Piwik PRO Analytics Suite Documentation

Release 16.26

Piwik PRO

Feb 03, 2023

Contents

CHAPTER 1

Web API

Our web API lets you access data and features in Piwik PRO. Whether you need to get raw data, manage tags or add a new user, no problem – our API is at hand and ready to be used in your next project.

See more

1.1 Getting started

To get started with our web API, you need to set up authorization. You can try to follow our example and make a first call – just to see how it's like to work with our API. We also offer documentation that you can use in Postman.

If you're ready to dig deeper, see the full list of *Methods*.

Next steps

1.1.1 Authorization

If you want to access API for the first time, you need to generate your API credentials and use them to create an access token. The token is needed to authenticate API calls.

Our API uses client credentials (OAuth grant type) for obtaining a user token. All data is sent and received as JSON and is compliant with the JSON API specification.

Create API keys

To create API keyes, follow these steps:

1. Log in to Piwik PRO.
2. Go to **Menu > Profile**.
3. Navigate to **API keys**.
4. Click **Create a key**.

5. Enter **Name** and click **OK**.
6. Copy **Client ID** and **Client secret**. They won't be available after you close this window.

Note: Credentials are valid until they are deleted in **Profile**.

Create an access token

To create an access token, follow these steps:

1. Piwik PRO API tokens use JWT format.
2. Make a call:

```
    curl -X POST 'https://<example>/auth/token' -H "Content-Type:␣
↪application/json" --data '{
    "grant_type": "client_credentials",
    "client_id": "<client_id>",
    "client_secret": "<client_secret>"
    }'
```

Note: If you are the Core plan user, replace <example> with <your_account_name>.piwik.pro.

1. Response example:

```
 {"token_type":"Bearer","expires_in":1800,"access_token":"<your_access_
↪token>"}
```

1. Now you can use <your_access_token> to communicate with Piwik PRO API. The token is a Bearer type, so you need to include it within the header in every API call.

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

Note: Every token is valid for 30 minutes. expires_in shows the expiration time in seconds.

Delete API keys

If you no longer want to use generated API credentials in access tokens, you need to delete them.

To delete API credentials, follow these steps:

1. Log in to Piwik PRO.
2. Go to **Menu** > **Profile**.
3. Navigate to **API keys**.
4. Choose credentials that you want to revoke and click **X**.

1.1.2 Make a first call

In this example, we want to perform some basic operations on a user. We'll do the following operations:

- Invite a user
- Get a created user
- Change the user's language
- Delete a user

Note: In our example, we use https://<example> as an account address. An account address has this format: https://example.piwik.pro.

Generate your access token

Example of a request:

POST /auth/token

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

Response example:

```
{
    "token_type":"Bearer",
    "expires_in":1800,
    "access_token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.
↪eyJpc3MiOiJwcG1zIiwiYXVkIjoiaHR0cHM6XC9cL3Rlc3RpbmcucGl3aWsucHJvXC9zZXR5LCJzdWIiOiJkNmNkZGMxMS
↪Nec2mYFRv6manzXjq0sHQxINZvu-fbDYT8AedVHBKYvu1F9hYKaFReY8rNgfsMANw2OX8-
↪IKpTrQb1DyRkG4nxpIEbob528_
↪lPd7roho5mtKlE8sfS9WZE1piYOwaNDySDEUwUowgj2xBiJqSODjxBI6qVhLkynGEEeNBVh-
↪lrUmlcjpYqUc3saHvX72L-rqbIHa_1dzGarR-dcPyns-RpKjZEILzUSYOHdM09KDti-xsG-
↪nbKHGdP8fVEEJPyupnAfJPOLHQg_j1c5IvJSvTKVF3j4_
↪zo6Zw5g8YkaheT9Iwph5BGHFRneXatcmbwKI8JzSDFi6CinzI-okYKRPbg"
}
```

Note: access_token contains your token. You'll need it for all API calls. Every token is valid for 30 minutes.

Invite a user

Request example:

POST /api/users/v2

```
curl -X POST 'https://<example>/api/users/v2' -H "Authorization: Bearer
↪<your_access_token>" -H "Content-Type: application/vnd.api+json" --data '{
  "data": {
    "type": "ppms/user",
    "attributes": {
      "email": "user@example.com",
      "language": "en-US"
    }
  }
}'
```

Replace in your request the following fields:

- <example> with your account address. Example: example.piwik.pro.
- <your_access_token> with your generated access token

Example of a response:

```
{
  "data": {
    "id": "b30e538d-4b05-4a75-ae25-7eb565901f38",
    "type": "ppms/user",
    "attributes": {
      "email": "user@example.com",
      "role": "USER",
      "addedAt": "2021-08-02T12:16:30+00:00",
      "language": "en-US"
    }
  }
}
```

Get a user

After inviting a user, you can get a user.

Request example:

GET /api/users/v2/<user_id>

```
curl 'https://<example>/api/users/v2/b30e538d-4b05-4a75-ae25-7eb565901f38' -
↪H "Authorization: Bearer <your_access_token>"
```

Note: The URL contains b30e538d-4b05-4a75-ae25-7eb565901f38. What is it? It is a user ID. If you want to update a given resource, you need to specify which one. You'll find a user ID in the `data/id` field in the response for adding a user.

Response example:

```
{
  "data": {
    "id": "b30e538d-4b05-4a75-ae25-7eb565901f38",
    "type": "ppms/user",
    "attributes": {
      "email": "user@example.com",
      "role": "USER",
      "addedAt": "2021-08-02T12:16:30+00:00",
      "language": "en-US"
    }
  }
}
```

Change the user's language

If you want to change the user's language after adding a user, you can use the following method.

Request example:

PATCH /api/users/v2/<user_id>

```
curl -X PATCH 'https://<example>/api/users/v2/b30e538d-4b05-4a75-ae25-
↪7eb565901f38' -H "Authorization: Bearer <your_access_token>" -H "Content-
↪Type: application/vnd.api+json" -v --data '{
  "data": {
    "type": "ppms/user",
```

(continues on next page)

(continued from previous page)

```
    "id": "b30e538d-4b05-4a75-ae25-7eb565901f38",
    "attributes": {
      "language": "de-DE"
    }
  }
}'
```

This request changed the user's language name from en-US to de-DE.

Here are some things to know:

- We use -X PATCH before the URL. It means that this request is available using HTTP PATCH method.
- You also need to specify data/id. It's a JSON API requirement.
- data/type is required. For example, when you want to work with a user resource, specify its type as ppms/user.
- You can set only parameters you want to update. For more user attributes, go to User edit reference

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

Delete a user

When you want to remove a user, you can use the following method.

Request example:

DELETE /api/users/v2/<user_id>

```
curl -X DELETE 'https://<example>/api/users/v2/b30e538d-4b05-4a75-ae25-
↪7eb565901f38' -H "Authorization: Bearer <your_access_token>"
```

API will only return 204 No Content status code.

1.1.3 Postman

Postman is the tool that lets you build, send and test API calls. You can easily import Piwik PRO API to Postman and try it out.

Here are the Swagger docs that you can you import:

- Access control
- Apps
- Audit log
- Meta Sites
- Modules
- Tracker settings
- Users
- User Groups
- And 12 more

To use Postman, follow these steps:

1. In Postman, click **Import > File** or **Link**.
2. Pick the file or link you want to import.
3. Done. All of your paths are imported.
4. Replace your account address in the URL. Example: `https://example.piwik.pro`.
5. Add your token: In the selected API call, click **Authorization**.
6. Use the **Bearer Token** type.
7. Paste your token.
8. Click **Send** to call API.

1.1.4 FAQ

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

All API calls need to be created with this header: `Content-Type: application/vnd.api+json`. If you use curl, you need to use this flag: `-H "Content-Type: application/vnd.api+json"`.

Postman allows configuring headers with the header tab.

API returns JWT not found

You need to use your API token with every API call. Always send your API token within this header: `Authorization: Bearer <your_access_token>`. If you use curl, you need to use this flag: `-H "Authorization: Bearer <your_access_token>"`.

Postman allows configuring tokens in the authorization tab. Choose the Bearer Token type and paste the token there. Remember to keep your token secure because it gives access to sensitive data.

API returns Expired JWT Token

Every token is valid for 30 minutes. After the token expires, you can create it again.

API returns access token not authorized

This message means that you sent an access token within a correct Authorization: Bearer field, but the token is invalid. Check your token and try again.

1.2 Resources

Here are all the things you might need when working with our web API.

See more

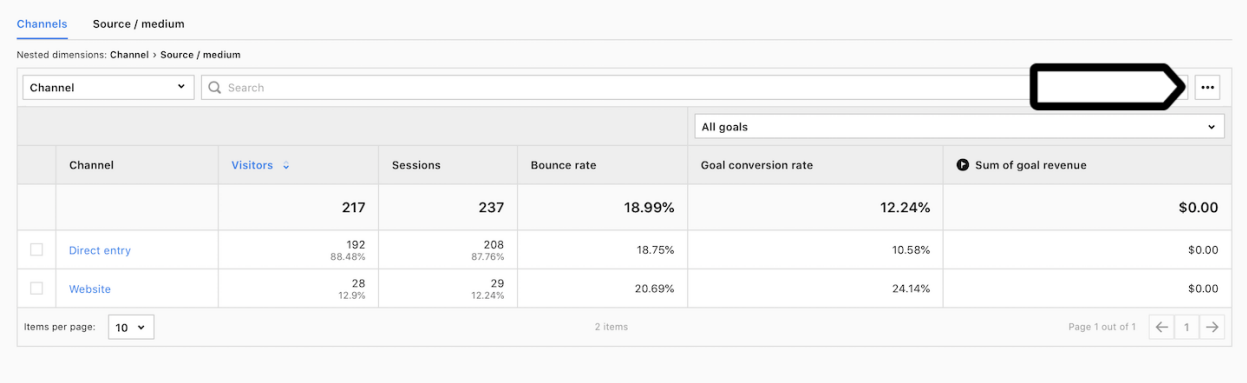
1.2.1 API call definition

When you want to query your database, you can use an API call definition in Piwik PRO. The definition is avaliable for each report inside Piwik PRO. Just make a few clicks and have your API call ready.

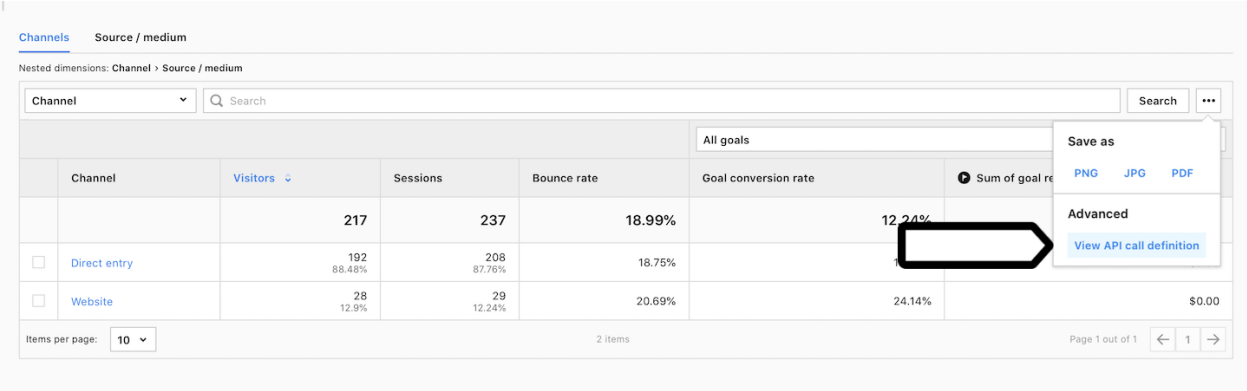
To fetch data directly from a report, follow these steps:

1. Log in to **Piwik PRO**.

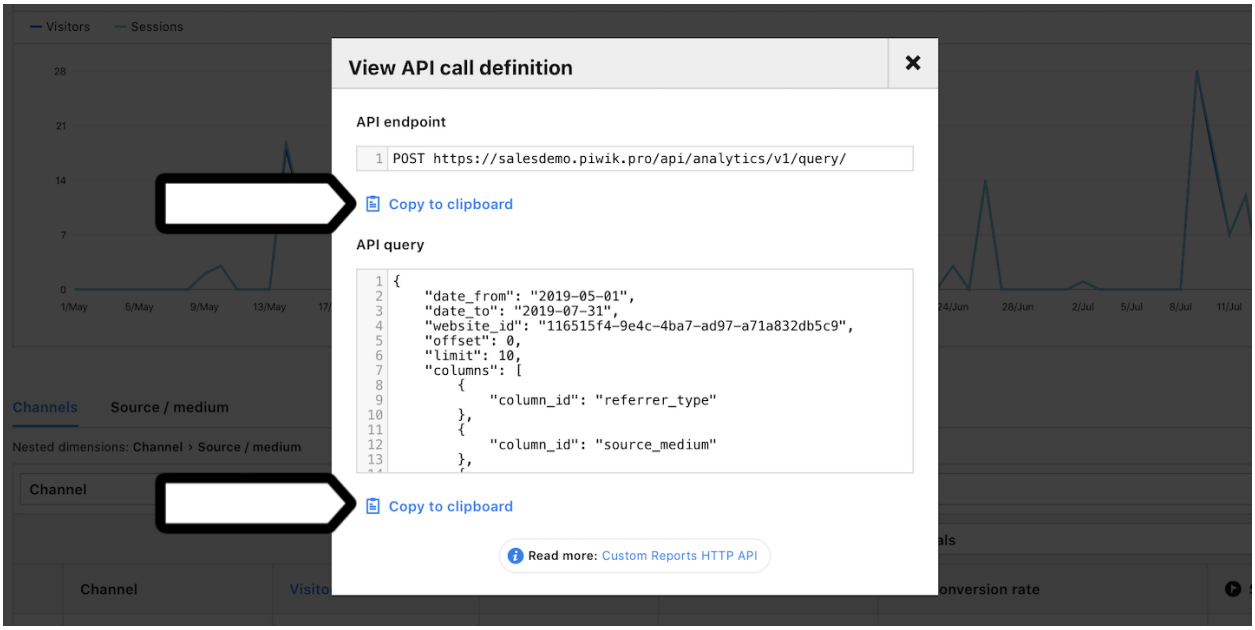
2. Go to **Menu > Analytics**.
3. Navigate to **Reports**, **Custom reports**, **Goals**, or **Ecommerce**.
4. On the left, click a report that you want to work with.
5. Click the three-dot icon next to the report section that you want to use.



6. Click **View API call definition**.



7. Copy an **API endpoint** or **API query**.



1.2.2 Permissions

Here's a list of permissions avaliable in Piwik PRO.

Site/App permissions

↓ Action Permission →	view	edit	edit-publish	manage
view	✓	✓	✓	✓
edit		✓	✓	✓
publish			✓	✓
manage				✓

- **view:** allows viewing an app
- **edit:** allows editing an app
- **publish:** allows publishing changes in Tag Manager
- **manage:** allows managing user permissions for an app
- **add:** allows adding apps (owner only)
- **delete:** allows deleting an app (owner only)

Meta site/app permissions

↓ Action Permission →	view	edit	manage
view	✓	✓	✓
edit		✓	✓
view_details			✓
edit_details			✓
list_apps_in_meta_site			✓
manage			✓

- **view:** allows viewing a meta site in Analytics

- **edit:** allows editing meta site reports in Analytics
- **view_details:** allows viewing meta site details in Administration
- **edit_details:** allows editing meta site details in Administration
- **list_apps_in_meta_site:** allows listing apps in a meta site
- **manage:** allows managing user permissions for a meta site
- **add:** allows creating new meta site (owner only)
- **delete:** allows deleting meta site (owner only)
- **add_app_to_meta_site:** allows adding apps to a meta site (owner only)
- **remove_app_from_meta_site:** allows removing apps from a meta site (owner only)

1.2.3 Metrics & dimensions

When making API calls you can use columns that contain dimensions, metrics and transformations. When you use one of our integrations, you can also use columns that contain additional metrics and dimensions – like the ones from Google Ads, Google Search Console, SharePoint.

Next steps

Piwik PRO: metrics & dimensions

Metrics

Here's a list of core metrics that you can use in API calls. You can create additional metrics using transformations.

Metric name	Column ID	Scope (1)	Type
Events	events	session	int
Consent form impressions	consent_form_impressions	event	int
Consent form clicks	consent_form_clicks	event	int
First consents	consents_first	event	int
Changed consents	consents_changed	event	int
Full consents	consents_full	event	int
Any consents	consents_any	event	int
No consents	consents_none	event	int
No decisions	consents_no_decision	event	int
Analytics consents	consents_analytics	event	int
A/B testing personalization consents	consents_ab_testing_personalization	event	int
Conversion tracking consents	consents_conversion_tracking	event	int
Marketing automation consents	consents_marketing_automation	event	int
Remarketing consents	consents_remarketing	event	int
User feedback consents	consents_user_feedback	event	int
Custom consent 1	consents_custom_1	event	int
Page views	page_views	session	int
Unique page views	unique_page_views	session	int
Entries	entries	session	int
Exits	exits	session	int
Bounces	bounces	session	int

Continued on next page

Table 1 – continued from previous page

Metric name	Column ID	Scope (1)	Type
Sessions	sessions	session	int
Visitors	visitors	session	int
% of returning visitors	returning_visitors_rate	session	float
Users	users	session	int
Visitor IPs	visitor_ips	session	int
Outlinks	outlinks	session	int
Unique outlinks	unique_outlinks	session	int
Downloads	downloads	session	int
Unique downloads	unique_downloads	session	int
Searches	searches	session	int
Unique searches	unique_searches	session	int
Custom events	custom_events	session	int
Unique custom events	unique_custom_events	session	int
Content impressions	content_impressions	session	int
Unique content impressions	unique_content_impressions	session	int
Content interactions	content_interactions	session	int
Unique content interactions	unique_content_interactions	session	int
Goal conversions	goal_conversions	session	int
Unique goal conversions	unique_goal_conversions	session	int
Ecommerce conversions	ecommerce_conversions	session	int
Ecommerce abandoned carts	ecommerce_abandoned_carts	session	int
Unique purchases	unique_purchases	event	int
Entry rate	entry_rate	session	float
Exit rate	exit_rate	session	float
Exit rate events	exit_rate_events	session	float
Bounce rate	bounce_rate	session	float
Bounce rate	bounce_rate_events	session	float
Content interaction rate	content_interaction_rate	session	float
Goal conversion rate	goal_conversion_rate	session	float
Ecommerce conversion rate	ecommerce_conversion_rate	session	float
Events per session	events_per_session	session	float

1. If you make a call that includes one or more columns with the `event` scope, the whole query will be calculated using events – not sessions. This might distort some custom metrics like average session time.

Dimensions

Here's a list of core dimensions that you can use in API calls.

Dimension name	Column ID	Scope (1)	Type	Database type (2)
Visitor ID	visitor_id	session	hex	uint64
User ID	user_id	session	str	string
Cookie ID	cookie_id	session	hex	uint64
Returning visitor	visitor_returning	session	[int, str]	uint8
Session number	visitor_session_number	session	int	uint16
Days since last session	visitor_days_since_last_session	session	int	uint16
Days since first session	visitor_days_since_first_session	session	int	uint16
Days since order	visitor_days_since_order	session	int	uint16
Events in session	session_total_events	session	int	uint16

Table 2 – continued from previous page

Dimension name	Column ID	Scope (1)	Type	Database type (2)
Session time	session_total_time	session	int	uint32
Page views in session	session_total_page_views	session	int	uint16
Outlinks in session	session_total_outlinks	session	int	uint16
Downloads in session	session_total_downloads	session	int	uint16
Site searches in session	session_total_site_searches	session	int	uint16
Custom events in session	session_total_custom_events	session	int	uint16
Content impressions in session	session_total_content_impressions	session	int	uint16
Content interactions in session	session_total_content_interactions	session	int	uint16
Goal conversions in session	session_total_goal_conversions	session	int	uint16
Ecommerce conversions in session	session_total_ecommerce_conversions	session	int	uint16
Abandoned carts in session	session_total_abandoned_carts	session	int	uint16
Unique page views in session	session_unique_page_views	session	int	uint16
Unique outlinks in session	session_unique_outlinks	session	int	uint16
Unique downloads in session	session_unique_downloads	session	int	uint16
Unique site searches in session	session_unique_searches	session	int	uint16
Unique custom events in session	session_unique_custom_events	session	int	uint16
Unique content impressions in session	session_unique_content_impressions	session	int	uint16
Unique content interactions in session	session_unique_content_interactions	session	int	uint16
Goals converted in session	session_goals	session	array	array of int32
Shopping stage	session_ecommerce_status	session	[int, str]	uint8
Source	source	session	str	string
Medium	medium	session	str	string
Source/Medium	source_medium	session	str	string
Keyword	keyword	session	str	string
Channel	referrer_type	session	[int, str]	uint8
Referrer URL	referrer_url	session	str	string
Campaign name	campaign_name	session	str	string
Campaign ID	campaign_id	session	str	string
Campaign content	campaign_content	session	str	string
Google Click ID	campaign_gclid	session	str	string
Operating system	operating_system	session	[str, str]	string(3)
Operating system version	operating_system_version	session	str	string
Browser engine	browser_engine	session	str	string
Browser name	browser_name	session	[str, str]	string(2)
Browser version	browser_version	session	str	string
Browser language	browser_language_iso639	session	[str, str]	string(2)
Browser fingerprint	browser_fingerprint	session	int	uint64
Device type	device_type	session	[int, str]	uint8
Device brand	device_brand	session	[str, str]	string(2)
Device model	device_model	session	str	string
Resolution	resolution	session	str	string
Resolution width	resolution_width	session	int	uint16
Resolution height	resolution_height	session	int	uint16
PDF plugin	plugin_pdf	session	int(0,1)	uint8
Flash plugin	plugin_flash	session	int(0,1)	uint8
Java plugin	plugin_java	session	int(0,1)	uint8
Director plugin	plugin_director	session	int(0,1)	uint8
QuickTime plugin	plugin_quicktime	session	int(0,1)	uint8
RealPlayer plugin	plugin_realplayer	session	int(0,1)	uint8

