceManagerGetProfileAttributes` method:

```
getTracker().audienceManagerGetProfileAttributes(new Tracker.
↳OnGetProfileAttributes() {
    @Override
    public void onAttributesReceived(Map<String, String> attributes) {
        // handle result
    }

    @Override
    public void onError(String errorData) {
        errorData = TextUtils.isEmpty(errorData) ? "Network error": errorData;
        // handle error
    }
});
```

- An `OnGetProfileAttributes` (required) – callback to handle request result (call is asynchronous), has two methods `void onAttributesReceived(Map<String, String> attributes)` and `void onError(String errorData)`.

- An attributes (output) – dictionary of key-value pairs, where each pair represents the attribute name (key) and value.
- An errorData (output) – in case of error, only this method will be called. The method passes the error string.

## Checking audience membership

*Requires Audience Manager*

Audiences are allowed to check whether or not the user belongs to a specific group of users defined in the data manger panel based on analytics data and audience manager profile attributes. You can check if the user belongs to a given audience, for example, to show a special offer. To check it, use the `checkAudienceMembership` method:

```
getTracker().checkAudienceMembership(audienceId, new Tracker.  
↳OnCheckAudienceMembership() {  
    @Override  
    public void onChecked(boolean isMember) {  
        // handle result  
    }  
  
    @Override  
    public void onError(String errorData) {  
        // handle error  
    }  
});
```

- An audienceId (required) – ID of the audience (Audience Manager -> Audiences tab)
- An OnCheckAudienceMembership (required) – callback to handle request result (call is asynchronous), has two methods `void onChecked(boolean isMember)` and `void onError(String errorData)`
- An isMember (output) – a boolean value that indicates if user belongs to audience with given ID
- An errorData (output) – in case of error, only this method will be called. The method passes the error string.

## 7.1.3 Advanced usage

### User ID

UserID will allow the association of events from various sources to the same user. Each time a new visitor enters your page, Piwik PRO assigns a cookie containing a random string of characters. The purpose of this cookie is for Piwik PRO to be able to recognize the same visitor whenever the website is visited again. However, instead of a random string, you can assign your visitors with your own human-friendly name (ex. visitor email). More about [UserID](#). In order to set UserID, use the `setUserId` method:

```
getTracker().setUserId("John Doe");
```

- A UserID (required) – any non-empty unique string identifying the user. Passing null will delete the current UserID

### User email address

*Used only by Audience Manager*

The user email address is an optional parameter for user identification. Similar to UserID, it allows the association of events from various sources to the same user. To set user email use the `setUserMail` method:



```
getTracker().setUserMail("john@doe.com");
```

- A userMail (required) – any non-null string representing email address

Setting up an email helps the Audience Manager to enrich existing profiles or merge profiles which come from other sources (if they also have an email). Check *Tracking user profile attributes* for more information.

## Visitor ID

To track user sessions on difference sources, the VisitorID parameter is used. VisitorID is randomly generated when the tracker instance is created, and stored between application launches. It is also possible to reset the VisitorID manually:

```
tracker.setVisitorId("0123456789abcdef");
```

- A VisitorID (required) – unique visitor ID, must be 16 characters hexadecimal string.

Every unique visitor must be assigned a different ID and this ID must not change after it is assigned. We recommend using UserID instead of VisitorID.

## Sessions

A session represents a set of user's interactions with your app. By default, Analytics is closing the session after 30 minutes of inactivity, counting from the last recorded event in session and when the user will open up the app again the new session is started. You can configure the tracker to automatically close the session when users have placed your app in the background for a period of time. That period is defined by the `setSessionTimeout` method.

```
tracker.setSessionTimeout(30 * 60 * 1000);
```

- A timeout (required) – session timeout time in ms.

You can manually start a new session when sending a hit to Piwik by using the `startNewSession` method.

```
tracker.startNewSession();
```

## Dispatching

Tracked events are stored temporarily on the queue and dispatched in batches every 30 seconds (default setting). This behavior can be changed with following options:

- `setDispatchInterval(0)` - incoming events will be dispatched immediately
- `setDispatchInterval(-1)` - incoming events will not be dispatched automatically. This lets you gain full control over dispatch process, by using manual dispatch, as in the example below.

```
Tracker tracker = ((MyApplication) getApplication()).getTracker();
tracker.setDispatchInterval(-1);
// Catch and track exception
try {
    cartItems = getCartItems();
} catch (Exception e) {
    tracker.trackException(e, e.getMessage(), false);
    tracker.dispatch();
    cartItems = null;
}
```