Table 2 – continued from previous page

Dimension name	Column ID	Scope (1)	Type	Database type (2)
Windows Media Player plugin	plugin_windowsmedia	session	int(0,1)	uint8
Gears plugin	plugin_gears	session	int(0,1)	uint8
Silverlight plugin	plugin_silverlight	session	int(0,1)	uint8
Cookie support	plugin_cookie	session	int(0,1)	uint8
Continent	location_continent_iso_code	session	[str, str]	string(2)
Country	location_country_name	session	[str, str]	string
Subdivision	location_subdivision_1_name	session	[str, str]	string
Subdivision 2	location_subdivision_2_name	session	[str, str]	string
City	location_city_name	session	[int, str]	string
Designated market area	location_metro_code	session	[str, str]	string(3)
Latitude	location_latitude	session	float	float64
Longitude	location_longitude	session	float	float64
Provider	location_provider	session	str	string
Organization	location_organization	session	str	string
Session exit URL	session_exit_url	session	str	string
Session exit title	session_exit_title	session	str	string
Session entry URL	session_entry_url	session	str	string
Session entry title	session_entry_title	session	str	string
Session second URL	session_second_url	session	str	string
Session second title	session_second_title	session	str	string
Session bounce	is_bounce	session	int(0,1)	uint8
Event ID	event_id	event	int	uint64
Session ID	session_id	session	int	uint64
Exit view	is_exit	event	int(0,1)	uint8
Entry view	is_entry	event	int(0,1)	uint8
Event type	event_type	event	[int, str]	uint8
Page URL	event_url	event	str	string
Page title	event_title	event	str	string
Outlink URL	outlink_url	event	str	string
Download URL	download_url	event	str	string
Search keyword	search_keyword	event	str	string
Search category	search_category	event	str	string
Search results count	search_results_count	event	int	uint16
Custom event category	custom_event_category	event	str	string
Custom event action	custom_event_action	event	str	string
Custom event name	custom_event_name	event	str	string
Custom event value	custom_event_value	event	float	float64
Content name	content_name	event	str	string
Content piece	content_piece	event	str	string
Content target	content_target	event	str	string
Previous page view URL	previous_event_url	event	str	string
Previous page view title	previous_event_title	event	str	string
Next page view URL	next_event_url	event	str	string
Next page view title	next_event_title	event	str	string
Event index	event_index	event	int	uint16
Page view index	page_view_index	event	int	uint16
Time on page	time_on_page	event	int	uint32
Page generation time	page_generation_time	event	float	float64
Goal name	goal_id	event	[int, str]	int32

Table 2 – continued from previous page

Dimension name	Column ID	Scope (1)	Type	Database type (2)
Goal revenue	goal_revenue	event	float	float64
Lost revenue	lost_revenue	event	float	float64
Order ID	order_id	event	str	string
Item count	item_count	event	int	uint16
Revenue	revenue	event	float	float64
Revenue (Subtotal)	revenue_subtotal	event	float	float64
Revenue (Tax)	revenue_tax	event	float	float64
Revenue (Shipping)	revenue_shipping	event	float	float64
Revenue (Discount)	revenue_discount	event	float	float64
Time until DOM is ready	timing_dom_interactive	event	int	uint32
Time to interact	timing_event_end	event	int	uint32
Consent form view source	consent_source	event	[int, str]	uint8
Consent form interaction type	consent_form_button	event	[int, str]	uint8
Consent scope	consent_scope	event	[int, str]	uint8
Consent action	consent_action	event	[int, str]	uint8
Analytics consent	consent_type_analytics	event	int(0,1)	uint8
AB testing personalization consent	consent_type_ab_testing_personalization	event	int(0,1)	uint8
Conversion tracking consent	consent_type_conversion_tracking	event	int(0,1)	uint8
Marketing automation consent	consent_type_marketing_automation	event	int(0,1)	uint8
Remarketing consent	consent_type_remarketing	event	int(0,1)	uint8
User feedback consent	consent_type_user_feedback	event	int(0,1)	uint8
Custom consent 1	consent_type_custom_1	event	int(0,1)	uint8
Event custom dimension 1	event_custom_dimension_1	event	str	string
Event custom dimension 2	event_custom_dimension_2	event	str	string
Event custom dimension 3	event_custom_dimension_3	event	str	string
Event custom dimension 4	event_custom_dimension_4	event	str	string
Event custom dimension 5	event_custom_dimension_5	event	str	string
Event custom variable key 1	event_custom_variable_key_1	event	str	string
Event custom variable value 1	event_custom_variable_value_1	event	str	string
Event custom variable key 2	event_custom_variable_key_2	event	str	string
Event custom variable value 2	event_custom_variable_value_2	event	str	string
Event custom variable key 3	event_custom_variable_key_3	event	str	string
Event custom variable value 3	event_custom_variable_value_3	event	str	string
Event custom variable key 4	event_custom_variable_key_4	event	str	string
Event custom variable value 4	event_custom_variable_value_4	event	str	string
Event custom variable key 5	event_custom_variable_key_5	event	str	string
Event custom variable value 5	event_custom_variable_value_5	event	str	string
Session custom dimension 1	session_custom_dimension_1	session	str	string
Session custom dimension 2	session_custom_dimension_2	session	str	string
Session custom dimension 3	session_custom_dimension_3	session	str	string
Session custom dimension 4	session_custom_dimension_4	session	str	string
Session custom dimension 5	session_custom_dimension_5	session	str	string
Session custom variable key 1	session_custom_variable_key_1	session	str	string
Session custom variable value 1	session_custom_variable_value_1	session	str	string
Session custom variable key 2	session_custom_variable_key_2	session	str	string
Session custom variable value 2	session_custom_variable_value_2	session	str	string
Session custom variable key 3	session_custom_variable_key_3	session	str	string
Session custom variable value 3	session_custom_variable_value_3	session	str	string
Session custom variable key 4	session_custom_variable_key_4	session	str	string

Table 2 – continued from previous page

Dimension name	Column ID	Scope (1)	Type	Database type (2)
Session custom variable value 4	session_custom_variable_value_4	session	str	string
Session custom variable key 5	session_custom_variable_key_5	session	str	string
Session custom variable value 5	session_custom_variable_value_5	session	str	string
Timestamp	timestamp	session	date	not applicable
Local hour	local_hour	session	int	not applicable
Time of redirections	redirections_time	event	int	not applicable
Domain Lookup Time	domain_lookup_time	event	int	not applicable
Server Connection Time	server_connection_time	event	int	not applicable
Server Response Time	server_response_time	event	int	not applicable
Page Rendering Time	page_rendering_time	event	int	not applicable
IPv4 address	ipv4_address	session	ipv4	not applicable
IPv6 address	ipv6_address	session	ipv6	not applicable
Website name	website_name	session	[str, str]	not applicable

1. If you make a call that includes one or more columns with the `event` scope, the whole query will be calculated using events – not sessions. This might distort some custom metrics like average session time.
2. `Database type` is the source column of a dimension. `Enum` shows the ID type. `Not applicable` shows a dymamic dimension.

Transformation: dimension to metric

Here's a list of transformations that you can use.

Transformation name	Transformation ID	Source types	Result type
Unique Count	unique_count	int, str	int
Min	min	int, float	(as source)
Max	max	int, float	(as source)
Average	average	int, float	float
Median	median	int, float	(as source)
Sum	sum	int, float	(as source)

Transformation: dimension to dimension

Here's a list of transformations that you can use.

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

Google Ads: metrics & dimensions

Metrics

Here's a list of metrics that are avaliable in Piwik PRO when you use Google Ads integration.

Metric name	Column ID	Scope	Type
Impressions (Google Ads)	google_ads_impressions	external	int
Clicks (Google Ads)	google_ads_clicks	external	int
Cost (Google Ads)	google_ads_cost	external	float
Average CPC (Google Ads)	google_ads_average_cpc	external	float
CTR (Google Ads)	google_ads_ctr	external	float
ROAS (Google Ads)	google_ads_roas	session	float

Dimensions

Here's a list of dimensions that are avaliable in Piwik PRO when you use Google Ads integration.

Dimension name	Column ID	Scope	Type	Database type (1)	Nullable	Notes
Source	source	session	str	string	False	
Medium	medium	session	str	string	False	
Source/Medium	source_medium	session	str	string	False	
Keyword	keyword	session	str	string	False	
Device type	device_type	session	[int, str]	uint8	True	`device_type.json`
Session entry URL	session_entry_url	session	str	string	False	
Timestamp	timestamp	session	date	not applicable	False	by default in Raw data API
Website Name	website_name	session	[str, str]	not applicable	False	website UUID
Customer ID (Google Ads)	google_ads_customer_id	session	str_id	string	False	
Customer Name (Google Ads)	google_ads_customer_name	session	[str, str]	not applicable	False	
Campaign ID (Google Ads)	google_ads_campaign_id	session	int_id	int64	False	
Campaign Name (Google Ads)	google_ads_campaign_name	session	[int, str]	not applicable	False	
Ad Group ID (Google Ads)	google_ads_ad_group_id	session	int_id	int64	False	
Ad Group Name (Google Ads)	google_ads_ad_group_name	session	[int, str]	not applicable	False	
Ad Group Ad ID (Google Ads)	google_ads_ad_group_ad_id	session	str_id	string	False	
Ad Group Ad Network Type (Google Ads)	google_ads_ad_network_type	session	[str, str]	string	False	`google_ads_ad_network_type.json`
Ad Group Keyword Match Type (Google Ads)	google_ads_keyword_match_type	external	[str, str]	string	False	`google_ads_keyword_match_type.json`, not available in Raw data API

1. `Database type` is the source column of a dimension. `Enum` shows the ID type. `Not applicable` shows a dymamic dimension.

Mixed queries

If you want to mix Piwik PRO metrics with Google Ads metrics in a single call or filter, you can only use the metrics that are common in both products. Otherwise, you'll receive a Bad Request response.

Here's a list of the metrics that you can use.

Dimension name	Column ID	Scope	Type	Database type (1)	Nullable	Notes
Source	source	session	str	string	False	
Medium	medium	session	str	string	False	
Source/Medium	source_medium	session	str	string	False	
Keyword	keyword	session	str	string	False	
Device type	device_type	session	[int, str]	uint8	True	`device_type.json`
Session entry URL	session_entry_url	session	str	string	False	
Timestamp	timestamp	session	date	not applicable	False	by default in Raw data API
Website Name	website_name	session	[str, str]	not applicable	False	website UUID
Customer ID (Google Ads)	google_ads_customer_id	session	str_id	string	False	
Customer Name (Google Ads)	google_ads_customer_name	session	[str, str]	not applicable	False	
Campaign ID (Google Ads)	google_ads_campaign_id	session	int_id	int64	False	
Campaign Name (Google Ads)	google_ads_campaign_name	session	[int, str]	not applicable	False	
Ad Group ID (Google Ads)	google_ads_ad_group_id	session	int_id	int64	False	
Ad Group Name (Google Ads)	google_ads_ad_group_name	session	[int, str]	not applicable	False	
Ad Group Ad ID (Google Ads)	google_ads_ad_group_ad_id	session	str_id	string	False	
Ad Group Ad Network Type (Google Ads)	google_ads_ad_network_type	session	[str, str]	string	False	`google_ads_ad_network_type.json`

1. `Database type` is the source column of a dimension. `Enum` shows the ID type. `Not applicable` shows a dymamic dimension.

SharePoint: metrics & dimensions

Metrics

Here's a list of metrics that are avaliable in Piwik PRO when you use SharePoint integration.

Metric name	Column ID	Scope	Type
SharePoint shares	sharepoint_shares	session	int
SharePoint likes	sharepoint_likes	session	int
SharePoint comments	sharepoint_comments	session	int
SharePoint promotions	sharepoint_promotions	session	int
SharePoint creations	sharepoint_creations	session	int
SharePoint edits	sharepoint_edits	session	int
SharePoint deletions	sharepoint_deletions	session	int
SharePoint opens	sharepoint_opens	session	int
SharePoint uploads	sharepoint_uploads	session	int
SharePoint item views	sharepoint_item_views	session	int
SharePoint item attachment views	sharepoint_item_attachment_views	session	int
SharePoint item shares	sharepoint_item_shares	session	int

Dimensions

Here's a list of dimensions that are avaliable in Piwik PRO when you use SharePoint integration.

Dimension name	Column ID	Scope (1)	Type	Database type (2)	Nullable	Notes
SharePoint display name	sharepoint_display_name	session	str	string	True	
SharePoint office	sharepoint_office	session	str	string	True	
SharePoint department	sharepoint_department	session	str	string	True	
SharePoint job title	sharepoint_job_title	session	str	string	True	
SharePoint shares in session	session_total_sharepoint_shares	session	int	uint16	False	
SharePoint likes in session	session_total_sharepoint_likes	session	int	uint16	False	
SharePoint comments in session	session_total_sharepoint_comments	session	int	uint16	False	
SharePoint promotions in session	session_total_sharepoint_promotions	session	int	uint16	False	
SharePoint creations in session	session_total_sharepoint_creations	session	int	uint16	False	
SharePoint edits in session	session_total_sharepoint_edits	session	int	uint16	False	
SharePoint deletions in session	session_total_sharepoint_deletions	session	int	uint16	False	
SharePoint opens in session	session_total_sharepoint_opens	session	int	uint16	False	
SharePoint uploads in session	session_total_sharepoint_uploads	session	int	uint16	False	
SharePoint item views in session	session_total_sharepoint_item_views	session	int	uint16	False	
SharePoint item attachment views in session	session_total_sharepoint_item_attachment_views	session	int	uint16	False	
SharePoint item shares in session	session_total_sharepoint_item_shares	session	int	uint16	False	
SharePoint action	sharepoint_action	event	[int, str]	uint8	True	`sharepoint_action.json`
SharePoint object type	sharepoint_object_type	event	[int, str]	uint8	True	`sharepoint_object_type.json`
SharePoint content type	sharepoint_content_type	event	str	string	True	
SharePoint author	sharepoint_author	event	str	string	True	
SharePoint author's display name	sharepoint_author_display_name	event	str	string	True	
SharePoint author's office	sharepoint_author_office	event	str	string	True	
SharePoint author's department	sharepoint_author_department	event	str	string	True	
SharePoint author's job title	sharepoint_author_job_title	event	str	string	True	
SharePoint file url	sharepoint_file_url	event	str	string	True	
SharePoint file type	sharepoint_file_type	event	str	string	True	

1. If you make a call that includes one or more columns with the `event` scope, the whole query will be calculated

using events – not sessions. This might distort some custom metrics like average session time.

2. `Database type` is the source column of a dimension. `Enum` shows the ID type. `Not applicable` shows a dymamic dimension.

Legacy vs. new metrics

If you migrate from Analytics classic to Analytics new, you need to use different metric names in your API calls. Here's a list that will help you with that.

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"}`
Goal conversions	nb_conversions Note: Ecommerce conversion was reported as goal conversion for goal_id 0.	`{"column_id": "goal_conversions"}`

Continued on next page

Table 3 – continued from previous page

Metric name	Legacy API	New API
Ecommerce conversions		`{"column_id": "ecommerce_conversions"}`
Goal conversions (specific goal)	goal_<idGoal>_nb_conversions	`{"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 Note: Definition switches depending on report	`{"column_id": "exit_rate"}`
Exit rate (events)		`{"column_id": "exit_rate_events"}`
Bounces	bounce_count	`{"column_id": "bounces"}`
Bounce rate	bounce_rate Note: Definition switches depending on report	`{"column_id": "bounce_rate"}`
Bounce rate (events)		`{"column_id": "bounce_rate_events"}`
% of returning visitors	–	`{"column_id": "returning_visitors_rate"}`
Visitor IPs	–	`{"column_id": "visitor_ips"}`
Events per session	nb_actions_per_visit Note: Does not include all event types	`{"column_id": "events_per_session"}`
Unique purchases	–	`{"column_id": "unique_purchases"}`

Note: Note: Event dimensions can only be used with metrics calculated for an event dimension.

Calculated metrics

Here's a list of common examples of calculated metrics. Not all possible combinations are listed here.

Metric name	Legacy API	New API
Sum of goal revenue	revenue ecommerce revenue was reported as goal revenue for goal_id 0	`{"column_id": "goal_revenue", "transformation_id": "sum"}`
Sum of ecommerce revenue		`{"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"}`

Note: Note: Event dimensions can only be used with metrics calculated for an event dimension.

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	-

1.2.4 Tags

Deprecated tags

Here's a list of tags that are deprecated and soon will be removed. We reccomend not using them.

- Clicktale Tracking Code tag
- DoubleClick Conversion tag

Renamed tags

Here's a list of renamed tags. The attribute value used for these tags remained unchanged so you don't need to change anything in your setup.

Old tag name	New tag name
Facebook Retargeting Pixel tag	Meta Pixel tag
DoubleClick Floodlight Sales tag	Floodlight Sales tag
DoubleClick Floodlight Counter tag	Floodlight Counter tag

1.2.5 Built-in variables

Here's a list of built-in variables avaliable in Tag Manager.

Name	Variable_id	Description
History source	caf01b44-cdbf-4a7e-a7e8-0150744b4e32	Contains information about the last method/event that updated the History object. It changes its value when a pushState, replaceState, popState or hashchange event occurs.
History state	c98aff1e-a24a-4ab7-bad4-d7e0f056b42e	Contains the current value of HTML5 history state which is of type any. It changes its value when a history state meets a defined condition.
Old history state	a489adb8-f2cf-4a09-8b7a-e6a048b943d7	Contains the previous value of HTML5 history state which is of type any. It changes its value when a previous history state meets a defined condition.
History fragment	d462fb15-d23c-40e2-bc54-494377accccb	Contains the current value of URL fragment identifier (hash). It changes its value when a hash in the page URL meets a defined condition.
Old history fragment	44ef5a4b-4009-4f66-8fb4-c8137c0ed1b0	Contains the previous value of URL fragment identifier (hash). It changes its value when a previous hash in the page URL meets a defined condition.
Click ID	f39468e9-ae2d-4e58-9e2c-73a4d44add6f	Contains the ID of the clicked element. It changes its value when the visitor clicks an element with the chosen ID.
Click Url	efd3558a-714d-4b33-9edd-2bbeb0b5eaf7	Contains the URL of the clicked element. It changes its value when the visitor clicks a link with the chosen URL.
Click Classes	efc09356-e1f2-40f7-8a0a-4c454429e678	Contains the class(es) of the clicked element. It changes its value when the visitor clicks an element with the chosen CSS class.
Click Element	db5a673e-dabe-4dfa-908c-d08b369d08c9	Contains DOM object of clicked HTML element.
Form ID	b1504526-de0d-4c27-9ff0-1ee7734561ed	Contains the ID of the submitted form element. It changes its value when the visitor submits a form with the chosen ID.
Form Url	0e45761b-d00d-4b18-88c1-eb8fc233d3da	Contains the URL of the submitted form element. It changes its value when the visitor submits a form that sends a request to the chosen URL.
Form Classes	d7c47336-342d-4c9b-b35a-fa13f238ca44	Contains the class(es) of the submitted form element. It changes its value when the visitor submits a form with the chosen CSS class.
Form Name	028f3993-dc39-4ad1-8602-f6345c91f3bb	Contains name of the submitted form.
Event	977a1f9d-4fdb-43bf-9549-bdf68a7b69a6	The name of the event emitted on dataLayer. It changes its value when a chosen custom event occurs in dataLayer.
Time on website	4a3cbab8-e239-	It changes its value when the visitor is on the website or page for a certain amount of time. For example, fires a tag when the visitor spends at least 15 seconds on a

1.3 Methods

CHAPTER 2

JS API

Piwik PRO offers two JS APIs – one for Analytics and one for Consent Manager. You can use them after you install Piwik PRO on your site or app. We've put together solid documentation for both JS APIs, so read on and see what you can do with them.

See more

2.1 Analytics

2.1.1 Getting started

Our JavaScript library has methods that let you send data from your site or web app to Piwik PRO. You can use these methods in JavaScript or within our dedicated libraries for Angular and React.

In short, our JavaScript library lets you collect data like:

- Page views
- Goals
- Internal searches
- Downloads and outlinks
- Custom events
- Ecommerce events
- And more

Next steps

- *Plain JavaScript*
- *Angular*
- *React*

- *Methods*

2.1.2 Plain JavaScript

Here are some guidelines on how to use our JavaScript library in Java Script.

Installation

Our JavaScript library can be used only after you installed our container's code (or only a tracking code) on your site. The code creates a `<script>` tag that asynchronously loads the JavaScript library in the website's body section.

If you haven't installed the code yet, you can find it directly in Piwik PRO in **Menu > Administration > Sites & apps > Installation**.

For more, see our installation guides:

- Install a container (with a tracking code)
- Google Tag Manager: install a container (with a tracking code)
- Google Tag Manager: install only a tracking code
- Instapage: install a container (with a tracking code)
- No Piwik PRO Tag Manager: install a tracking code
- Squarespace: install a container (with a tracking code)
- WordPress: install a container (with a tracking code)

Methods used for calls

In JavaScript, our methods can be called in a few ways:

- **JS (queue):** After installing our container's code, it'll create the `_paq` object (a queue). You can use the *push()* method to add methods to the queue. Our tracker will then access and proceed these methods. With this method, you can also use `this` keyword to send a few methods within one call.
- **JS (direct):** After installing our container's code, you can access our tracker directly (and don't use the queue) with the *getTracker()* or *getAsyncTracker()* method.

push()

The **push()** method adds methods to the `_paq` object (a queue). The methods are called after the container's code (or a tracking code) loads on a page. They are called synchronously (one by one).

Syntax

```
_paq.push(command)
```

Parameters

command (string, required)
An array containing our JavaScript methods.

Examples

To send a page view:

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

To send a custom event:

```
_paq.push(["trackEvent", "Button", "Sign up"]);
```

JS this keyword

The JavaScript `this` keyword lets you add a few methods to the `_paq` object (a queue) in a single call.

Examples

To send a page view and a custom event at once:

```
_paq.push([function () {
    this.trackPageView();
    this.trackEvent("Button", "Sign up");
}]);
```

getTracker()

The **getTracker()** method gives you a direct access to an async tracker. An async tracker is the basic one used for collecting data and using async tags. This method lets you read the return value of the sent method. It also lets you send methods for a few sites or apps at once.

Syntax

```
Piwik.getTracker(account-address, site-id)
```

Parameters

account-address (string, required)
Account address in Piwik PRO. Example: https://example.piwik.pro/

site-id (string, required)
Your site or app ID in Piwik PRO where you want to send data. Where to find it?

Return value

An object with account details in Piwik PRO.

Examples

To send a page view:

```
var jstc = Piwik.getTracker("https://example.com/", "45e07cbf-c8b3-42f3-a6d6-
↪a5a176f623ef");
jstc.trackPageView();
```

To send a custom event:

```
var jstc = Piwik.getTracker("https://example.com/", "45e07cbf-c8b3-42f3-a6d6-
↪a5a176f623ef");
jstc.trackEvent("trackEvent", "Button", "Sign up");
```

getAsyncTracker()

The **getAsyncTracker()** method gives you a direct access to an async tracker. An async tracker is used additionally if you've installed the async container on your site. This method lets you read the return value of the sent method. It also lets you send methods for a few sites or apps at once.

Syntax

```
Piwik.getTracker(account-address, site-id)
```

Parameters

account-address (string, required)
Account address in Piwik PRO. Example: https://example.piwik.pro/

site-id (string, required)
Your site or app ID in Piwik PRO where you want to send data. Where to find it?

Return value

An object with account details in Piwik PRO.

Examples

To send a page view:

```
var jstc = Piwik.getAsyncTracker("https://example.com/", "45e07cbf-c8b3-42f3-a6d6-
↪a5a176f623ef");
jstc.trackPageView();
```

To send a custom event:

```
var jstc = Piwik.getAsyncTracker("https://example.com/", "45e07cbf-c8b3-42f3-a6d6-
↪a5a176f623ef");
jstc.trackEvent("trackEvent", "Button", "Sign up");
```

Reserved variable names

When you use our JavaScript library, you can't name your variables with names that we've set as global variables – it can break the tracking. Here's the list of reserved variable names:

- Piwik
- _paq
- JSON_PIWIK
- piwikPluginAsyncInit
- piwikAsyncInit
- AnalyticsTracker
- piwik_install_tracker
- piwik_tracker_pause
- piwik_download_extensions
- piwik_hosts_alias
- piwik_ignore_classes
- piwik_log
- piwik_track
- sevenTag

2.1.3 Frameworks

If you use React or Angular to build your website or web app, you can install our dedicated libraries and use Piwik PRO to collect visitor data or user data. You don't need to use our JavaScript methods manually, they are all built in as ready-to-use services.

Here is a list of the supported frameworks together with the links to their dedicated libraries:

Framework	Library
Angular	@piwikpro/ngx-piwik-pro (NPM / Yarn)
React	@piwikpro/react-piwik-pro (NPM / Yarn)

See detailed descriptions:

Angular

This library lets you start collecting data from your web app. It also helps you control which data you collect — like page views, virtual page views, custom events, and more. The library contains modules with methods.

To call methods in Angular, you'll use `CustomEventsService` or `PageViewsService`.

Installation

To install JS library for Angular, follow these steps:

1. In your project folder, run the following command:

```
npm install @piwik-pro/ngx-piwik-pro
```

or

```
yarn add @piwikpro/ngx-piwik-pro
```

2. Add the `NgxPiwikProModule` module in your highest level app module. Call the **forRoot()** method by passing your account address (Example: https://example.piwik.pro/) and the site ID (Where to find it?):

```
import { NgxPiwikProModule } from '@piwik-pro/ngx-piwik-pro';

  @NgModule({
    declarations: [
      AppComponent
      ],
    imports: [
      BrowserModule,
      NgxPiwikProModule.forRoot('site-id', 'account-address')
      ],
    providers: [],
    bootstrap: [AppComponent]
    })
    export class AppModule { }
```

Note: This method makes sure that collected data is sent to the your account in Piwik PRO and is reported as a corresponding site or app.

3. Add tracking methods like page views or custom events to your application.
4. Data will appear in reports in about an hour. Data in the tracker debugger will appear instantly.

Additional setup for SPA

If your web app is built as a single-page application (SPA), you need to track virtual page views. In SPAs, when a user goes from one page to the other, the page doesn't reload. It loads just once at the beginning of the session. Because of that, page views can't be recorded in the usual way. You need virtual page views.

To automatically track virtual page views in Angular projects, you need to follow these steps:

1. Add `NgxPiwikProRouterModule` on AppModule to enable automatic tracking of Router events.

Example:

```
import { NgxPiwikProModule, NgxPiwikProRouterModule } from '@piwik-pro/ngx-piwik-pro';
...

@NgModule({
...
  imports: [
  ...
  NgxPiwikProModule.forRoot('account-address'),
  NgxPiwikProRouterModule
```

(continues on next page)

(continued from previous page)

```
  //  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  ]
  })
  export class AppModule {}
```

Note: The `NgxPiwikProRouterModule` module subscribes to Router events when the bootstrap component is created. After that, it cleans up any subscriptions related to the previous component when it is destroyed. If you use this module with server-side rendering or multiple bootstrap components, you may get some issues. In that case, we recommend subscribing to the page view events manually.

2. Additionally, you can use the following include/exclude settings:

- { include: ['/full-uri-match'] } – simple route matching
- { include: ['*/public/*'] } – wildcard route matching
- { include: [/^/public/.*/] } – regular expression route matching

Example:

```
import { NgxPiwikProModule, NgxPiwikProRouterModule } from '@piwik-pro/ngx-piwik-pro';
  ...

@NgModule({
  ...
  imports: [
    ...
    NgxPiwikProModule.forRoot('account-address'),
    NgxPiwikProRouterModule.forRoot({ include: [...], exclude: [...] })
//  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  ]
})
export class AppModule {}
```

Methods

Here's a list of all JS methods you can use in your Angular project. Descriptions and other information are available after clicking on links.

A

- *addDownloadExtensions()*
- *addEcommerceItem()*

C

- *clearEcommerceCart()*

D

- *deleteCustomDimension()*

E

- enableLinkTracking()

G

- getCustomDimensionValue()

- getEcommerceItems()
- getLinkTrackingTimer()
- getUserId()
- getVisitorId()
- getVisitorInfo()

R

- removeDownloadExtensions()
- removeEcommerceItem()
- resetUserId()

S

- setCustomDimensionValue()
- setDownloadClasses()
- setDownloadExtensions()
- setEcommerceView()
- setIgnoreClasses()
- setLinkClasses()
- setLinkTrackingTimer()
- setUserId()

T

- trackContentImpression()
- trackContentInteraction()
- trackEcommerceCartUpdate()
- trackEcommerceOrder()
- trackEvent()
- trackGoal()
- trackLink()
- trackPageView()
- trackSiteSearch()

React

This library lets you start collecting data from your web app. It also helps you control which data you collect — like screen views, custom events, goals, and more. The library contains modules with methods.

To call methods in React, you'll use `CustomEventsService` or `PageViewsService`.

Installation

To install JS library for React, follow these steps:

1. In your project folder, run the following command:

```
npm install @piwik-pro/react-piwik-pro
```

or

```
yarn add @piwikpro/react-piwik-pro
```

2. Add the `PiwikPro` module to your project files. Call the **initialize()** method by passing your account address (Example: https://example.piwik.pro/) and the site ID (Where to find it?):

```
import PiwikPro from '@piwik-pro/react-piwik-pro';

PiwikPro.initialize('site-id', 'account-address');

ReactDOM.render(<App />, document.getElementById('root'))
```

Note: This method makes sure that collected data is sent to the your account in Piwik PRO and is reported as a corresponding site or app.

3. Add tracking methods like screen views or custom events to your application.
4. Data will appear in reports in about an hour. Data in the tracker debugger will appear instantly.

Additional setup for SPA

If your web app is built as a single-page application (SPA), you need to track virtual page views. In SPAs, when a user goes from one page to the other, the page doesn't reload. It loads just once at the beginning of the session. Because of that, page views can't be recorded in the usual way. You need virtual page views.

To automatically track virtual page views in React projects, you need to follow these steps:

1. Something
2. Something
3. Something

Methods

Here's a list of all JS methods you can use in your React project. Descriptions and other information are available after clicking on links.

A

- *addDownloadExtensions()*
- *addEcommerceItem()*

C

- *clearEcommerceCart()*

D

- *deleteCustomDimension()*

E

- enableLinkTracking()

G

- getCustomDimensionValue()
- getEcommerceItems()
- getLinkTrackingTimer()
- getUserId()
- getVisitorId()
- getVisitorInfo()

R

- removeDownloadExtensions()
- removeEcommerceItem()
- resetUserId()

S

- setCustomDimensionValue()
- setDownloadClasses()
- setDownloadExtensions()
- setEcommerceView()
- setIgnoreClasses()
- setLinkClasses()
- setLinkTrackingTimer()
- setUserId()

T

- trackContentImpression()
- trackContentInteraction()
- trackEcommerceCartUpdate()
- trackEcommerceOrder()
- trackEvent()
- trackGoal()
- trackLink()
- trackPageView()
- trackSiteSearch()

2.1.4 Methods

Here's a list of all available JS methods for Analytics in Piwik PRO.

addDownloadExtensions()

The **addDownloadExtensions()** adds a file extension to the existing list of file extensions that should be tracked as downloads. Downloads are clicks on links to downloadable files.

Syntax

JS

```
addDownloadExtensions(extensions)
```

Angular

```
addDownloadExtensions(extensions: string[])
```

React

```
addDownloadExtensions(extensions: string[])
```

Parameters

extensions (string | array <string>, required)
The list of file extensions to be tracked as downloads. Can be written as string (Example: “zip|rar”) or an array (Example: [“zip”, “rar”]).

Examples

To add a mhj|docx extensions to the existing list of file extensions:

JS (queue)

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

JS (direct)

```
var jstc = Piwik.getTracker(
  "https://example.com/",
  "45e07cbf-c8b3-42f3-a6d6-a5a176f623ef"
);
jstc.addDownloadExtensions("mhj|docx");
```

Angular

```

```

React

```

```

Notes

- The list of file extensions is not persisted in a visitor's browser. You need to call it on each page load.
- **We check for extensions at the end of the URL path and in query parameter values. Here are a few examples of how we d**
  - http://example.com/path/file.zip
  - http://example.com/path/file.zip#hello
  - http://example.com/path/file.zip?a=102
  - http://example.com/path/?a=file.zip
  - http://example.com/path/?a=file.zip&b=29
- Here's the default list of file extensions tracked as downloads: 7z, aac, apk, arc, arj, asf, asx, avi, azw3, bin, csv, deb, dmg, doc, docx, epub, exe, flv, gif, gz, gzip, hqx, ibooks, jar, jpg, jpeg, js, mobi, mp2, mp3, mp4, mpg, mpeg, mov, movie, msi, msp, odb, odf, odg, ods, odt, ogg, ogv, pdf, phps, png, ppt, pptx, qt, qtm, ra, ram, rar, rpm, sea, sit, tar, tbz, tbz2, bz, bz2, tgz, torrent, txt, wav, wma, wmv, wpd, xls, xlsx, xml, z, zip.

Related methods

- trackLink()
- enableLinkTracking()
- disableLinkTracking()
- setIgnoreClasses()
- setLinkClasses()
- setDownloadClasses()
- setDownloadExtensions()
- removeDownloadExtensions()
- getConfigDownloadExtensions()

addEcommerceItem()

The **addEcommerceItem()** method adds a product to the cart.

Syntax

JS

```
addEcommerceItem(productSKU[, productName[, productCategory[, productPrice[,␣
↪productQuantity]]]])
```

Angular

```
addEcommerceItem(productSKU: string, productName: string, productCategory: string |␣
↪string[], productPrice: number, productQuantity: number)
```

React

```
addEcommerceItem(productSKU: string, productName: string, productCategory: string |␣
↪string[], productPrice: number, productQuantity: number)
```

Parameters

productSKU (string, required)
The stock-keeping unit of the added product.

productName (string, optional)
The name of the added product.

productCategory (string | array<string>, optional)
The category of the added product. It can be an array of up to 5 categories.

productPrice (number, optional)
The price of the added product.

productQuantity (number, optional)
The number of added items.

Examples

To add one product to the cart:

JS (queue)

```
_paq.push([
  "addEcommerceItem",
  "584340",
  "Specialized Stumpjumper",
  "Mountain bike",
  5000,
  1,
]);
```

JS (direct)

```
var jstc = Piwik.getTracker(
  "https://example.com/",
  "45e07cbf-c8b3-42f3-a6d6-a5a176f623ef"
);
jstc.addEcommerceItem(
  "addEcommerceItem",
  "584340",
  "Specialized Stumpjumper",
  "Mountain bike",
  5000,
```

(continues on next page)

(continued from previous page)

```
  1
);
```

Angular

```
```

React

```
```

To add two products to the cart:

JS (queue)

```
_paq.push([
  "addEcommerceItem",
  "584340",
  "Specialized Stumpjumper",
  "Mountain bike",
  5000,
  1,
]);
_paq.push([
  "addEcommerceItem",
  "460923",
  "Specialized Chamonix",
  "Helmets",
  200,
  1,
]);
```

JS (direct)

```
var jstc = Piwik.getTracker(
  "https://example.com/",
  "45e07cbf-c8b3-42f3-a6d6-a5a176f623ef"
);
jstc.addEcommerceItem(
  "addEcommerceItem",
  "584340",
  "Specialized Stumpjumper",
  "Mountain bike",
  5000,
  1
);
jstc.addEcommerceItem(
  "addEcommerceItem",
  "460923",
  "Specialized Chamonix",
  "Helmets",
  200,
  1
);
```

Angular

React

To add a product and send a cart update to Piwik PRO:

JS (queue)

```
_paq.push([
  "addEcommerceItem",
  "584340",
  "Specialized Stumpjumper",
  "Mountain bike",
  5000,
  1,
]);
_paq.push(["trackEcommerceCartUpdate", 5000]);
```

JS (direct)

```
var jstc = Piwik.getTracker(
  "https://example.com/",
  "45e07cbf-c8b3-42f3-a6d6-a5a176f623ef"
);
jstc.addEcommerceItem(
  "addEcommerceItem",
  "584340",
  "Specialized Stumpjumper",
  "Mountain bike",
  5000,
  1
);
jstc.trackEcommerceCartUpdate(5000);
```

Angular

React

To add two products to the cart and send a cart update:

JS (queue)

```
_paq.push([
  "addEcommerceItem",
  "584340",
  "Specialized Stumpjumper",
  "Mountain bike",
  5000,
  1,
]);
_paq.push([
  "addEcommerceItem",
```

(continues on next page)

(continued from previous page)

```
  "460923",
  "Specialized Chamonix",
  "Helmets",
  200,
  1,
]);
_paq.push(["trackEcommerceCartUpdate", 5200]);
```

JS (direct)

```
var jstc = Piwik.getTracker(
  "https://example.com/",
  "45e07cbf-c8b3-42f3-a6d6-a5a176f623ef"
);
jstc.addEcommerceItem(
  "addEcommerceItem",
  "584340",
  "Specialized Stumpjumper",
  "Mountain bike",
  5000,
  1
);
jstc.addEcommerceItem(
  "addEcommerceItem",
  "460923",
  "Specialized Chamonix",
  "Helmets",
  200,
  1
);
jstc.trackEcommerceCartUpdate(5200);
```

Angular

```

```

React

```

```

To track a confirmed order:

JS (queue)

```
// register all purchased items

_paq.push([
  "addEcommerceItem",
  "584340", // SKU
  "Specialized Stumpjumper", // name
  "Mountain bike", // category
  5000, // price
  1, // quantity
]);

_paq.push([
  "addEcommerceItem",
```

(continues on next page)

(continued from previous page)

```
  "460923", // SKU
  "Specialized Chamonix", // name
  "Helmets", // category
  200, // price
  1, // quantity
]);

// track order
_paq.push([
  "trackEcommerceOrder",
  "43967392", // order ID
  5250, // grand total (value + tax + discount + shipping)
  5200, // sub total (value + tax + discount)
  970, // tax
  150, // shipping
  100, // discount
]);
```

JS (direct)

```
var jstc = Piwik.getTracker(
  "https://example.com/",
  "45e07cbf-c8b3-42f3-a6d6-a5a176f623ef"
);
jstc.trackEcommerceOrder(
  "584340", // SKU
  "Specialized Stumpjumper", // name
  "Mountain bike", // category
  5000, // price
  1 // quantity
);

jstc.trackEcommerceOrder(
  "460923", // SKU
  "Specialized Chamonix", // name
  "Helmets", // category
  200, // price
  1 // quantity
);

// track order
jstc.trackEcommerceOrder(
  "43967392", // order ID
  5250, // grand total (value + tax + discount + shipping)
  5200, // sub total (value + tax + discount)
  970, // tax
  150, // shipping
  100 // discount
);
```

Angular

```

```

React

```
```

Notes

- The cart with added items is not stored in the browser storage. Make sure that you add all items again after the page reloads.
- If a product with the same SKU is already in the cart, it'll be removed and replaced with the one added with the addEcommerceItem() method.
- This method doesn't send any data to Piwik PRO. It just creates a cart. You can use the trackEcommerceCartUpdate() or trackEcommerceOrder() method to send cart data to Piwik PRO.

Related methods

- removeEcommerceItem()
- clearEcommerceCart()
- getEcommerceItems()
- setEcommerceView()
- trackEcommerceCartUpdate()
- trackEcommerceOrder()

addListener()

The **addListener()** method adds automatic link tracking to an HTML element. You can use it to track links added to a document after a page load.

Syntax

JS

```
addListener(domElement)
```

Parameters

domElement (DOM element, required)
The element that you want to track as a link.

Examples

To add a link to this element #dynamically-added-link:

JS (queue)

```
_paq.push(["addListener", document.querySelector("#dynamically-added-link")]);
```

JS (direct)

```
var jstc = Piwik.getTracker(
  "https://example.com/",
  "45e07cbf-c8b3-42f3-a6d6-a5a176f623ef"
);
jstc.addListener(document.querySelector("#dynamically-added-link"));
```

addTracker()

The **addTracker()** method creates a new tracker's instance with a new tracking endpoint.

Syntax

JS

```
addTracker(trackerUrl, siteId)
```

Parameters

trackerURL (string, required)
A tracker URL (tracking endpoint).

siteID (string, required)
A site or app ID in Piwik PRO where you want to send data. (Where to find it?)

Returns

A tracker's instance.
Format: Example:
Type: object

Examples

To create a new tracker's instance:

JS (queue)

```
_paq.push(["addTracker", "https://example.piwik.pro/ppms.php", "45e07cbf-c8b3-42f3-
↪a6d6-a5a176f623ef"]);
```

JS (direct)

```
var jstc = Piwik.getTracker(
  "https://example.com/",
  "45e07cbf-c8b3-42f3-a6d6-a5a176f623ef"
);
jstc.setTrackerUrl("https://example.piwik.pro/ppms.php");
```

Related methods

- setTrackerUrl()
- getTrackerUrl()

appendToTrackingUrl()

The **appendToTrackingUrl()** method adds a query string to the tracking request.

Syntax

JS

```
appendToTrackingUrl(appendToUrl)
```

Parameters

appendToUrl (string, required)
A custom query string that'll be attached to each tracking request. Parameter names and values need to be URL encoded. Example: lat=140&long=100

Examples

To add lat=140&long=100 parameter to each request:

JS (queue)

```
_paq.push(["appendToTrackingUrl", "lat=140&long=100"]);
```

JS (direct)

```
var jstc = Piwik.getTracker(
  "https://example.com/",
  "45e07cbf-c8b3-42f3-a6d6-a5a176f623ef"
);
jstc.appendToTrackingUrl("lat=140&long=100");
```

clearEcommerceCart()

The **clearEcommerceCart()** method removes all items added to the cart.

Syntax

JS

```
clearEcommerceCart()
```

Angular

```
clearEcommerceCart()
```

React

```
clearEcommerceCart()
```

Examples

To remove all items from the cart:

JS (queue)

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

JS (direct)

```
var jstc = Piwik.getTracker(
  "https://example.com/",
  "45e07cbf-c8b3-42f3-a6d6-a5a176f623ef"
);
jstc.clearEcommerceCart();
```

Angular

```

```

React

```

```

Notes

- The cart with cleared items is not stored in the browser storage. Make sure that you add or clear all items again after the page reloads.
- This method doesn't send any data to Piwik PRO. It just updates the cart. You can use the trackEcommerceCartUpdate() or trackEcommerceOrder() method to send cart data to Piwik PRO.

Related methods

- addEcommerceItem()
- removeEcommerceItem()
- getEcommerceItems()
- setEcommerceView()
- trackEcommerceCartUpdate()
- trackEcommerceOrder()

customCrossDomainLinkDecorator()

The **customCrossDomainLinkDecorator()** method sets a custom query parameter that holds a visitor ID when domains are linked. The default parameter is `pk_vid`.

This method works when you use enableCrossDomainLinking().

Syntax

JS

```
customCrossDomainLinkDecorator(urlDecorator)
```

Parameters

urlDecorator (function, required)
Defines parameters that are used to hold a visitor ID when domains are linked.

urlDecorator (url, value, name)
The urlDecorator() method accepts a URL, value and name, and returns a decorated URL.

url (string, required)
A page URL.

value (string, required)
The visitor ID that should be passed via the URL.

name (string, required)
The name of the visitor ID used in Piwik PRO. It can be customized.