In case when more than one event is in the queue, data is sent in bulk (using POST method with JSON payload). It is possible to compress the data before dispatch by using `setDispatchGzipped` method during the app initialization. See the example below for details:

```
private void initPiwik() {
    ...

    //configure dispatcher to compress JSON with gzip
    getTracker().setDispatchGzipped(true);

    ...
}
```

To take advantage of compressed requests you have to configure HTTP server of the tracker. Use `mod_deflate` (on Apache) or `lua_zlib` (on Nginx). Helpful resources:

- [lua\\_zlib](#)
- [lua-nginx-module](#)
- [inflate.lua samples](#)

### Custom queries

You should be able to use all common actions through the `TrackHelper` utility, but in some instances, you may want full control over what is sent to the server.

The base method for any event is `track`. You can create your own `TrackMe` objects, set the parameters and then send it:

```
TrackMe trackMe = new TrackMe()
trackMe.set...
/* ... */
Tracker tracker = ((YourApplication) getApplication()).getTracker();
tracker.track(trackMe);
```

### Default custom variables

SDK can automatically add information about the platform version, OS version and app version in custom variables with indexes 1-3. By default, this option is turned on. This can be changed via the `setIncludeDefaultCustomVars` method:

```
getTracker().setIncludeDefaultCustomVars(false);
```

In case you need to configure custom variables separately, turn off this option and see the section above regarding tracking custom variables.

### Local storage limits

You can set limits for storing events related to maximum size and time for which events are saved in local storage as below. Events older than the set limit will be discarded on the next dispatch attempt. The Piwik backend accepts backdated events for up to 24 hours by default.

To change offline cache age use the `setOfflineCacheAge` method:

```
tracker.setOfflineCacheAge(80085);
```

- A limit (required) – time in ms after which events are deleted, 0 = unlimited, -1 = disabled offline cache. By default, the limit is set to  $24 * 60 * 60 * 1000$  ms = 24 hours.

You can also specify how large the offline cache may be. If the limit is reached, the oldest files will be deleted first. To change offline cache size use the `setOfflineCacheSize` method:

```
tracker.setOfflineCacheSize(16 * 1000 * 1000);
```

- A limit (required) – size in bytes after which events are deleted, 0 = unlimited. By default, the limit is set to  $4 * 1024 * 1024$  bytes = 4 Mb.

## Opt out

You can enable an app-level opt-out flag that will disable Piwik PRO tracking across the entire app. Note that this flag must be set each time the app starts up and will default to `false`. To set the app-level opt-out, use:

```
getTracker().setOptOut(true);
```

## Dry run

The SDK provides a `dryRun` flag that, when set, prevents any data from being sent to Piwik. The `dryRun` flag should be set whenever you are testing or debugging an implementation and do not want test data to appear in your Piwik reports. To set the dry run flag, use:

```
getTracker().setDryRunTarget(Collections.synchronizedList(new ArrayList<Packet>()));
```

- A `dryRunTarget` (required) – a data structure the data should be passed into `List<Packet>` type. Set it to null to disable dry run.

## 7.1.4 License

*Piwik PRO Android SDK is released under the BSD-3 Clause license.*

Copyright 2018 Piwik PRO team

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Piwik team nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED

TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## 7.2 Piwik PRO SDK for iOS

### 7.2.1 SDK configuration

#### Server

- You need a Piwik PRO account on the cloud or an on-premises setup which your mobile app will communicate with. For details, please visit the [Piwik PRO website](#).
- Create a new website (or app) in the Piwik PRO web interface.
- Copy and note the Website ID from “Settings > Websites” and your server address.

#### Client

##### Including the library

Use the following in your Podfile:

```
pod 'PiwikPROSDK', '~> VERSION'
```

Replace VERSION with the latest release name, e.g. '~> 1.0.0'.

Then run `pod install`. In every file you wish to use the PiwikPROSDK, don't forget to import it.

##### Configuration

To configure the tracker you will need the URL address of your tracking server and website ID (you can find it in *Settings > Websites* on the web interface).