Returns

A decorated URL or null.
Format: Example: https://example.com?pk_vid=36
Type: string | null

Examples

To xxxxxxxxxxx:

JS (queue)

```
_paq.push(["customCrossDomainLinkDecorator", function (url, value, name) {
    var parsedUrl = new URL(url);
    parsedUrl.searchParams.append(name, value);
    return parsedUrl.href;
}]);
```

JS (direct)

```
var jstc = Piwik.getTracker(
  "https://example.com/",
  "45e07cbf-c8b3-42f3-a6d6-a5a176f623ef"
);
jstc.customCrossDomainLinkDecorator(function (url, value, name) {
    var parsedUrl = new URL(url);
    parsedUrl.searchParams.append(name, value);
    return parsedUrl.href;
}]);
```

Related methods

- enableCrossDomainLinking()
- disableCrossDomainLinking()
- isCrossDomainLinkingEnabled()
- setCrossDomainLinkingTimeout()
- getCrossDomainLinkingUrlParameter()
- customCrossDomainLinkVisitorIdGetter()

customCrossDomainLinkVisitorIdGetter()

The **customCrossDomainLinkVisitorIdGetter()** method gets a visitor ID from a page URL if customCrossDomainLinkDecorator() was set. The visitor ID is held in a query parameter and passed between domains when they are linked with enableCrossDomainLinking().

Syntax

JS

```
customCrossDomainLinkVisitorIdGetter(urlParser)
```

Parameters

urlParser (function, required)
Extracts a visitor ID from a page URL.

urlParser (url, name)
The urlParser() method accepts a URL and name, and returns a visitor ID.

url (string, required)
A page URL.

name (string, required)
A parameter name that holds a visitor ID.

Returns

A visitor ID extracted from a page URL.
Format: Example: c52b5d0969220761
Type: string

Example

To do something:

JS (queue)

```
_paq.push(["customCrossDomainLinkVisitorIdGetter", function (url, name) {
    return (new URL(url)).searchParams.get(name) || "";
}]);
```

JS (direct)

```
var jstc = Piwik.getTracker(
  "https://example.com/",
  "45e07cbf-c8b3-42f3-a6d6-a5a176f623ef"
);
jstc.customCrossDomainLinkVisitorIdGetter(function (url, name) {
    return (new URL(url)).searchParams.get(name) || "";
});
```

Related methods

- enableCrossDomainLinking()
- disableCrossDomainLinking()
- isCrossDomainLinkingEnabled()
- setCrossDomainLinkingTimeout()
- getCrossDomainLinkingUrlParameter()
- customCrossDomainLinkDecorator()

deleteCookies()

The **deleteCookies()** method deletes the existing visitor cookie (`_pk_id.*`) and session cookie (`_pk_ses.*`) that are responsible for recognizing visitors and their sessions. The cookies will be deleted with the next page view.

Syntax

JS

```
deleteCookies()
```

Examples

To delete visitor cookies:

JS (queue)

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

JS (direct)

```
var jstc = Piwik.getTracker(
  "https://example.com/",
  "45e07cbf-c8b3-42f3-a6d6-a5a176f623ef"
);
console.log(jstc.hasCookies());
```

Related methods

- enableCookies()
- disableCookies()
- hasCookies()
- setCookieNamePrefix()
- setCookieDomain()
- getCookieDomain()
- setCookiePath()
- getCookiePath()
- setSecureCookie()
- setVisitorCookieTimeout()
- getConfigVisitorCookieTimeout()
- setReferralCookieTimeout()
- setSessionCookieTimeout()
- getSessionCookieTimeout()
- setVisitorIdCookie()

deleteCustomDimension()

The **deleteCustomDimension()** method removes a custom dimension.

Syntax

JS

```
getCustomDimensionValue(customDimensionID)
```

Angular

```
deleteCustomDimension(customDimensionId: string)
```

React

```
deleteCustomDimension(customDimensionId: string)
```

Parameters

customDimensionID (number, required)
An ID of the custom dimension.

Examples

To remove a custom dimension:

JS (queue)

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

JS (direct)

```
var jstc = Piwik.getTracker(
  "https://example.com/",
  "45e07cbf-c8b3-42f3-a6d6-a5a176f623ef"
);
jstc.deleteCustomDimension(1);
```

Angular

```

```

React

```

```

Related methods

- setCustomDimensionValue()
- getCustomDimensionValue()

deleteCustomVariable()

Deprecated since version 0.0.0: This method is no longer recommended. Use the setCustomDimensionValue() and deleteCustomDimension() method instead.

The **deleteCustomDimension()** method removes a custom variable.

Syntax

JS

```
deleteCustomVariable(index[, scope])
```

Parameters

index (number, required)
Index from 1 to 5 where the variable is stored.

scope (string, optional)
Scope of the variable: "visit" or "page". The default value is "visit".

Examples

To remove a custom variable:

JS (queue)

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

JS (direct)

```
var jstc = Piwik.getTracker(
  "https://example.com/",
  "45e07cbf-c8b3-42f3-a6d6-a5a176f623ef"
);
jstc.deleteCustomVariable(1, "page");
```

disableCookies()

disableCrossDomainLinking()

disableHeartBeatTimer()

disableLinkTracking()

disablePerformanceTracking()

discardHashTag()

enableCookies()

enableCrossDomainLinking()

enableHeartBeatTimer()

enableJSErrorTracking()

enableLinkTracking()

getConfigDownloadExtensions()

getConfigIdPageView()

getConfigVisitorCookieTimeout()

getCookieDomain()

getCookiePath()

getCrossDomainLinkingUrlParameter()

getCurrentUrl()

getCustomDimension() •

2.2 Consent Manager

2.2.1 Getting started

Our JavaScript library has methods that let you use Consent Manager. You can use these methods in JavaScript.

In short, our JavaScript library lets you:

- Get compliance types
- Get and set complience settings
- Send a data request
- Open a consent form
- Track consent stats

Next steps

- *Plain JavaScript*
- *Methods*

2.2.2 Plain JavaScript

Here are some guidelines on how to use our Consent Manager JS API in Java Script.

Installation

Our JavaScript library can be used only after you installed our container's code (or only a tracking code) on your site. The code creates a `<script>` tag that asynchronously loads the JavaScript library in the website's body section.

If you haven't installed the code yet, you can find it directly in Piwik PRO in **Menu > Administration > Sites & apps > Installation**.

For more, see our installation guides:

- Install a container (with a tracking code)
- Google Tag Manager: install a container (with a tracking code)
- Google Tag Manager: install only a tracking code
- Instapage: install a container (with a tracking code)
- No Piwik PRO Tag Manager: install a tracking code
- Squarespace: install a container (with a tracking code)
- WordPress: install a container (with a tracking code)

Methods used for calls

In JavaScript, our methods can be called in this way:

- **JS (queue)**: After installing our container's code, it'll create the `_paq` object (a queue). You can use the `ppms.cm.api` method to add methods to the queue. Our tracker will then access and proceed these methods.

ppms.cm.api

The `ppms.cm.api` method adds methods to the `_paq` object (a queue). The methods are called after the container's code (or a tracking code) loads on a page. They are called synchronously (one by one).

Syntax

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

Parameters

command (string, required)
Command name

args (optional)
Command arguments. The number of arguments and their function depend on command.

Returns

Commands are expected to be run asynchronously and return no value. Type: undefined

Notes

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

Custom consent form

Our Consent Manager JS API lets you build a custom consent form in place of the default one.

To turn on custom consent form, follow these steps:

1. Log in to Piwik PRO.
2. Go to **Menu**.
3. Go to **Administration > Sites & apps**.
4. Navigate to **Privacy**.
5. Turn on **Ask visitors for consent**.
6. Turn on **Use a custom consent form**.

2.2.3 Methods

Here's a list of all available JS methods for Consent Manager in Piwik PRO.

getComplianceSettings()

Returns current privacy settings. Use this command to get visitor's decisions. This command returns an empty object if there were no decisions registered yet.

Syntax

```
ppms.cm.api('getComplianceSettings', onFulfilled, onRejected);
```

Parameters

onFulfilled(settings) (function, required)
The fulfillment handler callback (called with result).

> **settings** (object, required)
> The consent setting object
> Example: `{consents: {analytics: {status: -1, updatedAt: '2018-07-03T12:18:19.957Z'}}}`

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

- `-1`: A vistior didn't interact. For example, they closed a consent popup without any decision.
- `0`: A visitor rejected consent.
- `1`: A vistior approved consent.

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

error (string | object, required)
Error code or exception

Examples

See custom consent form example.

getComplianceTypes()

Fetches a list of consent types for the current setup. For the consent type to appear in the output, at least one tag must have it set.

Syntax

```
ppms.cm.api('getComplianceTypes', onFulfilled, onRejected);
```

Parameters

onFulfilled(types) (function, required)
The fulfillment handler callback (called with result).

types (string, required)
Array of consent types. Example: `["remarketing", "analytics"]`

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

error (string | object, required)
Error code or exception

Examples

See custom consent form example.

getNewComplianceTypes()

Fetches a list of the consent types which a visitor did not see yet.

Syntax

```
ppms.cm.api('getNewComplianceTypes', onFulfilled, onRejected);
```

Parameters

onFulfilled(types) (function, required)
The fulfillment handler callback (called with result).

> **types** (string, required)
> Array of consent types. Example: `["remarketing", "analytics"]`

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

> **error** (string | object, required)
> Error code or exception

Examples

See custom consent form example.

openConsentForm()

Command used to open consent form. Works only for built-in consent forms, it will not do anything if Custom consent form mode is enabled.

Syntax

```
ppms.cm.api('openConsentForm', onFulfilled, onRejected);
```

Parameters

onFulfilled(popupId, consentTypes, consents)
The fulfillment handler callback

> **popupId** (string)

ID of the consent popup. Example:
`"ppms_cm_consent_popup_30a851b6-6bf4-45f9-9a53-583401bb5d60"`

consentTypes (array)
Array of consent types. Example: `["analytics", "conversion_tracking", "remarketing"]`

consents (string)
Array list of all given consents. Example: `["analytics", "remarketing"]`

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

error (string | object, required)
Error code or exception

Examples

See custom consent form example.

sendDataRequest()

Command that sends a Data subject request to the Consent Manager.

Syntax

```
ppms.cm.api('sendDataRequest', request, onFulfilled, onRejected);
```

Parameters

request (object, 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() (function, required)
The fulfillment handler callback (called with result).

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

> **error** (string | object, required)
> Error code or exception

Examples

See custom consent form example.

setComplianceSettings()

Set compliance settings based on visitor's decisions. Use this command to save visitor's consent choices from the consent form. Consent Manager forces a page view after the command is invoked, so all tags requiring certain choices will be fired immediately after the consent is given.

Syntax

```
ppms.cm.api('setComplianceSettings', settings, onFulfilled, onRejected);
```

Parameters

settings (object, required)
The consent settings object. Example: `{consents: {analytics: {status: 1}}}`
Where `consent.analytics` is consent type and status indicate:

- `0`: A visitor rejected consent.
- `1`: A visitor approved consent.

onFulfilled() (function, required)
The fulfillment handler callback (called with result).

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

> **error** (string | object, required)
> Error code or exception

Examples

See custom consent form example.

setInitialComplianceSettings()

Sets initial compliance settings (no decision signal for each consent type) in the cookie. Use this command to save "no decision" for the available consent types, to further know that a visitor has seen the form. Result from getNewComplianceTypes() method can be passed directly.

Syntax

```
ppms.cm.api('setInitialComplianceSettings', settings, onFulfilled, onRejected);
```

Parameters

settings (object, required)
The consent settings object. Example: `{consents: ['analytics']}` or `['analytics']`.

onFulfilled() (function, required)
The fulfillment handler callback (called with result).

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

> **error** (string | object, required)
> Error code or exception

Examples

See custom consent form example.

trackAgreeToAllClick()

Command used to track clicks on the `Agree to all` button.

Syntax

```
ppms.cm.api('trackAgreeToAllClick', onFulfilled, onRejected);
```

Parameters

onFulfilled() (function, required)
The fulfillment handler callback (called with result).

onRejected(error)

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

error (string | object, required)
Error code or exception

Examples

See custom consent form example.

trackCloseButtonClick()

Command used to track clicks on the close button (`X`).

Syntax

```
ppms.cm.api('trackCloseButtonClick', onFulfilled, onRejected);
```

Parameters

onFulfilled() (function, required)
The fulfillment handler callback (called with result).

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

error (string | object, required)
Error code or exception

Examples

See custom consent form example.

trackMainFormView()

Command used to track Consent Form main view (automatic view, when user enters the website for the first time).

Syntax

```
ppms.cm.api('trackMainFormView', onFulfilled, onRejected);
```

Parameters

onFulfilled() (function, required)
The fulfillment handler callback (called with result).

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

> **error** (string | object, required)
> Error code or exception

Examples

See custom consent form example.

trackPrivacyPolicyLinkView()

Command used to track Consent Form view caused by clicking on Privacy Policy Link.

Syntax

```
ppms.cm.api('trackPrivacyPolicyLinkView', onFulfilled, onRejected);
```

Parameters

onFulfilled() (function, required)
The fulfillment handler callback (called with result).

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

> **error** (string | object, required)
> Error code or exception

Examples

See custom consent form example.

trackRejectAllClick()

Command used to track clicks on the `Reject all` button.

Syntax

```
ppms.cm.api('trackRejectAllClick', onFulfilled, onRejected);
```

Parameters

onFulfilled() (function, required)
The fulfillment handler callback (called with result).

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

> **error** (string | object, required)
> Error code or exception

Examples

See custom consent form example.

trackReminderWidgetView()

Command used to track Consent Form view caused by clicking on Reminder Widget.

Syntax

```
ppms.cm.api('trackReminderWidgetView', onFulfilled, onRejected);
```

Parameters

onFulfilled() (function, required)
The fulfillment handler callback (called with result).

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

> **error** (string | object, required)
> Error code or exception

Examples

See custom consent form example.

trackSaveChoicesClick()

Command used to track clicks on the `Save choices` button.

Syntax

```
ppms.cm.api('trackSaveChoicesClick', onFulfilled, onRejected);
```

Parameters

onFulfilled() (function, required)
The fulfillment handler callback (called with result).

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

> **error** (string | object, required)
> Error code or exception

Examples

See custom consent form example.

CHAPTER 3

SDKs

Our SDKs let you collect user data from mobile apps built for Android, Flutter, React Native and iOS. With over 30 built-in methods, you can easily track screen views, goals, ecommerce orders and more.

See more

3.1 Android SDK

3.1.1 Getting started

Our SDK for Android lets you collect user data from Android mobile apps. It contains over 30 methods that make it easy to track screen views, goals, ecommerce orders and more.

To get started, you need to set up your account in Piwik PRO, install our library and set up the tracker.

Set up Piwik PRO

Before you install our library for Android, you need to set up Piwik PRO. Here's what you need to do:

1. Log in to **Piwik PRO**.
2. Go to **Menu > Administration**.
3. Navigate to **Sites & apps**.
4. Click **Add a site or app**.
5. Type the app name and address and click **Save**.
6. Set the time zone and currency.
7. Note the site/app ID. The ID is below the app name. Example: `00000000-0000-0000-0000-000000000000`.
8. Note your account address. Example: `https://example.piwik.pro`.

Install the library

To install the library, follow these steps:

1. Add the JitPack repository to your root `build.gradle` file at the end of repositories:

```
allprojects {
repositories {
  ...
  maven { url 'https://jitpack.io' }
}
}
```

2. Add the dependency to the application module `build.gradle` file:

```
dependencies {
implementation 'pro.piwik:sdk-framework-android:VERSION'
}
```

Note: Replace `VERSION` with the latest release name. Example: `1.1.8`. (Where to find it?)

Set up the tracker

To set up the Piwik PRO tracker, you can use two methods: (1) create and manage the tracker in the Application class or (2) manage the tracker on your own.

Method #1

We recommend using this method for most cases. It forces the implementation of just one abstract method.

To set up the Piwik PRO tracker, follow these steps:

1. Extend the `PiwikApplication` class with your Android Application class. Use your account address (Example: `https://example.piwik.pro/`) and the site/app ID (Where to find it?)

```
public class YourApplication extends PiwikApplication{
    @Override
    public TrackerConfig onCreateTrackerConfig() {
        return TrackerConfig.createDefault("account-address", "site-id");
    }
}
```

Tip: See our demo app where we used this method.

2. Share the `Tracker` instance across your app. The `Tracker` is now thread-safe.

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

3. Done! Now your app can use Piwik PRO SDK.

4. We recommend using the `TrackHelper` class to track events. For tracking each event with `TrackHelper`, you will need to pass the `Tracker` instance.

```
Tracker tracker = ((PiwikApplication) getApplication()).getTracker();
TrackHelper.track().screen("Main screen").with(tracker);
```

Note: The `TrackerHelper` class has methods for all common actions, which can be chained to facilitate the correct order and use. Combine it with the IDE autocompletion and using the SDK will be even easier.

Method #2

To set up the Piwik PRO tracker, follow these steps:

1. Manage the tracker on your own. Use your account address (Example: `https://example.piwik.pro/`) and the site/app ID (Where to find it?).

```
public class YourApplication extends Application {
    private Tracker tracker;
    public synchronized Tracker getTracker() {
        if (tracker == null) tracker = Piwik.getInstance(this).newTracker(new␣
↪TrackerConfig(""account-address", "site-id", "Default Tracker"));
        return tracker;
    }
}
```

Note: We recommend using just one tracker instance for your app. Otherwise, you may end up with over-counted metrics.

2. Share the `Tracker` instance across your app. The `Tracker` is now thread-safe.

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

3. Done! Now your app can use Piwik PRO SDK.
4. We recommend using the `TrackHelper` class to track events. For tracking each event with `TrackHelper`, you will need to pass the `Tracker` instance.

```
Tracker tracker = ((YourApplication) getApplication()).getTracker();
TrackHelper.track().screen("Main screen").with(tracker);
```

Note: The `TrackerHelper` class has methods for all common actions, which can be chained to facilitate the correct order and use. Combine it with the IDE autocompletion and using the SDK will be even easier.

Kotlin

Our SDK is written in Java but can be used with Kotlin. If you refer to any of our methods in Kotlin, they will automatically appear as Kotlin syntax.

Here's an example of the **track().screen()** method in both languages:

Java

```
Tracker tracker = ((PiwikApplication) getApplication()).getTracker();
TrackHelper.track().screen("path").title("title").with(tracker);
```

Kotlin

```
val tracker: Tracker = (application as PiwikApplication).tracker
TrackHelper.track().screen("path").title("title").with(tracker)
```

Tip: For more on calling Java from Kotlin, see this article.

3.1.2 Methods

Here's a list of methods you can use with our SDK for Android:

OnCheckAudienceMembership()

Deprecated since version 16.0.0: This method is no longer recommended. Audience Manager is no longer available in the latest product version.

The **OnCheckAudienceMembership()** method checks if a user belongs to a specific audience.

Syntax

Java

```
getTracker().checkAudienceMembership(audienceId, new Tracker.
↪OnCheckAudienceMembership() {
  @Override
  public void onChecked(boolean isMember) {
      // handle result
  }

  @Override
  public void onError(String errorData) {
      // handle error
  }
});
```

Kotlin

```
tracker.checkAudienceMembership(
  binding.audienceId.text.toString(),
  object : OnCheckAudienceMembership {
    override fun onChecked(isMember: Boolean) {
      // handle result
    }

    override fun onError(errorData: String) {
    var errorData: String? = errorData
    errorData = if (TextUtils.isEmpty(errorData)) "Network error" else errorData
      // handle error
    }
  })
```

Parameters

audienceId (string, required)
The ID of the audience you want to check. You can find it in Audience Manager > Audiences.

OnCheckAudienceMembership() (required)
The callback to handle a request result. The call is asynchronous. It has two methods `void onChecked(boolean isMember)` and `void onError(String errorData)`.

isMember (boolean, output)
Whether the user belongs to a specific audience. True: Belongs. False: Doesn't belong.

errorData (output)
The error string. If an error occurs, only this method will be called.

Examples

To check if a user belongs to an audience:

Java

```
getTracker().checkAudienceMembership(audienceId, new Tracker.
↪OnCheckAudienceMembership() {
  @Override
  public void onChecked(boolean isMember) {
      // handle result
  }

  @Override
  public void onError(String errorData) {
      // handle error
  }
});
```

Kotlin

```
tracker.checkAudienceMembership(
  binding.audienceId.text.toString(),
  object : OnCheckAudienceMembership {
    override fun onChecked(isMember: Boolean) {
      // handle result
    }

    override fun onError(errorData: String) {
    var errorData: String? = errorData
    errorData = if (TextUtils.isEmpty(errorData)) "Network error" else errorData
      // handle error
    }
  })
```

Related methods

- *audienceManagerSetProfileAttribute()*
- *audienceManagerGetProfileAttributes()*

audienceManagerGetProfileAttributes()

Deprecated since version 16.0.0: This method is no longer recommended. Audience Manager is no longer available in the latest product version.

The **audienceManagerGetProfileAttributes()** method returns profile attributes. You can only access attributes that heve been granted access.

Syntax

Java

```
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
    }
});
```

Kotlin

```
tracker.audienceManagerGetProfileAttributes(object : OnGetProfileAttributes {
  override fun onAttributesReceived(attributes: Map<String, String>) {
    // handle result
  }

  override fun onError(errorData: String) {
    var errorData: String? = errorData
    errorData = if (TextUtils.isEmpty(errorData)) "Network error" else errorData
    // handle error
  }
})
```

Parameters

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

attributes (output)
The dictionary of key-value pairs. Key: attribute name. Value: attribute value.

errorData (output)
The error string. If an error occurs, only this method will be called.

Examples

To get profile attributes:

Java

```
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
    }
});
```

Kotlin

```
tracker.audienceManagerGetProfileAttributes(object : OnGetProfileAttributes {
  override fun onAttributesReceived(attributes: Map<String, String>) {
    // handle result
  }

  override fun onError(errorData: String) {
    var errorData: String? = errorData
    errorData = if (TextUtils.isEmpty(errorData)) "Network error" else errorData
    // handle error
  }
})
```

Notes

- You can only access attributes that have been granted access.

Related methods

- *audienceManagerSetProfileAttribute()*
- *OnCheckAudienceMembership()*

audienceManagerSetProfileAttribute()

Deprecated since version 16.0.0: This method is no longer recommended. Audience Manager is no longer available in the latest product version.

The **audienceManagerSetProfileAttribute()** method sets profile attributes in Audience Manager. The attributes can be sent to Audience Manager with a screen view or other event.