Open the *AppDelegate.m* file and add sdk import:

```
#import <PiwikPROSDK/PiwikPROSDK.h>
```

Configure the tracker with your website ID and URL in the application delegate:

```
- (BOOL)application:(UIApplication *)application_
↳didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // Configure the tracker in your application delegate
    [PiwikTracker sharedInstanceWithSiteID:@"01234567-89ab-cdef-0123-456789abcdef"
↳baseURL:[NSURL URLWithString:@"https://your.piwik.pro.server.com"]];
    return YES;
}
```

## 7.2.2 Using Piwik PRO SDK

SDK supports several different types of actions which can be tracked. If the event dispatch was unsuccessful (network error, server error, etc), the event will be saved in the disk cache and processing will be retried during the next dispatch attempt (in configured dispatch interval). Each event is stored in the disk cache for a specified cache age - the time which defines the maximum time for which event is saved locally.

### Tracking screen views

#### *Requires Analytics*

The basic functionality of the SDK is the tracking screen views which represent the content the user is viewing in the application. To track a screen you only need to provide the name of the screen. This name is internally translated by the SDK to an HTTP URL as the Piwik PRO server uses URLs for tracking views. Additionally, Piwik PRO SDK uses prefixes which are inserted in generated URLs for various types of action(s). For tracking screen views it will use prefix *screen* by default however automatic prefixing can be disabled with the *isPrefixingEnabled* option. To start tracking screen views, add the following code to your view controllers.

```
- (void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];
    [[PiwikTracker sharedInstance] sendView:@"Menu"];
}
```

- A screen name (required) – title of the action being tracked. The appropriate screen path will be generated for this action.

### Tracking custom events

#### *Requires Analytics*

Custom events can be used to track the user's interaction with various custom components and features of your application, such as playing a song or a video. Category and action parameters are required while the name and value are optional. You can read more about events in the Piwik PRO [documentation](#) and [ultimate guide to event tracking](#).

```
[[PiwikTracker sharedInstance] sendEventWithCategory:@"Video" action:@"Play" name:@"Pirates" value:@185];
```

The `sendEventWithCategory` method allows to specify next parameters:

- A category (required) – this String defines the event category. You may define event categories based on the class of user actions ( e.g. clicks, gestures, voice commands), or you may define them based upon the features available in your application (e.g. play, pause, fast forward, etc.).
- An action (required) – this String defines the specific event action within the category specified. In the example, we are essentially saying that the category of the event is user clicks, and the action is a button click.
- A name (optional) – this String defines a label associated with the event. For example, if you have multiple button controls on a screen, you might use the label to specify the specific View control identifier that was clicked.
- A value (optional) – this Float defines a numerical value associated with the event. For example, if you were tracking “Buy” button clicks, you might log the number of items being purchased, or their total cost.

## Tracking exceptions

### Requires Analytics

Tracking exceptions allow the measurement of exceptions and errors in your app. Exceptions are tracked on the server in a similar way as screen views, however, URLs internally generated for exceptions always use the *fatal* or *caught* prefix and, additionally, if the `isPrefixingEnabled` option is enabled, then the additional *exception* prefix is added.

```
[[PiwikTracker sharedInstance] sendExceptionWithDescription:@"Content download error" ↵  
↵isFatal:YES];
```

- A description (required) – provides the exception message.
- An `isFatal` (required) – true if an exception is fatal.

Bear in mind that Piwik is not a crash tracker, use this sparingly.

## Tracking social interactions

### Requires Analytics

Social interactions such as likes, shares and comments in various social networks can be tracked as below. This, again, is tracked in a similar way as screen views but the *social* prefix is used when the default `isPrefixingEnabled` option is enabled.

```
[[PiwikTracker sharedInstance] sendSocialInteractionWithAction:@"like" target:@"Dogs" ↵  
↵network:@"Facebook"];
```

- An interaction (required) – defines the social interaction, e.g. “Like”.
- A network (required) – defines the social network associated with interaction, e.g. “Facebook”
- A target (optional) – the target for which this interaction occurred, e.g. “Dogs”.

The URL corresponds to String, which includes network, interaction and target parameters separated by a slash.

## Tracking downloads

### Requires Analytics

You can track the downloads initiated by your application.

```
[[PiwikTracker sharedInstance] sendDownload:@"http://your.server.com/bonusmap.zip"];
```

- A URL (required) – the URL of the downloaded content.

No prefixes are used for tracking downloads, but each event of this type use an additional parameter `download` whose value equals to specified URL. On the analytics panel all, downloads can be viewed in the corresponding section.

## Tracking application installs

### Requires Analytics

You can also track installations of your application. This event is sent to the server only once per application version therefore if you wish to track installs, then you can add it in your application delegate immediately after configuring the tracker.

```

- (BOOL)application:(UIApplication *)application
↳didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // Configure the tracker in your application delegate
    [PiwikTracker sharedInstanceWithSiteID:@"01234567-89ab-cdef-0123-456789abcdef"
↳baseUrl:[NSURL URLWithString:@"https://your.piwik.pro.server.com"]];
    [[PiwikTracker sharedInstance] sendApplicationDownload];
    return YES;
}

```

Application installation is only tracked during the first launch. In the case of the application being installed but not run, the app installation will not be tracked.

## Tracking outlinks

### *Requires Analytics*

For tracking outlinks to external websites or other apps opened from your application use the `sendOutlink` method:

```
[[PiwikTracker sharedInstance] sendOutlink:@"http://great.website.com"];
```

- A URL (required) – defines the outlink target. HTTPS, HTTP and FTP are valid.

## Tracking search operations

### *Requires Analytics*

Tracking search operations allow the measurement of popular keywords used for various search operations performed inside your application. It can be done via the `sendSearchWithKeyword` method:

```
[[PiwikTracker sharedInstance] sendSearchWithKeyword:@"Space" category:@"Movies"
↳numberOfHits:@42];
```

- `keyword` (required) – the searched query that was used in the app.
- `category` (optional) – specify a search category.
- `numberOfHits` (optional) – we recommend setting the search count to the number of search results displayed on the results page. When keywords are tracked with a count of 0, they will appear in the “No Result Search Keyword” report.

## Tracking content impressions and interactions

### *Requires Analytics*

You can track the impression of the ad in your application as below:

```
[[PiwikTracker sharedInstance] sendContentImpressionWithName:@"name" piece:@"piece"
↳target:@"target"];
```

When the user interacts with the ad by tapping on it, you can also track it with a similar method:

```
[[PiwikTracker sharedInstance] sendContentInteractionWithName:@"name" piece:@"piece"
↳target:@"target"];
```

- A `contentName` (required) – the name of the content, e.g. “Ad Foo Bar”.

- A piece (optional) – the actual content. For instance the path to an image, video, audio, any text.
- A target (optional) – the target of the content e.g. the URL of a landing page.

### Tracking goals

#### *Requires Analytics*

Goaltracking is used to measure and improve your business objectives. To track goals, you first need to configure them on the server in your web panel. Goals such as, for example, subscribing to a newsletter can be tracked as below with the goal ID that you will see on the server after configuring the goal and optional revenue. The currency for the revenue can be set in the Piwik PRO Analytics settings. You can read more about goals [here](#).

```
[[PiwikTracker sharedInstance] sendGoalWithID:2 revenue:@30];
```

- A goal (required) – tracking request will trigger a conversion for the goal of the website being tracked with this ID.
- revenue (optional) – a monetary value that was generated as revenue by this goal conversion.

### Tracking ecommerce transactions

#### *Requires Analytics*

Ecommerce transactions (in-app purchases) can be tracked to help you improve your business strategy. To track a transaction you must provide two required values - the transaction identifier and `grandTotal`. Optionally, you can also provide values for `subTotal`, `tax`, `shippingCost`, `discount` and list of purchased items as in the example below.

```
[[PiwikTracker sharedInstance] sendTransaction:[PiwikTransaction_
↳transactionWithBlock:^(PiwikTransactionBuilder *builder) {
    builder.identifier = @"transactionID";
    builder.grandTotal = @5.0;
    builder.subTotal = @4.0;
    builder.tax = @0.5;
    builder.shippingCost = @1.0;
    builder.discount = @0.0;
    [builder addItemWithSku:@"sku1" name:@"bonus" category:@"maps" price:@2.0_
↳quantity:@1];
    [builder addItemWithSku:@"sku2" name:@"home" category:@"maps" price:@3.0_
↳quantity:@1];
}]];
```

- An identifier (required) – a unique string identifying the order
- `grandTotal` (required) – The total amount of the order, in cents
- `subTotal` (optional) – the subtotal (net price) for the order, in cents
- `tax` (optional) – the tax for the order, in cents
- `shipping` (optional) – the shipping for the order, in cents
- `discount` (optional) – the discount for the order, in cents
- Items (optional) – the items included in the order, use the `addItemWithSku` method to instantiate items