Attributes are all kinds of information about a user that help you build audiences for marketing and advertising purposes.

Syntax

Java

```
TrackHelper.track()
  .audienceManagerSetProfileAttribute("name", "value")
  .add("name", "value")
  .with(getTracker());
```

Kotlin

```
TrackHelper.track()
  .audienceManagerSetProfileAttribute("name", "value")
  .add("name", "value")
  .with(tracker)
```

Parameters

name (string, required)
The name of the profile attribute. Example: plan type.

value (string, required)
The value of the profile attribute. Example: premium.

add() (chain method)
Other attributes that you want to send with the same event.

Examples

To set the attribute `plan type` with the value `premium` and send it with `.with(getTracker())` to Audience Manager:

Java

```
TrackHelper.track()
  .audienceManagerSetProfileAttribute("plan type", "premium")
  .add("", "")
  .with(getTracker());
```

Kotlin

```
TrackHelper.track()
  .audienceManagerSetProfileAttribute(("plan type"), "premium")
  .add("", "")
  .with(tracker)
```

Notes

- Each event always sends the following attributes: Site or app ID, Visitor ID and Device ID.
- If setAnonymizationState(false) is set and User ID and Email are set, each event will also send User ID and Email.
- You can see all added attributes in Audience Manager > Profile.

Related methods

- *audienceManagerGetProfileAttributes()*
- *OnCheckAudienceMembership()*

getDeviceId()

The **getDeviceId()** method returns the device ID set for a specific user.

Syntax

Java

```
getTracker().getDeviceId();
```

Kotlin

```
tracker.deviceId
```

Returns

The unique device ID
Format: Example: abcd123e-a123-bcFG-d123
Type: String

Examples

To get a device ID:

Java

```
getTracker().getDeviceId();
```

Kotlin

```
tracker.deviceId
```

Related methods

- *setTrackDeviceId()*
- *setDeviceId()*

isAnonymizationOn()

The **isAnonymizationOn()** method checks whether a user is marked as anonymous or non-anonymous.

Syntax

Java

```
((PiwikApplication) getApplication()).getTracker().isAnonymizationOn();
```

Kotlin

```
(application as PiwikApplication).tracker.isAnonymizationOn
```

Returns

Whether a user is marked as anonymous or non-anonymous.
Format: True: is anonymous. False: is non-anonymous.
Type: Boolean

Examples

To check whether a specific user is anonymous or non-anonymous:

Java

```
((PiwikApplication) getApplication()).getTracker().isAnonymizationOn();
```

Kotlin

```
(application as PiwikApplication).tracker.isAnonymizationOn
```

Related methods

- *setAnonymizationState()*

items.addItem()

The **items.addItem()** method adds a product to the cart.

Syntax

Java

```
EcommerceItems items = new EcommerceItems();

items.addItem(new EcommerceItems
  .Item("productSKU")
  .name("productName")
  .category("productCategory")
  .price(productPrice)
  .quantity(productQuantity));
```

Kotlin

```
var items: EcommerceItems = EcommerceItems()

items.addItem(EcommerceItems
  .Item("productSKU")
  .name("productName")
  .category("productCategory")
  .price(productPrice)
  .quantity(productQuantity))
```

Parameters

productSKU (string, required)
The stock-keeping unit of the added product.

productName (string, optional)
The name of the added product.

productCategory (string | array<string>, optional)
The category of the added product. It can be an array of up to 5 categories.

productPrice (number, optional)
The price of the added product.

productQuantity (number, optional)
The number of added items.

Examples

To track a confirmed order:

Java

```
Tracker tracker = ((YourApplication) getApplication()).getTracker();
EcommerceItems items = new EcommerceItems();

// register all purchased items
// EcommerceItems.Item("<SKU>").name("<name>").category("<category>").price(<price>).
↪quantity(<quantity>)

items.addItem(new EcommerceItems
  .Item("584340")
  .name("Specialized Stumpjumper")
  .category("Mountain bike")
  .price(500000)
  .quantity(1));

items.addItem(new EcommerceItems
  .Item("460923")
```

(continues on next page)

(continued from previous page)

```
  .name("Specialized Chamonix")
  .category("Helmets")
  .price(20000)
  .quantity(1));

// track order

TrackHelper.track()
  .order("43967392", 525000)
  .subTotal(520000)
  .tax(97000)
  .shipping(15000)
  .discount(10000)
  .items(items)
  .with(tracker);
```

Kotlin

```
val tracker: Tracker = (application as PiwikApplication).tracker
var items: EcommerceItems = EcommerceItems()

// register all purchased items
// EcommerceItems.Item("<SKU>").name("<name>").category("<category>").price(<price>).
↪quantity(<quantity>)

items.addItem(EcommerceItems
  .Item("584340")
  .name("Specialized Stumpjumper")
  .category("Mountain bike")
  .price(500000)
  .quantity(1))

items.addItem(EcommerceItems
  .Item("460923")
  .name("Specialized Chamonix")
  .category("Helmets")
  .price(20000)
  .quantity(1))

// track order

TrackHelper.track()
  .order("43967392", 525000)
  .subTotal(520000)
  .tax(97000)
  .shipping(15000)
  .discount(10000)
  .items(items)
  .with(tracker)
```

Notes

- The cart with added items is not stored in local storage. Make sure to add all items again after the page reloads.
- If a product with the same SKU is already in the cart, it'll be removed and replaced with the product added with the items.addItem() method.

- This method doesn't send any data to Piwik PRO. It just creates a cart. You can use the track().order() method to send cart data to Piwik PRO.

Related methods

- *track().order()*

setAnonymizationState()

The **setAnonymizationState()** method marks a user as anonymous or non-anonymous. If set to anonymous, the user's IP address, location information (only the country is known), user ID and device ID are not collected. Every time the application is started, a new visitor ID is generated for anonymous users.

The setAnonymizationState(true) is set by default. This means that each user is anonymous by default.

Syntax

Java

```
((PiwikApplication) getApplication()).getTracker().setAnonymizationState(isAnonymous);
```

Kotlin

```
(application as PiwikApplication).tracker.setAnonymizationState(
  isAnonymous
)
```

Parameters

isAnonymous (boolean, required)
Weather a user is anonymous or non-anonymous. True: anonymous. False: non-anonymous.

Examples

To mark a visitor as non-anonymous:

Java

```
((PiwikApplication) getApplication()).getTracker().setAnonymizationState(false);
```

Kotlin

```
(application as PiwikApplication).tracker.setAnonymizationState(
  false
)
```

Related methods

- *isAnonymizationOn()*

setDeviceId()

The **setDeviceId()** method sets a custom device ID.

Syntax

Java

```
getTracker().setDeviceId(deviceID)
```

Kotlin

```
tracker.deviceId = deviceID
```

Parameters

deviceID (string, optional)
A custom device ID. If the value is not set, an automatic value is generated.

Examples

To set a device ID to `ABC123`:

Java

```
getTracker().setDeviceId(ABC123)
```

Kotlin

```
tracker.deviceId = ABC123
```

Notes

- The device ID won't be sent if setAnonymizationState(true) is set.

Related methods

- *setTrackDeviceId()*
- *getDeviceId()*

setDispatchInterval()

The **setDispatchInterval()** method sets a custom dispatch interval time. Tracked events are temporarily stored in a queue and dispatched in batches. The default dispatch interval time is 30 seconds (3000 milliseconds) – batches are sent every 30 seconds.

Syntax

Java

```
getTracker().setDispatchInterval(milliseconds)
```

Kotlin

```
tracker.dispatchInterval = milliseconds
```

Parameters

milliseconds (number, required)
The interval time (in milliseconds) for dispatching tracked events. If 0 milliseconds, events will be sent right away. If -1 milliseconds, events won't be sent automatically and can be send manually.

Examples

To set the dispatch interval time to 60 seconds (60*1000 milliseconds):

Java

```
getTracker().setDispatchInterval(60 * 1000)
```

Kotlin

```
tracker.dispatchInterval = 60 * 1000
```

To block sending events automatically and send them manually:

Java

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

Kotlin

```
val tracker: Tracker = (application as PiwikApplication).getTracker()
tracker.dispatchInterval = -1
// Catch and track exception
try {
  cartItems = tracker.getCartItems()
} catch (e: java.lang.Exception) {
  TrackHelper.track().exception(e).description(e.message)
  tracker.dispatch()
  cartItems = null
}
```

Notes

- If more than one event is queued, the data is sent in bulk using the POST method with a JSON payload.

setDryRunTarget()

The **setDryRunTarget()** method sets a dry-run flag and lets you test and debug tracking. The dry-run flag prevents data from being sent to Piwik PRO and instead prints it to the console.

Syntax

Java

```
getTracker().setDryRunTarget(dryRunTarget);
```

Kotlin

```
tracker.dryRunTarget = Collections.synchronizedList(dryRunTarget)
```

Parameters

dryRunTarget (Collection, required)
The data structure to which the data shoud be sent. Type: List<Packet>. Set it to null to disable the dry-run flag.

Examples

To set the dry-run flag:

Java

```
getTracker().setDryRunTarget(Collections.synchronizedList(new ArrayList<Packet>()));
```

Kotlin

```
tracker.dryRunTarget = Collections.synchronizedList(Collections.
↪synchronizedList(ArrayList()))
```

setIncludeDefaultCustomVars()

The **setIncludeDefaultCustomVars()** method turns on or off fetching platform, OS and app version from the tracker instance. It is turned on by default.

Syntax

Java

```
getTracker().setIncludeDefaultCustomVars(isFetched);
```

Kotlin

```
tracker.includeDefaultCustomVars = isFetched
```

Parameters

isFetched (boolean, required)
Whether platform, OS and app version are fetched from the tracker instance. True: is fetched. False: is not fetched.

Examples

To turn off automatic fetching of platform, OS and app version from the tracker instance:

Java

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

Kotlin

```
tracker.includeDefaultCustomVars = false
```

Notes

- If setIncludeDefaultCustomVars(true) is set, indexes 1-3 are used to track the platform, OS and app version as custom variables.

Related methods

- *track().dimension()*
- *track().variable()*
- *track().visitVariables()*

setOfflineCacheAge()

The **setOfflineCacheAge()** method sets the time limit for storing events in local storage. The default value is 24 hours (24 * 60 * 60 * 1000 milliseconds).

Syntax

Java

```
tracker.setOfflineCacheAge(milliseconds);
```

Kotlin

```
tracker.offlineCacheAge = milliseconds
```

Parameters

milliseconds (number, required)
The time (in milliseconds) after which events are removed from local storage. Default value: 24 hours (24 * 60 * 60 * 1000 milliseconds). If 0 is set, events are stored forever (unlimited time). If -1 is set, event storing is turned off.

Examples

To set the time limit to 12 hours (12 * 60 * 60 * 1000 milliseconds):

Java

```
tracker.setOfflineCacheAge(12 * 60 * 60 * 1000);
```

Kotlin

```
tracker.offlineCacheAge = 12 * 60 * 60 * 1000
```

Related methods

- *setOfflineCacheSize()*

setOfflineCacheSize()

The **setOfflineCacheSize()** method sets the size limit for storing events in local storage. The default value is 4 Mb (4*1024*1024 bytes).

Syntax

Java

```
tracker.setOfflineCacheSize(bytes);
```

Kotlin

```
tracker.offlineCacheSize = bytes
```

Parameters

bytes (number, required)
The size limit (in bytes) for storing events in local storage. Default value: 4 Mb (4*1024*1024 bytes). If 0 is set, the size is unlimited.

Examples

To set the size limit to 2 Mb (2 * 1024 * 1024 bytes):

Java

```
tracker.setOfflineCacheSize(2 * 1024 * 1024);
```

Kotlin

```
tracker.offlineCacheSize = 2 * 1024 * 1024
```

Related methods

- *setOfflineCacheAge()*

setOptOut()

The **setOptOut()** method sets the opt-out flag for the entire app. Once the opt-out flag is set, no data is collected.

By default, setOptOut(false) is set.

Syntax

Java

```
getTracker().setOptOut(isOptOut);
```

Kotlin

```
tracker.isOptOut = isOptOut
```

Parameters

isOptOut (boolean, required)
Whether the opt-out flag is set. True: is set. False: is not set.

Examples

To set the opt-out flag and not collect any data:

Java

```
getTracker().setOptOut(true);
```

Kotlin

```
tracker.isOptOut = true
```

setPrefixing()

The **setPrefixing()** method turns on or off automatic prefixing. If turned on, URLs will get prefixes automatically when certain methods are used. Example: The track().screen() method will add the `screen` prefix to the URL.

By default, setPrefixing(true) is set. This means that prefixes are added automatically.

Syntax

Java

```
getTracker().setPrefixing(isAutomatic);
```

Kotlin

```
tracker.isPrefixing = isAutomatic
```

Parameters

isAutomatic (boolean, required)
Whether URLs get prefixes automatically. True: prefixes are added automatically. False: prefixes are not added automatically.

Examples

To turn off automatic prefixing:

Java

```
getTracker().setPrefixing(false);
```

Kotlin

```
tracker.isPrefixing = false
```

setSessionTimeout()

The **setSessionTimeout()** method sets the the session timeout. The default value is 30 minutes.

Syntax

Java

```
getTracker().setSessionTimeout(milliseconds);
```

Kotlin

```
tracker.setSessionTimeout(milliseconds)
```

Parameters

milliseconds (number, required)
The time (in milliseconds) after which the session expires.

Examples

To set the expiration time to 60 minutes (60*60*1000 milliseconds):

Java

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

Kotlin

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

setTrackDeviceId()

The **setTrackDeviceId()** method turns on or off the fetching of the device ID from the tracker instance. The device ID is the advertising ID (AAID) assigned by Google.

By default, setTrackDeviceId(true) is set.

Syntax

Java

```
getTracker().setTrackDeviceId(isFetched);
```

Kotlin

```
tracker.isTrackDeviceId = isFetched
```

Parameters

isFetched (boolean, required)
Whether the user ID is fetched automatically from the tracker instance. The user ID is the advertising ID (AAID) assigned by Google. True: is fetched. False: is not fetched.

Examples

To turn off fetching the device ID from the tracker instance:

Java

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

Kotlin

```
tracker.isTrackDeviceId = false
```

Notes

- The device ID won't be sent if setAnonymizationState(true) is set.
- If your app uses the device ID (AAID) and you plan to submit your app to Google Play, you will need to ask each user for permission to share their data.

Related methods

- *setDeviceId()*
- *getDeviceId()*

setUserId()

The **setUserId()** method sets the user ID for a specific user. It can be the same identifier as in your CMS, CRM or sales system. The user ID helps to recognize the user across devices.

Syntax

Java

```
getTracker().setUserId("userId");
```

Kotlin

```
tracker.userId = "userId"
```

Parameters

userId (string, required)
A non-empty, unique identifier of the user. Example: customer ID. It needs to be unique for each user. Can be up to 1024 bytes (1024 ASCII characters).

Examples

To set a user ID as `ABC123` and send it with a screen view:

Java

```
getTracker().setUserId("ABC123");
TrackHelper.track()
  .screen("example/welcome")
  .title("Welcome")
  .with(tracker);
```

Kotlin

```
tracker.userId = "ABC123"
TrackHelper.track()
  .screen("example/welcome")
  .title("Welcome")
  .with(tracker);
```

Notes

- The user ID won't be sent if setAnonymizationState(true) is set.
- For more on the user ID, see this article.

setUserMail()

Deprecated since version 16.0.0: This method is no longer recommended. Audience Manager is no longer available in the latest product version.

The **setUserMail()** method sets a user email. The email can be sent to Audience Manager with a screen view or other event. The email enriches the user profile in Audience Manager and helps to recognize events belonging to the same user (only in the user profile in Audience Manager, but not in Analytics).

Syntax

Java

```
getTracker().setUserMail("userEmail");
```

Kotlin

```
getTracker().setUserMail("userEmail");
```

Parameters

userEmail (string, required)
The user's email address.

Examples

To set a user's email address and send it to Audience Manager with a screen view:

Java

```
getTracker().setUserMail("john.doe@example.com");
TrackHelper.track()
  .screen("example/welcome")
  .title("Welcome")
  .with(tracker);
```

Kotlin

```
tracker.userMail = "john.doe@example.com"
TrackHelper.track()
  .screen("example/welcome")
  .title("Welcome")
  .with(tracker);
```

Notes

- The user's email address is used only by Audience Manager. It is visible in Audience Manager > Profiles.
- The user's email address won't be sent if setAnonymizationState(true) is set.

setVisitorId()

The **setVisitorId()** method sets a custom visitor ID. By default, the visitor ID is automatically set when the tracker instance is created and is stored between app launches. The visitor ID helps to recognize sessions belonging to the same user.

Syntax

Java

```
getTracker().setVisitorId("visitorID");
```

Kotlin

```
tracker.visitorId = "visitorID"
```

Parameters

visitorID (string, required)
The visitor ID set for a specific user. Format: 16-character hexadecimal string. Example: 0123456789abcdef.

Examples

To set a visitor ID to `0123456789abcdef`:

Java

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

Kotlin

```
tracker.visitorId = "0123456789abcdef"
```

Notes

- If setAnonymizationState(true) is set, a new visitor ID is created each time the app is launched.
- Each user should have a unique visitor ID assigned. The visitor ID shouldn't change between app launches.
- We recommend using the user ID instead of the visitor ID.

startNewSession()

The **startNewSession()** method starts a new session.

Syntax

Java

```
getTracker().startNewSession();
```

Kotlin

```
tracker.startNewSession()
```

Examples

To start a new session:

Java

```
getTracker().startNewSession();
```

Kotlin

```
tracker.startNewSession()
```

track().campaign()

The **track().campaign()** method tracks online campaigns that bring traffic to your mobile app. To track a campaign, you need to add campaign parameters to each campaign link and then use this method to pass that data. Campaign data is collected with the first tracked screen event.

Syntax

Java

```
TrackHelper.track()
  .campaign("campaignURL");
```

Kotlin

```
TrackHelper.track()
  .campaign("campaignURL")
```

Parameters

campaignURL (string, required)
The URL you used in your campaign to bring traffic to your mobile app. Valid formats: HTTPS, HTTP and FTP. Example: `http://example.com?pk_campaign=Summer_Promo&pk_keyword=banking_app`

Note: You can tag campaigns manually or use our Piwik PRO URL builder. For now, only the pk_campaign and pk_keyword work on SDKs, so don't use any other parameters.

Examples

To pass campaign data from the campaign link `http://example.com?pk_campaign=Summer_Promo&pk_keyword=banking_app`:

Java

```
Tracker tracker = ((PiwikApplication) getApplication()).getTracker();
TrackHelper.track()
  .campaign("http://example.com?pk_campaign=Summer_Promo&pk_keyword=banking_app");

TrackHelper.track()
  .screen("example/welcome")
  .title("Welcome")
  .with(tracker);
```

Kotlin

```
val tracker: Tracker = (application as PiwikApplication).tracker
TrackHelper.track()
  .campaign("http://example.com?pk_campaign=Summer_Promo&pk_keyword=banking_app")

TrackHelper.track()
  .screen("example/welcome")
  .title("Welcome")
  .with(tracker)
```

Notes

- For now, only the pk_campaign and pk_keyword parameters work on SDKs, so don't use any other parameters.
- Piwik PRO recognizes the pk_campaign and pk_keyword parameters by default. But if you are having problems tracking your campaigns, make sure these parameters are added in Administration > Sites & apps > Data collection > Campaigns > Campaign parameters. Read more.

track().dimension()

The **track().dimension()** method sets a value for the custom dimension. The value can be sent to Piwik PRO with a screen view or other event.

Syntax

Java

```
TrackHelper.track()
  .dimension(customDimensionId, "customDimensionValue");
```

Kotlin

```
TrackHelper.track()
  .dimension(customDimensionId, "customDimensionValue")
```

Parameters

customDimensionId (number, required)
The ID of the custom dimension.

customDimensionValue (string, required)
The value of the custom dimension.

Examples

To set a custom dimension with the ID `1` and the value `5 stars` and send it with a screen view:

Java

```
TrackHelper.track()
  .dimension(1, "5 stars");
  .screen("example/product-rating")
  .title("Product rating")
  .with(tracker)
```

Kotlin

```
TrackHelper.track()
  .dimension(1, "5 stars");
  .screen("example/product-rating")
  .title("Product rating")
  .with(tracker)
```

To set a custom dimension with the ID `2` and the value `paid subscriber` and send it with an event:

Java

```
TrackHelper.track()
  .dimension(2, "paid subscriber");

TrackHelper.track()
  .event("Button", "Sign up")
  .with(tracker);
```

Kotlin

```
TrackHelper.track()
  .dimension(2, "paid subscriber");

TrackHelper.track()
  .event("Button", "Sign up")
  .with(tracker);
```

Notes

- After a dimension is sent with an event, it is deleted and will not be sent with the next event. So, you have to set it each time you want to send it.

track().download()

The **track().download()** method records clicks on links to downloadable files. You can use it to track app downloads.

Syntax

Java

```
TrackHelper.track()
  .sendDownload("downloadURL")
  .with(getTracker());
```

Kotlin

```
TrackHelper.track()
  .sendDownload("downloadURL")
  .with(tracker)
```

Parameters

downloadURL (string, required)
The download URL. Example: `https://example.com/paid-app.zip`

Examples

To track a click on `https://example.com/paid-app.zip` as a download:

Java

```
Tracker tracker = ((PiwikApplication) getApplication()).getTracker();
TrackHelper.track()
  .sendDownload("https://example.com/paid-app.zip")
  .with(tracker);
```

Kotlin

```
val tracker: Tracker = (application as PiwikApplication).tracker
TrackHelper.track()
  .sendDownload("https://example.com/paid-app.zip")
  .with(tracker)
```

Notes

- Downloads are visible in the download report in Analytics > Reports > Downloads.
- Here's the default list of file extensions tracked as downloads: 7z, aac, apk, arc, arj, asf, asx, avi, azw3, bin, csv, deb, dmg, doc, docx, epub, exe, flv, gif, gz, gzip, hqx, ibooks, jar, jpg, jpeg, js, mobi, mp2, mp3, mp4, mpg, mpeg, mov, movie, msi, msp, odb, odf, odg, ods, odt, ogg, ogv, pdf, phps, png, ppt, pptx, qt, qtm, ra, ram, rar, rpm, sea, sit, tar, tbz, tbz2, bz, bz2, tgz, torrent, txt, wav, wma, wmv, wpd, xls, xlsx, xml, z, zip.

track().event()

The **track().event()** method records actions performed by users on your mobile app – like button presses, gestures or voice commands.