## Tracking campaigns

### *Requires Analytics*

Tracking campaign URLs created with the online [Campaign URL Builder tool](#) allow you to measure how different campaigns (for example with Facebook ads or direct emails) bring traffic to your application. You can register a custom URL schema in your project settings to launch your application when users tap on the campaign link. You can track these URLs from the application delegate as below. The campaign information will be sent to the server together with the next analytics event. More details about campaigns can be found in the [documentation](#).

```
- (BOOL)application:(UIApplication *)app openURL:(NSURL *)url options:(NSDictionary_
↳ *)options
{
    return [[PiwikTracker sharedInstance] sendCampaign:url.absoluteString];
}
```

- A URL (required) – the campaign URL. HTTPS, HTTP and FTP are valid - the URL must contain a campaign name and keyword parameters.

## Tracking with custom variables

### *Requires Analytics*

To track custom name-value pairs assigned to your users or screen views, you can use custom variables. A custom variable can have a visit scope, which means that they are assigned to the whole visit of the user or action scope meaning that they are assigned only to the next tracked action such as screen view. You can find more information about custom variables [here](#):

It is required for names and values to be encoded in UTF-8.

```
[[PiwikTracker sharedInstance] setCustomVariableForIndex:1 name:@"filter" value:@"lcd
↳ " scope:CustomVariableScopeAction];
```

- An index (required) – a given custom variable name must always be stored in the same “index” per session. For example, if you choose to store the variable name = “Gender” in index = 1 and you record another custom variable in index = 1, then the “Gender” variable will be deleted and replaced with new custom variable stored in index 1. Please note that some of the indexes are already reserved. See [Default custom variables](#) section for details.
- A name (required) – this String defines the name of a specific Custom Variable such as “User type”. Limited to 200 characters.
- A value (required) – this String defines the value of a specific Custom Variable such as “Customer”. Limited to 200 characters.
- A scope (required) – this String allows the specification of the tracking event type - “visit”, “action”, etc. The scope is the value from the enum `CustomVariableScope` and could be `CustomVariableScopeVisit` or `CustomVariableScopeAction`.

## Tracking with custom dimensions

### *Requires Analytics*

You can also use custom dimensions to track custom values as below. Custom dimensions can also can have a visit or action scope but first have to be defined on the server in your web panel. More details about custom dimensions can be found in the [documentation](#):

```
[[PiwikTracker sharedInstance] setCustomDimensionForIndex:1 value:@"english"  
↪scope:CustomDimensionScopeVisit];
```

- An index (required) – a given custom dimension must always be stored in the same “index” per session, similar to custom variables. In example 1 is our dimension slot.
- A value (required) – this String defines the value of a specific custom dimension such as “English”. Limited to 200 characters.
- A scope (required) – this String allows the specification of the tracking event type - “visit”, “action”, etc. Scope is the value from enum CustomDimensionScope and could be CustomDimensionScopeVisit or CustomDimensionScopeAction.

## Tracking profile attributes

### *Requires Audience Manager*

The Audience Manager stores visitors’ profiles, which have data from a variety of sources. One of them can be a mobile application. It is possible to enrich the profiles with more attributes by passing any key-value pair like gender: male, favourite food: Italian, etc. It is recommended to set additional user identifiers such as *email* or *User ID*. This will allow the enrichment of existing profiles or merging profiles rather than creating a new profile. For example, if the user visited the website, browsed or filled in a form with his/her email (his data was tracked and profile created in Audience Manager) and, afterwards started using a mobile application, the existing profile will be enriched only if the email was set. Otherwise, a new profile will be created.

For sending profile attributes use the `sendProfileAttributeWithName` method:

```
[[PiwikTracker sharedInstance] sendProfileAttributeWithName:@"food" value:@"chips"];
```

- A name (required) – defines profile attribute name (non-null string).
- A value (required) – defines profile attribute value (non-null string).

Aside from attributes, each event also sends parameters, that are retrieved from the tracker instance:

- WEBSITE\_ID - always sent,
- USER\_ID - if It is set. *Read more* about the User ID,
- EMAIL - if It is set. *Read more* about the email,
- VISITOR\_ID - always sent, ID of the mobile application user, generated by SDK
- DEVICE\_ID - it is a device IDFA, which is not set by default (due to platform limitations). In order to set device ID see *Device ID* section below.

Profile attributes for the user that are tracked will be shown on the Audience Manager - Profile Browser tab.