Syntax

Java

```
TrackHelper.track()
  .event("category", "action")
  .path("path")
  .name("name")
  .value(value)
  .with(getTracker());
```

Kotlin

```
TrackHelper.track()
  .event("category", "action")
  .path("path")
  .name("name")
  .value(value)
  .with(tracker)
```

Parameters

category (string, required)
The category of the tracked event. You can define event categories based on actions (clicks, gestures, voice commands) or features (play, pause, fast forward).

action (string, required)
The action of the tracked event. Example: A category could be user clicks, an action could be a button click.

name (string, optional)

The name of the tracked event. For example, if you have multiple button controls on the screen, you can use the name to record the specific ID of the button that was clicked.

value (float, optional)
The value you want to assign to the tracked event. For example, if you're tracking "Buy" button presses, you can record the number of purchased items or the total cost.

path (string, optional)
The URL path set for this event.

Examples

To send a custom event when a user clicks a sign-up button on ‘‘/main/sign-up" and assign the value ‘‘100" to the event:

Java

```
Tracker tracker = ((PiwikApplication) getApplication()).getTracker();
TrackHelper.track()
  .event("Clicks", "Button")
  .path("/main/signup")
  .name("Sign up")
  .value(100)
  .with(tracker);
```

Kotlin

```
val tracker: Tracker = (application as PiwikApplication).tracker
TrackHelper.track()
  .event("Clicks", "Button")
  .path("/main/signup")
  .name("Sign up")
  .value(100)
  .with(tracker)
```

Notes

- For more on custom events, see this article.

track().exception()

The **track().exception()** method records caught exceptions (errors) in your app. For each exception, you need to define handling code.

Syntax

Java

```
TrackHelper.track()
  .exception(ex)
  .description("description")
  .with(getTracker());
```

Kotlin

```
TrackHelper.track()
  .exception(ex)
  .description("description")
  .with(tracker)
```

Parameters

ex (Throwable, optional)
The caught exception instance. The exception instance is automatically translated into a URL and the following information is added to it: package name, activity path, method name and line number where the crash occurred.

description (string, optional)
Additional information about the issue.

Examples

To send a caught exception:

Java

```
TrackHelper.track()
  .exception(new Exception("OnPurposeException"))
  .description("Download error")
  .with(getTracker());
```

Kotlin

```
TrackHelper.track()
  .exception(Exception("OnPurposeException"))
  .description("Download error")
  .with(tracker)
```

track().goal()

The **track().goal()** method tracks completed goals in your app. You can set any event as a goal, like when visitors sign up, buy your product, download a whitepaper or do anything else you think is important for your business.

Syntax

Java

```
TrackHelper.track()
  .goal("goalID")
  .revenue(conversionValue)
  .with(getTracker());
```

Kotlin

```
TrackHelper.track()
  .goal("goalID")
  .revenue(conversionValue)
  .with(tracker)
```

Parameters

goalID (string , required)
The ID of the goal you want to track. (To find a goal ID go to Menu > Analytics > Goals.)

conversionValue (number, optional)
The value of the goal. It's used to calculate the goal revenue.

Examples

To send a goal with the ID `27ecc5e3-8ae0-40c3-964b-5bd8ee3da059` and the value `20`:

Java

```
Tracker tracker = ((PiwikApplication) getApplication()).getTracker();
TrackHelper.track()
  .goal("27ecc5e3-8ae0-40c3-964b-5bd8ee3da059")
  .revenue(20)
  .with(tracker);
```

Kotlin

```
val tracker: Tracker = (application as PiwikApplication).tracker
TrackHelper.track()
  .goal("27ecc5e3-8ae0-40c3-964b-5bd8ee3da059")
  .revenue(20)
  .with(tracker)
```

Notes

- After you set up a goal in Analytics > Goals > Add a goal, the goal is tracked automatically. The track().goal() method can be used in addition to the automatic method.
- For more on goals, see our help article.

track().impression()

The **track().impression()** method tracks the impressions of a content block and passes data about the content name, piece and target.

Syntax

Java

```
TrackHelper.track()
  .impression("contentName")
  .piece("contentPiece")
  .target("contentTarget")
  .with(getTracker());
```

Kotlin

```
TrackHelper.track()
  .impression("contentName")
  .piece("contentPiece")
  .target("contentTarget")
  .with(tracker)
```

Parameters

contentName (string, required)
The name of the tracked content block.

contentPiece (string, required)
The piece of the tracked content block. Example: a creative, banner or video.

contentTarget (string, required)
The target of the tracked content block. Example: a link in the content block.

Examples

To track the impression of a content block on your mobile app:

Java

```
Tracker tracker = ((PiwikApplication) getApplication()).getTracker();
TrackHelper.track()
  .impression("gravel bikes collection")
  .piece("banner")
  .target("https://example.com/bikes/")
  .with(tracker);
```

Kotlin

```
val tracker: Tracker = (application as PiwikApplication).tracker
TrackHelper.track()
  .impression("gravel bikes collection")
  .piece("banner")
  .target("https://example.com/bikes/")
  .with(tracker)
```

Notes

- To track content impressions, this option needs to be turned on: Menu > Tag Manager > Tags > Piwik PRO (tracking code) > Data collection > Interactions with popups and content (on). Read more
- Tracked impressions will be visible in Analytics > Reports > Content performance.

Related methods

- *track().interaction()*

track().interaction()

The **track().interaction()** method tracks the interactions with a content block and passes data about the content name, piece and target.

Syntax

Java

```
TrackHelper.track()
  .impression("contentName", "contentInteraction ")
  .piece("contentPiece")
  .target("contentTarget")
  .with(getTracker());
```

Kotlin

```
TrackHelper.track()
  .impression("contentName", "contentInteraction ")
  .piece("contentPiece")
  .target("contentTarget")
  .with(tracker)
```

Parameters

contentName (string, required)
The name of the tracked content block.

contentInteraction (string, required)
The type of interaction with the tracked content block. Example: click.

contentPiece (string, required)
The piece of the tracked content block. Example: a creative, banner, or video.

contentTarget (string, required)
The target of the tracked content block. Example: a link in the content block.

Examples

To track the interaction with a content block on your mobile app:

Java

```
Tracker tracker = ((PiwikApplication) getApplication()).getTracker();
TrackHelper.track()
  .impression("gravel bikes collection", "click")
  .piece("banner")
  .target("https://example.com/bikes/")
  .with(getTracker());
```

Kotlin

```
val tracker: Tracker = (application as PiwikApplication).tracker
TrackHelper.track()
  .impression("gravel bikes collection", "click")
  .piece("banner")
  .target("https://example.com/bikes/")
  .with(tracker)
```

Notes

- To track interactions with content, this option needs to be turned on: Menu > Tag Manager > Tags > Piwik PRO (tracking code) > Data collection > Interactions with popups and content (on). Read more
- Tracked interactions will be visible in Analytics > Reports > Content performance.

Related methods

- *track().impression()*

track().order()

The **track().order()** method tracks a confirmed order.

Syntax

Java

```
TrackHelper.track()
  .order("orderID", orderGrandTotal)
  .subTotal(orderSubTotal)
  .tax(orderTax)
  .shipping(orderShipping)
  .discount(orderDiscount)
  .items(items)
  .with(getTracker());
```

Kotlin

```
TrackHelper.track()
  .order("orderID", orderGrandTotal).subTotal(orderSubTotal)
  .tax(orderTax)
  .shipping(orderShipping)
  .discount(orderDiscount)
  .items(items)
  .with(tracker)
```

Parameters

orderID (string, required)
The unique order ID.

orderGrandTotal (number, required)
Total payment for the order. Includes tax, shipping and discounts. Format: 1/100 of the base currency unit. Example: 100 is 1 USD.

orderSubTotal (number, optional)
Payment for the order without shipping. Format: 1/100 of the base currency unit. Example: 100 is 1 USD.

orderTax (number, optional)
Tax included in the order. Format: 1/100 of the base currency unit. Example: 100 is 1 USD.

orderShipping (number, optional)
Shipping costs for the order. Format: 1/100 of the base currency unit. Example: 100 is 1 USD.

orderDiscount (number, optional)
Discounts included in the order. Format: 1/100 of the base currency unit. Example: 100 is 1 USD.

Examples

To track a confirmed order:

Java

```
Tracker tracker = ((YourApplication) getApplication()).getTracker();
EcommerceItems items = new EcommerceItems();

// register all purchased items
// EcommerceItems.Item("<SKU>").name("<name>").category("<category>").price(<price>).
↪quantity(<quantity>)

items.addItem(new EcommerceItems
  .Item("584340")
  .name("Specialized Stumpjumper")
  .category("Mountain bike")
  .price(500000)
  .quantity(1));

items.addItem(new EcommerceItems
  .Item("460923")
  .name("Specialized Chamonix")
  .category("Helmets")
  .price(20000)
  .quantity(1));

// track order

TrackHelper.track()
  .order("43967392", 525000)
  .subTotal(520000)
  .tax(97000)
  .shipping(15000)
  .discount(10000)
  .items(items)
  .with(tracker);
```

Kotlin

```
val tracker: Tracker = (application as PiwikApplication).tracker
var items: EcommerceItems = EcommerceItems()

// register all purchased items
// EcommerceItems.Item("<SKU>").name("<name>").category("<category>").price(<price>).
↪quantity(<quantity>)

items.addItem(EcommerceItems
  .Item("584340")
  .name("Specialized Stumpjumper")
  .category("Mountain bike")
  .price(500000)
  .quantity(1))

items.addItem(EcommerceItems
  .Item("460923")
  .name("Specialized Chamonix")
  .category("Helmets")
  .price(20000)
  .quantity(1))

// track order
```

(continues on next page)

(continued from previous page)

```
TrackHelper.track()
  .order("43967392", 525000)
  .subTotal(520000)
  .tax(97000)
  .shipping(15000)
  .discount(10000)
  .items(items)
  .with(tracker)
```

Related methods

- *items.addItem()*

track().outlink()

The **track().outlink()** method records clicks on links to external websites or apps (different domains).

Syntax

Java

```
TrackHelper.track()
  .outlink("outlink")
  .with(getTracker());
```

Kotlin

```
TrackHelper.track()
  .outlink("outlink")
  .with(tracker)
```

Parameters

outlink (string, required)
The outlink. Example: `https://example.com`.

Examples

To track an outlink to `https://example.com`:

Java

```
Tracker tracker = ((PiwikApplication) getApplication()).getTracker();
TrackHelper.track()
  .outlink("https://example.com")
  .with(tracker);
```

Kotlin

```
val tracker: Tracker = (application as PiwikApplication).tracker
TrackHelper.track()
  .outlink("https://example.com")
  .with(tracker)
```

track().screen()

The **track().screen()** method records a screen view on your mobile app. The screen view is similar to the page view on a website.

Syntax

Java

```
TrackHelper.track()
  .screen("path")
  .title("title")
  .with(tracker);
```

Kotlin

```
TrackHelper.track()
  .screen("path")
  .title("title")
  .with(tracker)
```

Parameters

path (string, required)
The path set for your screen. Example: `example/welcome`. The path is automatically translated into a URL and given the `screen` prefix if tracker.setPrefixing(true) is set.

Note: Set the current instance of the Android `Activity` class instead of the path if you want to use the activity stack. It'll then automatically set the activity stack as the path and the activity title as the title.

title (string, optional)
The title set for your screen. Example: Welcome.

Examples

To send a screen view with the path `example/welcome` and the title `Welcome`:

Java

```
public class activityClass extends Activity {
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
```

(continues on next page)

(continued from previous page)

```
  Tracker tracker = ((PiwikApplication) getApplication()).getTracker();
  TrackHelper.track()
    .screen("example/welcome")
    .title("Welcome")
    .with(tracker);
  }
}
```

Kotlin

```
public class activityClass : Activity() {
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    val tracker: Tracker = (application as PiwikApplication).tracker
    TrackHelper.track()
      .screen("example/welcome")
      .title("Welcome")
      .with(tracker)
  }
}
```

To send a screen view and automatically use the activity stack as the path and the activity name as the title (if your activity class is activityClass):

Java

```
public class activityClass extends Activity {
...
Tracker tracker = ((PiwikApplication) getApplication()).getTracker();
TrackHelper.track().screen(activityClass).with(tracker);
...
}
```

Kotlin

```
public class activityClass  : Activity() {
...
val tracker: Tracker = (application as PiwikApplication).tracker
TrackHelper.track().screen(activityClass).with(tracker)
...
}
```

Related methods

- *track().screens()*

track().screens()

The **track().screens()** method automatically records screen views on your mobile app. It automatically uses the activity stack as the path and the activity name as the title.

Syntax

Java

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

Kotlin

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

Examples

To automatically record screen views:

Java

```
Tracker tracker = ((PiwikApplication) getApplication()).getTracker();
TrackHelper.track()
  .screens(getApplication())
  .with(tracker);
```

Kotlin

```
val tracker: Tracker = (application as PiwikApplication).tracker
TrackHelper.track()
  .screens(getApplication())
  .with(tracker)
```

Related methods

- *track().screen()*

track().search()

The **track().search()** method tracks searches on your internal search engine.

Syntax

Java

```
TrackHelper.track()
  .search("keyword")
  .category("category")
  .count(searchCount)
  .with(getTracker());
```

Kotlin

```
TrackHelper.track()
  .search("keyword")
  .category("category")
  .count(searchCount)
  .with(tracker)
```

Parameters

keyword (string, required)
The keyword the user typed into the search box.

category (string | array<string>, optional)
The category selected in the search engine.

searchCount (number, optional)
The number of search results.

Examples

To send an internal search with the keyword "ATM in London" and 20 search results, but no category:

Java

```
Tracker tracker = ((PiwikApplication) getApplication()).getTracker();
TrackHelper.track()
  .search("ATM in London")
  .category("")
  .count(20)
  .with(tracker);
```

Kotlin

```
val tracker: Tracker = (application as PiwikApplication).tracker
TrackHelper.track()
  .search("ATM in London")
  .category("")
  .count(20)
  .with(tracker)
```

track().sendApplicationDownload()

The **track().sendApplicationDownload()** method records app installations. The event is sent the first time the app is launched, once per installation. If a user installs your app but doesn't run it, the event is not sent.

Syntax

Java

```
TrackHelper.track()
  .sendApplicationDownload()
  .with(getTracker());
```

Kotlin

```
TrackHelper.track()
  .sendApplicationDownload()
  .with(tracker)
```

Example

To track the installation of your app:

Java

```
Tracker tracker = ((PiwikApplication) getApplication()).getTracker();
TrackHelper.track()
  .sendApplicationDownload()
  .with(tracker);
```

Kotlin

```
val tracker: Tracker = (application as PiwikApplication).tracker
TrackHelper.track()
  .sendApplicationDownload()
  .with(tracker)
```

track().socialInteraction()

The **track().socialInteraction()** method records likes, shares and comments on social media on your app.

Syntax

Java

```
TrackHelper.track()
  .socialInteraction("interaction", "network")
  .with(getTracker());
```

Kotlin

```
TrackHelper.track()
  .socialInteraction("interaction", "network")
  .with(tracker)
```

Parameters

interaction (string, required)
The interaction type. Example: like, share, comment.

network (string, required)
The social media for which the interaction happened. Example: Facebook, Instagram, YouTube.

Examples

To track a Facebook like on your app:

Java

```
TrackHelper.track()
  .socialInteraction("like", "Facebook")
  .with(getTracker());
```

Kotlin

```
TrackHelper.track()
  .socialInteraction("like", "Facebook")
  .with(tracker)
```

track().variable()

Deprecated since version 16.0.0: This method is no longer recommended. Audience Manager is no longer available in the latest product version.

The **track().variable()** method sets a custom variable in the screen scope. The value can be sent to Piwik PRO with a screen view or other event.

Syntax

Java

```
TrackHelper.track()
  .variable(index, "name", "value");
```

Kotlin

```
TrackHelper.track()
  .variable(index, "name", "value")
```

Parameters

index (number, required)
The index where the variable is stored.

Note: If setIncludeDefaultCustomVars(true) is set, you can only use an index greater than 2 because this method automatically tracks some items under the index 1-2. The setIncludeDefaultCustomVars(true) method is set by default.

name (string, required)

The name of the variable. Valid format: UTF-8.

value (string, optional)
The value of the variable. Valid format: UTF-8. Limited to 200 characters.

Examples

To set a custom variable in the screen scope and send it with a screen view:

Java

```
TrackHelper.track()
  .variable(1, "rating", "5");

TrackHelper.track()
  .screen("example/product-rating")
  .title("Product rating")
  .with(getTracker());
```

Kotlin

```
TrackHelper.track()
  .variable(1, "rating", "5")

TrackHelper.track()
  .screen("example/product-rating")
  .title("Product rating")
  .with(tracker)
```

Another way to set a custom variable and send it with a screen view:

Java

```
TrackHelper.track()
  .variable(1, "rating", "5")
  .screen("example/product-rating")
  .title("Product rating")
  .with(getTracker());
```

Kotlin

```
TrackHelper.track()
  .variable(1, "rating", "5")
  .screen("example/product-rating")
  .title("Product rating")
  .with(tracker)
```

Notes

- The screen scope refers to events like a screen view or file download and holds a captured variable for each event. The value is removed after an event is called.

Related methods

- *setIncludeDefaultCustomVars()*
- *track().visitVariables()*
- *track().dimension()*

track().visitVariables()

Deprecated since version 16.0.0: This method is no longer recommended. Audience Manager is no longer available in the latest product version.

The **track().visitVariables()** method sets a custom variable in the visit (session) scope. The value can be sent to Piwik PRO with a screen view or other event.

Syntax

Java

```
TrackHelper.track()
  .visitVariables(index, "name", "value");
```

Kotlin

```
TrackHelper.track()
  .visitVariables(index, "name", "value")
```

Parameters

index (number, required)
The index where the variable is stored.

Note: If setIncludeDefaultCustomVars(true) is set, you can't use the index 4-5 because this method automatically tracks some items under these indexes. The setIncludeDefaultCustomVars(true) method is set by default.

name (string, required)
The name of the variable. Valid format: UTF-8. Limited to 200 characters.

value (string, optional)
The value of the variable. Valid format: UTF-8. Limited to 200 characters.

Examples

To set a custom variable in the visit (session) scope and send it with a screen view:

Java

```
TrackHelper.track()
  .visitVariables(1, "age", "25")

TrackHelper.track()
  .screen("example/welcome")
  .title("Welcome")
  .with(getTracker());
```

Kotlin

```
TrackHelper.track()
  .visitVariables(1, "age", "25")

TrackHelper.track()
  .screen("example/welcome")
  .title("Welcome")
  .with(tracker)
```

Notes

- The visit (session) scope refers to the entire session and holds the captured custom dimension for the entire session.

Related methods

- *setIncludeDefaultCustomVars()*
- *track().variable()*
- *track().dimension()*

3.2 Flutter SDK

3.2.1 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 "Administration > Websites & apps > Installation" and your server address.

Client

Run this command:

With Dart:

```
$ dart pub add flutter_piwikpro
```

With Flutter:

```
$ flutter pub add flutter_piwikpro
```

This will add a line like this to your package's pubspec.yaml (and run an implicit dart pub get):

```
dependencies:
  flutter_piwikpro: ^0.0.1
```

Alternatively, your editor might support dart pub get or flutter pub get. Check the docs for your editor to learn more.

Import it

Now in your Dart code, you can use:

import 'package:flutter_piwikpro/flutter_piwikpro.dart';

Configuration

You'll need to configure the tracker before using any other methods - for that you will need the base URL address of your tracking server and website ID (you can find it in Administration > Websites & apps > Installation on the web interface).

```
await FlutterPiwikPro.sharedInstance.configureTracker(baseURL: 'https://your.piwik.
↪pro.server.com', siteId: '01234567-89ab-cdef-0123-456789abcdef');
```

iOS and Android parameters:

- `String baseURL` - base URL of your tracking server
- `String siteId` - ID of your website or application

Usage and general info

Every method from the sdk is async, and every method can throw exceptions - for example if you try to use sdk methods without configuring the tracker first - which you can capture using the standard try-catch approach. For example:

```
try {
    final result =
    await FlutterPiwikPro.sharedInstance.trackDownload('http://your.server.com/
↪bonusmap2.zip');
    print(result);
} catch (exception) {
    //handle an exception
}
```

If a method call is succesful, most of the methods, unless specified, will return a String that describes which method was called, and which parameters were used, for example:

```
FlutterPiwikPro - configureTracker completed with parameters: baseURL: https://your.
↪piwik.pro.server.com, siteId: 01234567-89ab-cdef-0123-456789abcdef
```

3.2.2 Using Piwik PRO SDK Flutter Wrapper

Data Anonymization

Anonymization is a feature that allows tracking a user's activity for aggregated data analysis even if the user doesn't consent to track the data. If a user does not agree to being tracked, he will not be identified as the same person across multiple sessions.

Personal data will not be tracked during the session (i.e. user ID) If the anonymization is enabled, a new visitor ID will be created each time the application starts.

Anonymization is enabled by default.

You can turn the anonymization on and off by calling `setAnonymizationState`:

```
await FlutterPiwikPro.sharedInstance.setAnonymizationState(true);
```

- `bool shouldAnonymize` - pass `true` to enable anonymization, or `false` to disable it.

Tracking Screen Views

The basic functionality of the SDK is 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).

To track screen views you can use the `trackScreen` method:

```
    await FlutterPiwikPro.sharedInstance.trackScreen(screenName: "menuScreen");
```

iOS and Android parameters:

- `String path` – title of the action being tracked. The appropriate screen path will be generated for this action.