## Reading user profile attributes

### *Requires Audience Manager*

It is possible to read the attributes of a given profile, however, with some limitations. Due to of security reasons to avoid personal data leakage, it is possible to read only attributes that were enabled for API access (whitelisted) in the Attributes section of Audience Manager. To get user profile attributes use the `audienceManagerGetProfileAttributes` method:

```
[[PiwikTracker sharedInstance] audienceManagerGetProfileAttributes:^(NSDictionary_
↳*profileAttributes, NSError * _Nullable error) {
    // do something with attributes list
}];
```

- `completionBlock` (required) – callback to handle request result (call is asynchronous)
- `profileAttributes` (output) – dictionary of key-value pairs, where each pair represent attribute name (key) and value.
- `errorData` (output) – in case of error only, this method will be called. This method passes the error string.

## Checking audience membership

### *Requires Audience Manager*

Audience check allows one to check if the user belongs to a specific group of users defined in the audience manger panel based on analytics data and audience manager profile attributes. You can check if the user belongs to a given audience, for example, to display him/her some type of special offer like in the example below:

```
[[PiwikTracker sharedInstance] checkMembershipWithAudienceID:@"12345678-90ab-cdef-
↳1234-567890abcdef" completionBlock:^(BOOL isMember, NSError * _Nullable error) {
    // do something if is member or handle error
}];
```

- `audienceId` (required) – ID of the audience (Audience Manager -> Audiences tab)
- `completionBlock` (required) – callback to handle request result (call is asynchronous)
- `isMember` (output) – a boolean value that indicates if the user belongs to an audience with a given ID
- `error` (output) – in case of error only, this method will be called. Method pass the error string.

## 7.2.3 Advanced usage

### User ID

The user ID is an additional, optional non-empty unique string identifying the user (not set by default). It can be, for example, a unique username or user's email address. If the provided user ID is sent to the analytics part together with the visitor ID (which is always automatically generated by the SDK), it allows the association of events from various platforms (for example iOS and Android) to the same user provided that the same user ID is used on all platforms. More about [UserID](#). In order to set User ID use `userID` field:

```
[[PiwikTracker sharedInstance].userID = @"User Name";
```

- `userID` (required) – any non-empty unique string identifying the user. Passing null will delete the current user ID

### User email address

The user email address is another additional, optional string for user identification - if the provided user email is sent to the audience manager part when you send the custom profile attribute configured on the audience manager web panel. Similarly to the user ID, it allows the association of data from various platforms (for example iOS and Android) to the same user as long as the same email is used on all platforms. To set user email use the `userEmail` field:

```
[PiwikTracker sharedInstance].userEmail = @"user@email.com";
```

- A userEmail (required) – any non-null string representing email address

It is recommended to set the user email to track audience manager profile attributes as it will create a better user profile.

### Visitor ID

SDK uses various IDs for tracking the user. The main one is visitor ID, which is internally randomly generated once by the SDK on the first usage and is then stored locally on the device. The visitor ID will never change unless the user removes the application from the device so that all events sent from his device will always be assigned to the same user in the Piwik PRO web panel. We recommend using userID instead of VisitorID.

### Sessions

A session represents a set of user's interactions with your app. By default, Analytics is closing the session after 30 minutes of inactivity, counting from the last recorded event in session and when the user will open up the app again the new session is started. You can configure the tracker to automatically close the session when users have placed your app in the background for a period of time. That period is defined by the `sessionTimeout`:

```
[PiwikTracker sharedInstance].sessionTimeout = 1800
```

- sessionTimeout (required) – session timeout time in seconds. Default: 1800 seconds (30 minutes).

### Device ID

The device ID is used to track the IDFA (identifier for advertising). The IDFA is an additional, optional non-empty unique string identifying the device. If you wish to use the IDFA for tracking then you should set the device ID by yourself. Note that if you plan to send your application to the App Store and your application uses IDFA, but does not display ads, then it may be rejected in the App Store review process. You can set the IDFA as in the example below:

```
#import <AdSupport/ASIdentifierManager.h>

NSString *idfa = [[[ASIdentifierManager sharedInstance] advertisingIdentifier]_
↳ UUIDString];
[PiwikTracker sharedInstance].deviceID = idfa;
```

### Dispatching

All tracking events are saved locally and by default. They are automatically sent to the server every 30 seconds. You can change this interval to any other number as below:

```
[PiwikTracker sharedInstance].dispatchInterval = 60;
```

### Gzip compression

You can enable gzip compression for communication with the server as below. By default, requests to the server do not use compression.

```
[PiwikTracker sharedInstance].useGzip = YES;
```

This feature must also be set on server-side using `mod_deflate/APACHE` or `lua_zlib/NGINX` (`lua_zlib - lua-nginx-module - inflate.lua samples`).