Additional Android only parameters:

- `String? title` (optional) – the title of the action being tracked.

Tracking Custom Events

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. You can read more about events in the Piwik PRO documentation and ultimate guide to event tracking.

To track custom events you can use the `trackCustomEvent` method:

```
await FlutterPiwikPro.sharedInstance.trackCustomEvent(
    action: 'test action',
    category: 'test category',
    name: 'test name',
    value: 120);
```

iOS and Android parameters:

- `String category` – this String defines the event category. You may define event categories based on the class of user actions (e.g. taps, gestures, voice commands), or you may define them based upon the features available in your application (e.g. play, pause, fast forward, etc.).
- `String action` – 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.
- `String? 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 identifier of a button that was clicked.
- `double? 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.

Additional Android only parameters:

- `String? path` (optional) - the path under which this event occurred.

Tracking Exceptions

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.

To track exceptions you can use the `trackException` method:

```
await FlutterPiwikPro.sharedInstance.trackException(description: "description of an␣
↪exception", isFatal: false);
```

iOS and Android parameters:

- `String description` – provides the exception message.
- `bool isFatal` – true if an exception is fatal.

Tracking Social Interactions

Social interactions such as likes, shares and comments in various social networks can be tracked as below. This is tracked in a similar way as screen views.

To track social interactions you can use the `trackSocialInteraction` method:

```
await FlutterPiwikPro.sharedInstance.trackSocialInteraction(
    interaction: 'like',
    network: 'Facebook',
    target: 'Dogs');
```

iOS and Android parameters

- `String interaction` – defines the social interaction, e.g. "Like".
- `String network` – defines the social network associated with interaction, e.g. "Facebook"
- `String? target` (optional) – the target for which this interaction occurred, e.g. "Dogs".

Tracking Downloads

You can track downloads initiated by your application by using the `trackDownload` method:

```
await FlutterPiwikPro.sharedInstance.trackDownload('http://your.server.com/bonusmap2.
↪zip');
```

iOS and Android parameters

- `String url` - URL of the downloaded content.

Tracking Application Installs

You can also track installations of your application. This event is sent to the server only once per apps version (additional events won't be sent).

You can track app installs using the `trackAppInstall` method:

```
await FlutterPiwikPro.sharedInstance.trackAppInstall();
```

Tracking Outlinks

For tracking outlinks to external websites or other apps opened from your application you can use the trackOutlink method:

```
await FlutterPiwikPro.sharedInstance.trackOutlink('http://great.website.com');
```

iOS and Android parameters

- `String url` - defines the outlink target. HTTPS, HTTP and FTP are valid.

Tracking Search Operations

Tracking search operations allow the measurement of popular keywords used for various search operations performed inside your application. To track them you can use the `trackSearch` method:

```
await FlutterPiwikPro.sharedInstance.trackSearch(keyword: 'Space', category: "Movies",
↪ numberOfHits: 100);
```

iOS and Android parameters

- `String keyword` – the searched query that was used in the app.
- `String category` – specify a search category.
- `int? 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

You can track the impression of an ad using the `trackContentImpression` method:

```
await FlutterPiwikPro.sharedInstance.trackContentImpression(
    contentName: "name",
    piece: 'piece',
    target: 'target');
```

iOS and Android parameters

- `String contentName` – the name of the content, e.g. "Ad Foo Bar".
- `String? piece` (optional) – the actual content. For instance the path to an image, video, audio, any text.
- `String? target` (optional) – the target of the content e.g. the URL of a landing page.

Tracking Content Interactions

When a user interacts with an ad by tapping on it, you can track it using the `trackContentInteraction` method:

```
await FlutterPiwikPro.sharedInstance.trackContentInteraction(
    contentName: "name",
    piece: 'piece',
    target: 'target',
    contentInteraction: 'Clicked really hard');
```

iOS and Android parameters

- `String contentName` – the name of the content, e.g. "Ad Foo Bar".
- `String? piece` (optional) – the actual content. For instance the path to an image, video, audio, any text.
- `String? target` (optional) – the target of the content e.g. the URL of a landing page.
- `String? contentInteraction` (optional) - a type of interaction that occured, e.g. "tap"

Tracking Goals

Goal tracking 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 To track goals you can use the `trackGoal` method:

```
await FlutterPiwikPro.sharedInstance.trackGoal(goal: 10, revenue: 102.2);
```

iOS and Android parameters

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

Tracking Ecommerce Transactions

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. To track an ecommerce transaction you can use the `trackEcommerceTransaction` method:

```
final ecommerceTransactionItems = [
EcommerceTransactionItem(category: 'cat1', sku: 'sku1', name: 'name1', price: 20,␣
↪quantity: 1),
EcommerceTransactionItem(category: 'cat2', sku: 'sku2', name: 'name2', price: 10,␣
↪quantity: 1),
EcommerceTransactionItem(category: 'cat3', sku: 'sku3', name: 'name3', price: 30,␣
↪quantity: 2),
];
await FlutterPiwikPro.sharedInstance.trackEcommerceTransaction(
    identifier: "transactionID",
    grandTotal: 100,
    subTotal: 10,
    tax: 5,
    shippingCost: 100,
    discount: 6,
    transactionItems: ecommerceTransactionItems,
);
```

iOS and Android parameters

- `String identifier` – a unique string identifying the order
- `int grandTotal` – The total amount of the order, in cents
- `int? subTotal` (optional) – the subtotal (net price) for the order, in cents
- `int? tax` (optional) – the tax for the order, in cents
- `int? shippingCost` (optional) – the shipping for the order, in cents
- `int? discount` (optional) – the discount for the order, in cents
- `List<EcommerceTransactionItem>? transactionItems` (optional) – the items included in the order

Tracking Campaigns

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. To track a campaign you can use the `trackCampaign` method:

```
await FlutterPiwikPro.sharedInstance.trackCampaign("http://example.org/offer.html?pk_
↪campaign=Email-SummerDeals&pk_keyword=LearnMore");
```

iOS and Android parameters

- `String url` - the campaign URL. HTTPS, HTTP and FTP are valid - the URL must contain a campaign name and keyword parameters.

Tracking Custom Variables

The feature will soon be disabled. We recommend using custom dimensions instead.

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. It is required for names and values to be encoded in UTF-8. You can add a custom variable using the `trackCustomVariable` method:

```
await FlutterPiwikPro.sharedInstance.trackCustomVariable(
    index: 1,
    name: 'filter',
    value: 'lcd',
    scope: CustomVariableScope.visit);
```

iOS and Android parameters

- `int index` – 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.
- `String name` – this String defines the name of a specific Custom Variable such as "User type". Limited to 200 characters.
- `String value` – this String defines the value of a specific Custom Variable such as "Customer". Limited to 200 characters.
- `CustomVariableScope scope` – this String allows the specification of the tracking event type - "visit", "action", etc. The scope is the value from the enum CustomVariableScope and can be `visit` or `action`.

Tracking Custom Dimensions

You can also use custom dimensions to track custom values. Custom dimensions first have to be defined on the server in your web panel. More details about custom dimensions can be found in the documentation You can add a custom

dimension using the `trackCustomDimension` method:

```
await FlutterPiwikPro.sharedInstance.trackCustomDimension(id: 1, value: 'english');
```

iOS and Android parameters

- `int index` – 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.
- `String value` – this String defines the value of a specific custom dimension such as "English". Limited to 200 characters.

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. To set profile attributes you can use the `trackProfileAttribute` method:

```
await FlutterPiwikPro.sharedInstance.trackProfileAttribute(name: 'food', value: 'chips
↪');
```

iOS and Android parameters

- `String name` – defines profile attribute name (non-null string).
- `String value` – defines profile attribute value (non-null string).

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

- `WEBSITE_ID` – always sent.
- `USER_ID` – if set.
- `EMAIL` – if set.
- `VISITOR_ID` – always sent, ID of the mobile application user, generated by the SDK.
- `DEVICE_ID` – Advertising ID that, by default, is fetched automatically when the tracker instance is created (only on Android).

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 you can use the `readUserProfileAttributes` method:

```
await FlutterPiwikPro.sharedInstance.readUserProfileAttributes()
```

Returned Value

- `Future<Map<String, String>>` - this method returns a Map of key-value pairs, where each pair represent attribute name (key) and value (instead of a usual String that describes which method was called with which parameters)

Checking Audience Membership

Requires Audience Manager

Checking audience membership 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 a user belongs to a given audience, for example, to display him/her some type of special offer. You can check audience membership using the `checkAudienceMembership` method:

```
await FlutterPiwikPro.sharedInstance.checkAudienceMembership('audienceId');
```

iOS and Android parameters

- `String audienceId` – ID of the audience (Audience Manager -> Audiences tab)

Returned Value

- `Future<bool>` - this method returns a bool value (true if a user is a member of an audience, false otherwise) instead of a usual String that describes which method was called with which parameters.

3.2.3 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. You can read more about User ID here. You can set a user id using the `setUserId` method:

```
await FlutterPiwikPro.sharedInstance.setUserId('testUserId')
```

iOS and Android parameters

- `String id` – any non-empty unique string identifying the user. Passing null will delete the current user ID

User Email Address

The user email address is an another string used for identifying users - a 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.

It is recommended to set the user email to track audience manager profile attributes as it will create a better user profile.

You can set a user email using the `setUserEmail` method:

```
await FlutterPiwikPro.sharedInstance.setUserEmail('user@email.com');
```

iOS and Android parameters

- `String email` – a string representing an email address

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. When the anonymization is enabled, a new visitor id is generated each time the application is started. We recommend using userID instead of VisitorID. Still, you can set a visitor ID using the `setVisitorId` method:

```
await FlutterPiwikPro.sharedInstance.setVisitorId('Id');
```

iOS and Android parameters

- `String id` - a string containing a new Visitor ID

Setting Session Timeout

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. You can change the session timeout using the `setSessionTimeout` method:

```
await FlutterPiwikPro.sharedInstance.setSessionTimeout(1000)
```

iOS and Android parameters

- `int timeoutInMilliseconds` - Session timeout in milliseconds. Default: 1800000 (30 minutes)

Setting Dispatch Interval

All tracking events are saved locally and by default. They are automatically sent to the server every 30 seconds. You can change this interval using the `setDispatchInterval` method:

```
await FlutterPiwikPro.sharedInstance.setDispatchInterval(10000)
```

iOS and Android parameters

- `int intervalInMilliseconds` - Dispatch interval in milliseconds. Default: 30000

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.

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

You can disable this behavior using the `setIncludeDefaultVariables` method:

```
await FlutterPiwikPro.sharedInstance.setIncludeDefaultVariables(false);
```

iOS and Android parameters

- `bool shouldInclude` - a boolean value that removes Default Variables when `false`

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. You can opt out of tracking using the `optOut` method:

```
await FlutterPiwikPro.sharedInstance.optOut(true);
```

iOS and Android parameters

- `bool shouldOptOut` - a boolean value that disables all tracking in the app when set to `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. You can set set the dry run flag using the `dryRun` method:

```
await FlutterPiwikPro.sharedInstance.dryRun(true);
```

iOS and Android parameters

- `bool shouldDryRun` - a boolean value that prevents any data being sent to a tracker when set to `true`

3.3 iOS SDK

3.3.1 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 "Administration > Websites & apps > Installation" 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. `'~> 1.0.7'`.

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 *Administration* > *Websites & apps* > *Installation* 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;
}
```

Using Piwik PRO SDK with the Swift programming language

The Piwik PRO SDK is written in the Objective-C programming language. However, after installing the library from cocoapods, Xcode automatically generates Swift syntax for Objective-C calls. When you edit a Swift file and type in an Objective-C class name, Swift version of the header file will be displayed.

Example of using the method to track a view in Objective-c:

```
[[PiwikTracker sharedInstance] sendView:@"Menu"];
```

Same example in Swift:

```
import PiwikPROSDK

PiwikTracker.sharedInstance()?.sendView(view: "Menu")
```

If there is a need to create the bridging header, see the apple tutorial "Importing Objective-C into Swift" for additional information.

3.3.2 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.

Data anonymization

Anonymization is the feature that allows tracking a user's activity for aggregated data analysis even if the user doesn't consent to track the data. If a user does not agree to be tracked, he will not be identified as the same person across multiple sessions.

Personal data will not be tracked during the session (i.e. *user ID*, *device ID*) If the anonymization is enabled, a new *visitor ID* will be created each time the application starts.

Anonymization is enabled by default.

You can turn the anonymization on and off by setting the value of the variable `isAnonymizationEnabled`:

```
[PiwikTracker sharedInstance].isAnonymizationEnabled = NO;
```

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

Custm 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

The feature will soon be disabled. We recommend using custom dimensions instead.

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 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] setCustomDimensionForID:1 value:@"english"];
```

- 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.

Assigning a value to an already used index will overwrite the previously assigned value.

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.

3.3.3 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 userMail (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. When the anonymization is enabled, a new visitor id is generated each time the application is started. 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 sharedManager] 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;
```

3.4 React Native SDK

Piwik PRO SDK for React Native

3.4.1 Installation

```
npm install @piwikpro/react-native-piwik-pro-sdk
```

3.4.2 Configuration

In order to set up the Piwik PRO tracker you have to call `init` method passing a server address and website ID (you can find it in `Administration` -> `Sites & apps`):

```
import PiwikProSdk from "@piwikpro/react-native-piwik-pro-sdk";

// ...

await PiwikProSdk.init('https://your.piwik.pro.server.com', '01234567-89ab-cdef-0123-
↪456789abcdef')
```

Parameters:

- `serverAddress: string` *(required)* – URL of your Piwik PRO server.
- `websiteId: string` *(required)* – ID of your website or application.

Note: Each tracking method is implemented as a Promise which will be rejected if the `PiwikProSdk` has not been initialized.

3.4.3 Using Piwik PRO SDK

Data anonymization

Anonymization is the feature that allows tracking a user's activity for aggregated data analysis even if the user doesn't consent to track the data. If a user does not agree to be tracked, he will not be identified as the same person across multiple sessions.

Personal data will not be tracked during the session (i.e. user ID, device ID). If the anonymization is enabled, a new visitor ID will be created each time the application starts.

Anonymization is enabled by default.

You can turn the anonymization on and off using the `setAnonymizationState` method:

```
await PiwikProSdk.setAnonymizationState(false);
```

Parameters:

- `anonymizationState: boolean` *(required)* – new anonymization state.

You can also check the anonymization status using the `isAnonymizationOn` method:

```
const anonymizationState = await PiwikProSdk.isAnonymizationOn();
```

Returns:

- `anonymizationState: boolean` – current anonymization state.

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 track a screen you only need to provide the screen path. 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 `setPrefixing(false)` option.

```
const options = {
  title: 'actionTitle',
  customDimensions: { 1: 'some custom dimension value' },
};
await PiwikProSdk.trackScreen(`your_screen_path`, options);
```

Parameters:

- `path: string` *(required)* – screen path (it will be mapped to the URL path).
- `options` – screen tracking options, object containing four properties (all of them are optional):
  - `title: string` – the title of the action being tracked (it will be omitted in iOS application).
  - `customDimensions` – the object specifying *custom dimensions*.
  - `screenCustomVariables` – the object specifying *screen custom variables*.
  - `visitCustomVariables` – the object specifying *visit custom variables*.

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.

```
const options = {
  name: 'customEvent',
  path: 'some/path',
  value: 1.5,
  customDimensions: { 1: 'some custom dimension value' },
}
await PiwikProSdk.trackCustomEvent(`custom_event`, 'custom_event_action', options);
```

Parameters:

- `category: string` *(required)* – 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.).
- `action: string` *(required)* – 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.
- `options` – custom event options, object containing five properties (all of them are optional):
  - `name: string` – 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.
  - `value: number` – float, 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.
  - `path: string` – the path under which this event occurred (it will be omitted in iOS application).
  - `customDimensions` – the object specifying *custom dimensions*.
  - `visitCustomVariables` – the object specifying *visit custom variables*.

For more resources, please visit documentation.

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 uses the `'fatal'` or `'caught'` *prefix*, and additionally the `'exception'` prefix if `isPrefixingOn()` option is enabled (`true`).

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

```
const options = {
  visitCustomVariables: { 4: { name: 'food', value: 'pizza' } },
  customDimensions: { 1: 'some custom dimension value' },
};
await PiwikProSdk.trackException('exception', false, options);
```

Parameters:

- `description: string` *(required)* – the exception message.
- `isFatal: boolean` *(required)* – true if an exception is fatal. Determines whether the exception prefix will be `'fatal'` or `'caught'`.
- `options` – exception tracking options, object containing two properties (all of them are optional):
  - `customDimensions` – the object specifying *custom dimensions*.
  - `visitCustomVariables` – the object specifying *visit custom variables*.

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 `isPrefixing()` option is enabled.

```
const options = {
  visitCustomVariables: { 4: { name: 'food', value: 'pizza' } },
  target: 'Photo',
};
await PiwikProSdk.trackSocialInteraction(`Like`, 'Facebook', options);
```

Parameters:

- `interaction: string` *(required)* – the social interaction, e.g. 'Like'.
- `network: string` *(required)* – social network associated with interaction, e.g. 'Facebook'.
- `options` – social interaction tracking options, object containing three properties (all of them are optional):
  - `target: string` – the target for which this interaction occurred, e.g. 'Photo'.
  - `customDimensions` – the object specifying *custom dimensions*.
  - `visitCustomVariables` – the object specifying *visit custom variables*.

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

Tracking downloads

Requires Analytics

You can track the downloads initiated by your application:

```
const options = {
  visitCustomVariables: 4: { name: 'food', value: 'pizza' },
  customDimensions: { 1: 'beta', 2: 'gamma', },
};
await PiwikProSdk.trackDownload(`http://your.server.com/bonusmap.zip`, options);
```

Parameters:

- `url: string` *(required)* – URL of the downloaded content.
- `options` – download tracking options, object containing two properties (all of them are optional):
  - `customDimensions` – object specifying *custom dimensions*.
  - `visitCustomVariables` – object specifying *visit custom variables*.

All downloads can be viewed in the corresponding section in the analytics panel.

Note: Generated URLs may differ between Android and iOS.

Tracking outlinks

Requires Analytics

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

```
const options = {
  visitCustomVariables: 4: { name: 'food', value: 'pizza' },
  customDimensions: { 1: 'beta', 2: 'gamma', },
};
await PiwikProSdk.trackOutlink(`http://your.server.com/bonusmap.zip`, options);
```

Parameters:

- URL *(required)* – outlink target. HTTPS, HTTP and FTP are valid.
- `options` – outlinks tracking options, object containing two properties (all of them are optional):
  - `customDimensions` – object specifying *custom dimensions*.
  - `visitCustomVariables` – object specifying *visit custom variables*.

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 `trackSearch` method:

```
const options = {
  category: `Movies`,
  count: 3,
  visitCustomVariables: 4: { name: 'food', value: 'pizza' },
  customDimensions: { 1: 'beta', 2: 'gamma', },
};
await PiwikProSdk.trackSearch('Space', options);
```

Parameters:

- `keyword: string` *(required)* – searched query that was used in the app.
- `options` – search tracking options, object containing four properties (all of them are optional):
  - `category: string` – search category.
  - `count: number` – 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.
  - `customDimensions` – object specifying *custom dimensions*.
  - `visitCustomVariables` – object specifying *visit custom variables*.

Tracking content impressions and interactions

Requires Analytics

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

```
const options = {
  piece: 'banner',
  target: 'https://www.dn.se/',
  visitCustomVariables: 4: { name: 'food', value: 'pizza' },
  customDimensions: { 1: 'beta', 2: 'gamma', },
};
await PiwikProSdk.trackImpression('Some content impression', options);
```

When the user interacts with the ad by tapping on it, you can also track it with a similar method:

```
const options = {
  piece: 'banner',
  target: 'https://www.dn.se/',
  visitCustomVariables: 4: { name: 'food', value: 'pizza' },
  customDimensions: { 1: 'beta', 2: 'gamma', },
};
await PiwikProSdk.trackInteraction('Some content interaction', options);
```

Parameters:

- `contentName: string` *(required)* – name of the content, e.g. 'Ad Foo Bar'.
- `options` – impression tracking options, object containing four properties (all of them are optional):
  - `piece: string` – actual content. For instance, path to the image, video, audio or any text.
  - `target: string` – the target of the content. For instance the URL of a landing page.
  - `customDimensions` – object specifying *custom dimensions*.
  - `visitCustomVariables` – object specifying *visit custom variables*.

Tracking goals

Requires Analytics

Goal tracking 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.

```
const options = {
  revenue: 30,
  visitCustomVariables: 4: { name: 'food', value: 'pizza' },
  customDimensions: { 1: 'beta', 2: 'gamma', },
};
await PiwikProSdk.trackGoal(1, options);
```

Parameters:

- `goal: number` *(required)* – tracking request will trigger a conversion for the goal of the website being tracked with this ID.
- `options` – goal tracking options, object containing three properties (all of them are optional):
  - `revenue: number` – monetary value that was generated as revenue by this goal conversion.
  - `customDimensions` – object specifying *custom dimensions*.
  - `visitCustomVariables` – object specifying *visit custom variables*.

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.