### Default custom variables

The SDK, by default, automatically adds some information in custom variables about the device (index 1), system version (index 2) and app version (index 3). By default, this option is turned on. This behavior can be disabled with the following setting:

```
[PiwikTracker sharedInstance].includeDefaultCustomVariable = NO;
```

In case you need to configure custom variables separately, turn off this option and see the section above about tracking custom variables.

### Local storage limits

You can set limits for storing events related to maximum size and time for which events are saved in local storage. By default, the maximum number of queued events is set to 500 and there is no age limit. It can be changed as below:

```
[PiwikTracker sharedInstance].maxNumberOfQueuedEvents = 100;  
[PiwikTracker sharedInstance].maxAgeOfQueuedEvents = 60 * 60 * 24;
```

- `maxNumberOfQueuedEvents` (required) – the maximum number of events after which events in the queue are deleted. By default, the limit is set to 500.
- `maxAgeOfQueuedEvents` (required) – time in ms after which events are deleted. By default, the limit is set to  $7 * 24 * 60 * 60 * 1000$  ms = 7 days.

### Opt-out

You can disable all tracking in the application by using the opt-out feature. No events will be sent to the server if the opt-out is set. By default, opt-out is not set and events are tracked.

```
[PiwikTracker sharedInstance].optOut = YES;
```

## 7.2.4 License

*Piwik PRO iOS SDK is available under the MIT license.*

Copyright 2018 Piwik PRO team

All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT

HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

**Application** Website or application tracked by PPAS.

**App ID** PPAS *application* identifier (previously **website ID**, **site ID** or **idSite**).

**User** Visitor on tracked *application*.

**Analytics ID** ID assigned to *user* by *Analytics* for the duration of *Analytics* session. It is stored in browser cookie.

**User ID** Permanent ID assigned to *user* by *application* (e.g. username). You can read more about it [here](#).

**Device ID** Device ID (device identification) is a distinctive number associated with a smartphone or similar handheld device. Device IDs are separate from hardware serial numbers.

**Identifier** Unique *user* ID (e.g. *analytics ID*, *user ID*, *device ID* or email).

**Visit** Period of continuous *user* activity on *application*. It ends in the following situations:

- after a period of inactivity (option set to 30 minutes by default)
- at midnight (option enabled by default)
- on campaign change (option enabled by default)
- when HTTP referrer points to different website (option disabled by default)

**Audience** Named set of *attribute* conditions used to define a group of *users* matching them.

**Attribute** Named value assigned to *user* profile.

**Attribute whitelist** List of *user attributes* that are publicly available via Audience Manager API.

---

**Note:** It is still necessary to identify the *user* with his *analytics ID* to access this information.

---

**PII** Personally Identifiable Information.

**Analytics attribute** *Attribute* generated from value provided by *Analytics* (e.g. browser and device data, location data, etc.). You can read more about *attribute* sources [here](#).

---

**Note:** If *custom attribute* uses the same name - it will be represented as a separate *attribute*.

---

**Custom attribute** *Attribute* generated from value provided by source other than *Analytics* (e.g. *Form Tracker*, *SDK*). You can read more about *attribute* sources [here](#).

**Warning:** *Custom attribute* will store only latest value provided by any custom source.

---

**Note:** If *analytics attribute* uses the same name - it will be represented as a separate *attribute*.

---

**Analytics** PPAS component gathering statistics about each *user* of the *application* (previously **Piwik**).



## Symbols

`_paq.push()` (*\_paq method*), 18

## A

`addDownloadExtensions()` (*built-in function*), 37

`addEcommerceItem()` (*built-in function*), 33

`addListener()` (*built-in function*), 41

Analytics, **108**

Analytics attribute, **107**

Analytics ID, **107**

App ID, **107**

`appendToTrackingUrl()` (*built-in function*), 40

Application, **107**

Attribute, **107**

Attribute whitelist, **107**

Audience, **107**

## C

Custom attribute, **108**

## D

`dataLayer.push()` (*dataLayer method*), 55

`deanonymizeUser()` (*built-in function*), 41

`deleteCookies()` (*built-in function*), 39

`deleteCustomDimension()` (*built-in function*), 35

`deleteCustomVariable()` (*built-in function*), 35

Device ID, **107**

`device_id` (*None attribute*), 49

`disableCookies()` (*built-in function*), 39

`discardHashTag()` (*built-in function*), 40

## E

`email` (*None attribute*), 50

`enableCrossDomainLinking()` (*built-in function*), 33

`enableHeartBeatTimer()` (*built-in function*), 33

`enableLinkTracking()` (*built-in function*), 37

## F

`fields` (*None attribute*), 52

## G

`getCurrentUrl()` (*built-in function*), 40

`getCustomDimension()` (*built-in function*), 23, 35

`getCustomVariable()` (*built-in function*), 22, 35

`getLinkTrackingTimer()` (*built-in function*), 38

`getPiwikUrl()` (*built-in function*), 40

`getUserId()` (*built-in function*), 38

`getVisitorId()` (*built-in function*), 38

`getVisitorInfo()` (*built-in function*), 38

## H

`hasCookies()` (*built-in function*), 39

## I

Identifier, **107**

## K

`killFrame()` (*built-in function*), 41

## L

`logAllContentBlocksOnPage()` (*built-in function*), 36

## O

`onFulfilled()` (*built-in function*), 46, 47, 50, 61–64

`onRejected()` (*built-in function*), 46–48, 50, 61–64

## P

PII, **107**

`Piwik.getAsyncTracker()` (*Piwik method*), 32

`Piwik.getTracker()` (*Piwik method*), 32

`ppms.am.api()` (*ppms.am method*), 46

`ppms.cm.api()` (*ppms.cm method*), 55

## R

`redirectFile()` (*built-in function*), 41

`regetUserId()` (*built-in function*), 38

`removeDownloadExtensions()` (*built-in function*), 37

## S

setApiUrl() (*built-in function*), 40  
setCampaignKeywordKey() (*built-in function*), 41  
setCampaignNameKey() (*built-in function*), 41  
setCookieDomain() (*built-in function*), 39  
setCookieNamePrefix() (*built-in function*), 39  
setCookiePath() (*built-in function*), 39  
setCrossDomainLinkingTimeout() (*built-in function*), 33  
setCustomDimension() (*built-in function*), 35  
setCustomRequestProcessing() (*built-in function*), 42  
setCustomUrl() (*built-in function*), 40  
setCustomVariable() (*built-in function*), 34  
setDocumentTitle() (*built-in function*), 40  
setDomains() (*built-in function*), 40  
setDoNotTrack() (*built-in function*), 41  
setDownloadClasses() (*built-in function*), 37  
setDownloadExtensions() (*built-in function*), 37  
setEcommerceView() (*built-in function*), 34  
setGenerationTimeMs() (*built-in function*), 40  
setHeartBeatTimer() (*built-in function*), 41  
setIgnoreClasses() (*built-in function*), 38  
setLinkClasses() (*built-in function*), 37  
setLinkTrackingTimer() (*built-in function*), 38  
setReferralCookieTimeout() (*built-in function*), 39  
setReferrerUrl() (*built-in function*), 40  
setRequestContentType() (*built-in function*), 42  
setRequestMethod() (*built-in function*), 41  
setSecureCookie() (*built-in function*), 39  
setSessionCookieTimeout() (*built-in function*), 39  
setUserId() (*built-in function*), 38  
setUserIsAnonymous() (*built-in function*), 41  
setVisitorCookieTimeout() (*built-in function*), 39  
storeCustomVariablesInCookie() (*built-in function*), 35

## T

trackAllContentImpressions() (*built-in function*), 35  
trackContentImpression() (*built-in function*), 36  
trackContentImpressionsWithinNode() (*built-in function*), 36  
trackContentInteraction() (*built-in function*), 36  
trackContentInteractionNode() (*built-in function*), 36  
trackEcommerceCartUpdate() (*built-in function*), 34  
trackEcommerceOrder() (*built-in function*), 34

trackEvent() (*built-in function*), 33  
trackGoal() (*built-in function*), 33  
trackingType (*None attribute*), 52  
trackLink() (*built-in function*), 36  
trackPageView() (*built-in function*), 32  
trackSiteSearch() (*built-in function*), 33  
trackVisibleContentImpressions() (*built-in function*), 35  
type (*None attribute*), 52

## U

useLabels (*None attribute*), 52  
User, **107**  
User ID, **107**  
user\_id (*None attribute*), 49

## V

Visit, **107**