```
const options: TrackEcommerceOptions = {
  discount: 0,
  shipping: 1000,
  subTotal: 33110,
  tax: 9890,
  items: [
    {
      sku: '0123456789012',
      category: "Men's T-shirts",
      name: 'Polo T-shirt',
      price: 3000,
      quantity: 2,
    },
  ],
  visitCustomVariables: 4: { name: 'food', value: 'pizza' },
  customDimensions: { 1: 'beta', 2: 'gamma', },
};
await PiwikProSdk.trackEcommerce('order_1', 124144, options);
```

Parameters:

- `orderId: string` *(required)* – unique string identifying the order.
- `grandTotal: number` *(required)* – total amount of the order, in cents.
- `options` – goal tracking options, object containing five properties (all of them are optional):
  - `subTotal: number` – subtotal (net price) for the order, in cents.
  - `tax: number` – tax for the order, in cents.
  - `shipping: number` – shipping for the order, in cents.
  - `discount: number` – discount for the order, in cents.
  - `items` – items included in the order, array of objects containing five required properties:
    - `sku: string` – identifier of the item.
    - `name: string` – name of the item.
    - `category: string` – category of the item.
    - `price: string` – price of the single item, in cents.
    - `quantity: string` – quantity of the item.
  - `customDimensions` – object specifying *custom dimensions*.
  - `visitCustomVariables` – object specifying *visit custom variables*.

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:

```
const options = {
  visitCustomVariables: 4: { name: 'food', value: 'pizza' },
  customDimensions: { 1: 'beta', 2: 'gamma', },
```

(continues on next page)

(continued from previous page)

```
};
await PiwikProSdk.trackCampaign('http://example.org/offer.html?pk_campaign=Email-
↪SummerDeals&pk_keyword=LearnMore', options);
```

Parameters:

- `url: string` *(required)* – the campaign URL. HTTPS, HTTP and FTP are valid, however, the URL must contain campaign name and keyword parameters.
- `options` – campaign tracking options, object containing two properties (all of them are optional):
  - `customDimensions` – object specifying *custom dimensions*.
  - `visitCustomVariables` – object specifying *visit custom variables*.

Note: On iOS the campaign information will be sent to the server together with the next analytics event.

Tracking custom variables

The feature will soon be disabled. We recommend using custom dimensions *instead.*

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. occurred 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 `screenCustomVariables` object, for the visit scope – `visitCustomVariables` in the screen tracking method options:

```
const options = {
  screenCustomVariables: { 4: { name: 'food', value: 'pizza' } },
  visitCustomVariables: { 5: { name: 'drink', value: 'water' } },
};
await PiwikProSdk.trackScreen(`your_screen_path`, options);
```

Please note that for the *Default custom variables* option, use the custom variables of the visit scope with indexes 1-3. Custom variables of each scope is the object with the following format:

```
const customVariables = {
  4: { name: 'food', value: 'pizza' },
  5: { name: 'drink', value: 'water' },
}
```

where:

- `index: number`, the key *(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 with 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`.
- `name: string` *(required)* – the name of a specific custom variable such as 'User type' (Limited to 200 characters).

- `value: string` *(required)* – the value of a specific custom variable such as 'Customer' (Limited to 200 characters).

Tracking custom dimensions

Requires Analytics

To track a custom key-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 actions (see the documentation of other tracking methods), for example:

```
const customDimensions = {
  1: 'dashboard',
  2: 'menu',
}
await PiwikProSdk.trackScreen(`your_screen_path`, { customDimensions });
```

`1` and `2` are dimension IDs. `dashboard`, `menu` are the dimension values for the tracked screen view 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 `trackProfileAttributes` method:

```
const profileAttributes: TrackProfileAttributes = [
  { name: 'food', value: 'pizza' },
  { name: 'drink', value: 'water' },
];
// Profile attributes can be also a single object:
// const profileAttributes: TrackProfileAttributes = { name: 'food', value: 'pizza' };
await PiwikProSdk.trackProfileAttributes(profileAttributes);
```

Parameters:

- `profileAttributes` – an object or an array of objects with two required properties:
  - `name: string` *(required)* – profile attribute name.
  - `value: string` *(required)* – the profile attribute value.

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

- `WEBSITE_ID` – always sent.
- `USER_ID` – if set. Read more about the *User ID*.
- `EMAIL` – if set. Read more about the *email*.
- `VISITOR_ID` – always sent, ID of the mobile application user, generated by the SDK.

- `DEVICE_ID` – Advertising ID that, by default, is fetched automatically when the tracker instance is created (only on Android).

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 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 the Audience Manager. You can get user profile attributes in the following manner:

```
const attributes = await PiwikProSdk.getProfileAttributes();
console.log(attributes);
// {"device_type": "desktop", ...}
```

Returns:

- `attributes: object` – dictionary of key-value pairs, where each pair represents the attribute name (key) and value. In case of error (for example when user profile does not yet exist), returns error message.

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:

```
const audienceId = 'a83d4aac-faa6-4746-96eb-5ac110083f8e';
const isMember = await PiwikProSdk.checkAudienceMembership(audienceId);
console.log(isMember);
// true
```

Parameters:

- `audienceId: string` *(required)* – ID of the audience (Audience Manager -> Audiences).

Returns:

- `isMember: boolean` – value indicating whether user belongs to the audience with given ID or error message if an error occurred.

3.4.4 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, 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 user ID. In order to set user ID use the `setUserId` method:

```
await PiwikProSdk.setUserId("John Doe");
```

Parameters:

- `userId: string` *(required)* – any non-empty unique string identifying the user. Passing null will delete the current user ID.

You can obtain current user ID value with `getUserId`:

```
const currentUserId = await PiwikProSdk.getUserId();
```

Returns:

- `userId: string` – current user ID.

User email address

Used only by Audience Manager

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

```
await PiwikProSdk.setUserEmail('john@doe.com');
```

Parameters:

- `email: string` *(required)* – non-empty 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.

You can obtain current user email value with `getUserEmail`:

```
const currentUserEmail = await PiwikProSdk.getUserEmail();
```

Returns:

- `email: string` – current user email.

Visitor ID

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

```
await PiwikProSdk.setVisitorId("0123456789abcdef");
```

Parameters:

- `visitorId: string` *(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 *user ID* instead of visitor ID.

You can check current visitor ID value with `getVisitorId`:

```
const currentVisitorId = await PiwikProSdk.getVisitorId();
```

Returns:

- `visitorId: string` – current visitor ID.

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`:

```
await PiwikProSdk.setSessionTimeout(1800);
```

Parameters:

- `sessionTimeout: number` *(required)* – session timeout time in seconds. Default: 1800 seconds (30 minutes).

You can obtain current `sessionTimeout` value with `getSessionTimeout`:

```
const currentSessionTimeout = await PiwikProSdk.getSessionTimeout();
console.log(currentSessionTimeout); // 1800
```

Returns:

- `sessionTimeout: number` – current session timeout value in seconds.

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

```
await PiwikProSdk.startNewSession();
```

Dispatching

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

```
const dispatchInterval = 25; // 25 seconds
await PiwikProSdk.setDispatchInterval(dispatchInterval);
```

Parameters:

- `dispatchInterval: number` *(required)* – new dispatch interval (in seconds).

If `dispatchValue` is equal to `0` then events will be dispatched immediately. When its value is negative then events will not be dispatched automatically. This gives you full control over dispatch process using manual `dispatch`:

```
await PiwikProSdk.dispatch();
```

You can obtain current `dispatchInterval` value with `getDispatchInterval`:

```
const currentDispatchInterval = await PiwikProSdk.getDispatchInterval();
```

Returns:

- `dispatchInterval: number` – current dispatch interval (in seconds) or negative number if automatic dispatch has been disabled.

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:

```
await PiwikProSdk.setIncludeDefaultCustomVariables(true);
```

Parameters:

- `includeDefaultCustomVariables: boolean` *(required)* – flag that determines whether default custom variables should be added to each tracking event.

The status of the option can be checked with `getIncludeDefaultCustomVariables`:

```
const includeDefaultCustomVariables = await PiwikProSdk.
↪getIncludeDefaultCustomVariables();
```

Returns:

- `includeDefaultCustomVariables: boolean` – flag that determines whether default custom variables should be added to each tracking event.

Opt out

You can set 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 by default is set to `false`. To enable the app-level opt-out, use:

```
await PiwikProSdk.setOptOut(true);
```

Parameters:

- `optOut: boolean` *(required)* – flag that determines whether opt-out is enabled.

You can obtain current `optOut` value with `getOptOut`:

```
const currentOptOutState = await PiwikProSdk.getOptOut();
console.log(currentOptOutState); // false
```

Returns:

- `optOut: boolean` – current opt-out state.

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 `dryRun` flag, use:

```
await PiwikProSdk.setDryRun(true);
```

Parameters:

- `dryRun: boolean` *(required)* – flag that determines whether dry run is enabled.

You can obtain current `dryRun` value with `getDryRun`:

```
const currentDryRunState = await PiwikProSdk.getDryRun();
```

Returns:

- `dryRun: boolean` – current dry run state.

Prefixing

In case of tracking events like screen view, exception or social interaction event path in the tracker will contain corresponding prefix. You can disable prefixing with:

```
await PiwikProSdk.setPrefixing(false);
```

Parameters:

- `prefixingEnabled: boolean` *(required)* – flag that determines whether prefixing is enabled

You can also check the prefixing status using the `isPrefixingOn` method:

```
const currentPrefixingState = await PiwikProSdk.isPrefixingOn();
console.log(currentPrefixingState); // false
```

Returns:

- `prefixingEnabled: boolean` – current prefixing state.

3.4.5 License

MIT

CHAPTER 4

Integrations

If you need to set up Piwik PRO on a PWA or AMP or collect data using web log analytics, we've got solutions for you. See the listed documentation and apply it to your specific project.

See more

4.1 Accelerated Mobile Pages integration

Accelerated Mobile Pages (AMP) is an open source framework designed to optimize browsing on mobile devices. This technology can render static content pages much faster than traditional methods. To do that AMP removed the possibility of executing JavaScript on such pages (excluding few approved libraries), so traditional analytic scripts won't work on such pages. You can still measure user engagement using an amp-analytics library.

4.1.1 Basic setup

This setup allows you to track page views. Copy following code to your AMP page while replacing:

- `<INSTANCE_DOMAIN>` - PPAS instance domain (e.g. `analytics.example.com`)
- `<APP_ID>` - PPAS application ID (e.g. `12345678-1234-1234-1234-1234567890ab`)
- `<TRACKER_HASH>` - Cookie hash generated by JavaScript Tracking Client. Check *how to get cookie hash* section for detailed information.

```
<script async custom-element="amp-analytics" src="https://cdn.ampproject.org/v0/amp-
↪analytics-0.1.js"></script>
<amp-analytics type="ppasanalytics">
  <script type="application/json">
  {
    "vars": {
      "host": "<INSTANCE_DOMAIN>",
      "website_id": "<APP_ID>",
      "website_hash": "<TRACKER_HASH>"
```

(continues on next page)

(continued from previous page)

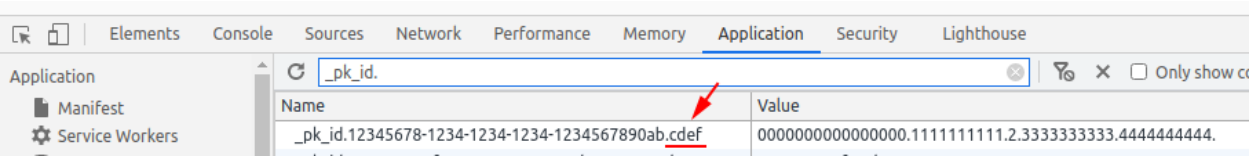
```
    }
  }
  </script>
</amp-analytics>
```

4.1.2 How to get JavaScript Tracking Client cookie hash

If there is no non-AMP page tracked by traditional JavaScript Tracking Client, this value may be removed from configuration or left empty. It's used to guarantee that same cookie will be used by AMP and non-AMP pages on client domain. This value should be taken from the name of the ID cookie generated by JavaScript Tracking Client. Each JavaScript Tracking Client generates unique cookie name based on its configuration. Follow these instructions to get hash from cookie generated by JavaScript Tracking Client:

- Setup JavaScript Tracking Client on non-AMP page (if it was not done already).
- Open tracked page in the browser.
- Open developer tools in the browser and look for cookie starting with `_pk_id.`. Cookie name should look similar to this: `_pk_id.12345678-1234-1234-1234-1234567890ab.cdef`. The part after first dot is the value of App ID of the cookie (if there are multiple cookies starting with `_pk_id.` it may be used to identify correct cookie). After second dot you'll find the cookie hash generated by JavaScript Tracking Client (in the example its value is `cdef`). Copy this part and replace `<TRACKER_HASH>` with it.

Here you can see how to look for JavaScript Tracking Client cookie in Google Chrome developer tools:



4.1.3 Tracking custom events

To track custom event you should attach a trigger on the interactive page element and define event values. To do that add to the configuration the triggers section and set up event trigger.

This example will send custom event when page element using "mybutton" ID will be clicked:

```
<amp-analytics type="ppasanalytics">
  <script type="application/json">
  {
    "vars": {
      "host": <instance_domain>,
      "website_id": <app_id>,
      "website_hash": <tracker_hash>
    },
    "triggers": {
      "exampleEvent": {
        "selector": "#mybutton",
        "on": "click",
        "request": "customevent",
        "vars": {
          "event_category": "buttons",
          "event_action": "click",
```

(continues on next page)

(continued from previous page)

```
          "event_name": "testButton"
        }
      }
    }
  }
  </script>
</amp-analytics>
```

These are parameters used by custom event:

- "selector" - CSS selector for element that should be watched
- "on" - HTML event type
- "vars" - Variables that should be used by this event. Custom events expect:
  - "event_category" - required
  - "event_action" - required
  - "event_name" - optional
  - "event_value" - optional

4.1.4 Tracking download events

To track download event attach trigger to a link in a similar way to *custom event*.

This example will send download event when page element using "mydownload" ID will be clicked:

```
<amp-analytics type="ppasanalytics">
  <script type="application/json">
  {
    "vars": {
      "host": <instance_domain>,
      "website_id": <app_id>,
      "website_hash": <tracker_hash>
    },
    "triggers": {
      "exampleEvent": {
        "selector": "#mydownload",
        "on": "click",
        "request": "download",
        "vars": {
          "download_url": "https://example.com/whitepaper.pdf"
        }
      }
    }
  }
  </script>
</amp-analytics>
```

These are parameters used by download event:

- "selector" - CSS selector for element that should be watched
- "on" - HTML event type
- "vars" - Variables that should be used by this event. Custom events expect:

  - "download_url" - required

4.1.5 Tracking outlink events

To track outlink event attach trigger to a link in a similar way to *custom event*.

This example will send outlink event when page element using "myoutlink" ID will be clicked:

```
<amp-analytics type="ppasanalytics">
  <script type="application/json">
  {
    "vars": {
      "host": <instance_domain>,
      "website_id": <app_id>,
      "website_hash": <tracker_hash>
    },
    "triggers": {
      "exampleEvent": {
        "selector": "#myoutlink",
        "on": "click",
        "request": "outlink",
        "vars": {
          "outlink_url": "https://another-site.com/"
        }
      }
    }
  }
  </script>
</amp-analytics>
```

These are parameters used by outlink event:

- "selector" - CSS selector for element that should be watched
- "on" - HTML event type
- "vars" - Variables that should be used by this event. Custom events expect:
  - "outlink_url" - required

4.1.6 Tracking goal conversions

To track goal conversion attach trigger to a link in a similar way to *custom event*.

This example will send goal conversion when page element using "mygoal" ID will be clicked:

```
<amp-analytics type="ppasanalytics">
  <script type="application/json">
  {
    "vars": {
      "host": <instance_domain>,
      "website_id": <app_id>,
      "website_hash": <tracker_hash>
    },
    "triggers": {
      "exampleEvent": {
        "selector": "#mygoal",
        "on": "click",
```

(continues on next page)

(continued from previous page)

```
        "request": "goal",
        "vars": {
          "goal_id": "1",
          "revenue": "59.99"
        }
      }
    }
  }
  </script>
</amp-analytics>
```

These are parameters used by goal event:

- "selector" - CSS selector for element that should be watched
- "on" - HTML event type
- "vars" - Variables that should be used by this event. Custom events expect:
  - "goal_id" - required
  - "revenue" - optional

4.1.7 Track internal search events

To track internal search event attach trigger to a link in a similar way to *custom event*.

This example will send internal search event when page element using "mysearch" ID will be clicked:

```
<amp-analytics type="ppasanalytics">
  <script type="application/json">
  {
    "vars": {
      "host": <instance_domain>,
      "website_id": <app_id>,
      "website_hash": <tracker_hash>
    },
    "triggers": {
      "exampleEvent": {
        "selector": "#mysearch",
        "on": "click",
        "request": "search",
        "vars": {
          "search_keyword": "apple",
          "search_category": "fruits",
          "search_result_count": "10",
        }
      }
    }
  }
  </script>
</amp-analytics>
```

These are parameters used by internal search event:

- "selector" - CSS selector for element that should be watched
- "on" - HTML event type

- "vars" - Variables that should be used by this event. Custom events expect:
  - "search_keyword" - required
  - "search_category" - required
  - "search_result_count" - optional

4.1.8 Complete page example

This example shows complete AMP page with 2 buttons. It will send page view, custom event and goal conversion.

```
<!doctype html>
<html amp lang="en">
    <head>
        <meta charset="utf-8">
        <title>AMP example page</title>
        <meta name="viewport" content="width=device-width">
        <link rel="canonical" href="example.html">

        <style amp-boilerplate>body{-webkit-animation:-amp-start 8s steps(1,end) 0s 1␣
↪normal both;-moz-animation:-amp-start 8s steps(1,end) 0s 1 normal both;-ms-
↪animation:-amp-start 8s steps(1,end) 0s 1 normal both;animation:-amp-start 8s␣
↪steps(1,end) 0s 1 normal both}@-webkit-keyframes -amp-start{from{visibility:hidden}
↪to{visibility:visible}}@-moz-keyframes -amp-start{from{visibility:hidden}to
↪{visibility:visible}}@-ms-keyframes -amp-start{from{visibility:hidden}to
↪{visibility:visible}}@-o-keyframes -amp-start{from{visibility:hidden}to
↪{visibility:visible}}@keyframes -amp-start{from{visibility:hidden}to
↪{visibility:visible}}</style><noscript><style amp-boilerplate>body{-webkit-
↪animation:none;-moz-animation:none;-ms-animation:none;animation:none}</style></
↪noscript>

        <script async src="https://cdn.ampproject.org/v0.js"></script>
        <script async custom-element="amp-analytics" src="https://cdn.ampproject.org/
↪v0/amp-analytics-0.1.js"></script>
    </head>
    <body>
        <amp-analytics type="ppasanalytics">
            <script type="application/json">
                {
                    "vars": {
                        "host": "example.piwik.pro",
                        "website_id": "12345678-1234-1234-1234-1234567890ab",
                        "website_hash": "cdef"
                    },
                    "triggers": {
                        "trackRecommendation": {
                            "on": "click",
                            "selector": "#recommend",
                            "request": "customevent",
                            "vars": {
                                "event_category": "social",
                                "event_action": "recommend",
                                "event_name": "News letter"
                            }
                        },
                        "trackSubscription": {
                            "on": "click",
```

(continues on next page)

(continued from previous page)

```
                            "selector": "#subscribe",
                            "request": "goal",
                            "vars": {
                                "goal_id": "1"
                            }
                        }
                    }
                }
            </script>
        </amp-analytics>

        <h1>Welcome</h1>
        <div>
            <button id="recommend">Share this page with friends</button>
        </div>
        <div>
            <button id="subscribe">Subscribe to news letter</button>
        </div>
    </body>
</html>
```

4.2 Progressive Web Applications integration

If you're building an application that works offline, then understanding how users are interacting with your app when they don't have connectivity is crucial to optimizing that experience.

Analytics providers like Piwik PRO typically require a network connection to send data to their servers, which means if connectivity is unavailable, those requests will fail and those interactions will be missing from your analytics reports. It'll be like they never happened.

Our integration with Progressive Web Applications solves this problem for Piwik PRO users by leveraging Service Worker's ability to detect failed requests.

Piwik PRO receives all data via HTTP requests to the Analytics, which means a Service Worker script can add a fetch handler to detect failed requests sent to the Analytics. It can store these requests in IndexedDB and then retry them later once connectivity is restored.

The PWA module for Piwik PRO does exactly this. It also adds fetch handlers to cache the ppms.js and the container scripts, so they can also be run offline. Lastly, when failed requests are retried, the module also automatically sets (or updates) the `cdt` in the request payload to ensure timestamps in Piwik PRO reflect the time of the original user interaction.

4.2.1 Enabling the Piwik PRO module for Progressive web applications

To enable collecting data from your PWAs using Piwik PRO Analytics, call the initialize() method in your service worker:

```
import PiwikPro from '@piwikpro/pwa-piwik-pro';

PiwikPro.initialize({
    containerURL: 'example.com',
    containerId: '12345678-1234-1234-1234-1234567890ab'
});
```

This is all that's required to queue and retry failed requests to Piwik PRO, and it's the simplest way to get Piwik PRO working offline.

However, if using only the code above, the retried requests are indistinguishable from requests that succeed on the first try. This means you'll receive all the interaction data from offline users, but you won't be able to tell which interactions occurred while the user was offline.

To address this concern, you can use one of the optional methods described below.

4.2.2 Enable automatic tracking of the status of the user's Internet connection

If you want to be able to differentiate retried requests from non-retried requests, you can use a command that will start automatic tracing of the internet connection status. With this solution, when the internet is lost, a Custom Event will be generated containing information about the status of the internet connection.

In your application, include the default PiwikPro object in the highest level application module. ie `index`.

```
import PiwikPro from '@piwikpro/pwa-piwik-pro';

PiwikPro.enableInternetConnectionTracking();
```

4.2.3 Enable automatic tracking of the app install event

If you want to additionally track as a Custom Event the information about when your customers have installed the application, you can do so using the method:

```
import PiwikPro from '@piwikpro/pwa-piwik-pro';

PiwikPro.enableInstallTracking();
```

4.3 Web Log Analytics

Log analytics is a python script, that allows you to import logs of common web servers (nginx, apache, iss and more) directly to Piwik PRO. It's a free software available under GPLv3 license, available on GitHub and PyPi

4.3.1 Installation of the log analytics script

You can install the script in one of two ways:

- By using Python's package manager - *pip* - this is the preferred method
- By downloading the script to your machine manually

Python package

```
pip install piwik-pro-log-analytics
```

Python script

```
curl https://raw.githubusercontent.com/PiwikPRO/log-analytics/master/piwik_pro_log_
↪analytics/import_logs.py > import_logs.py
chmod +x import_logs.py
```

4.3.2 Set up log import

You need to run the Log Importer tool with the correct parameters. Some of them must be present, while others are optional.

Sample command:

Python package

```
piwik_pro_log_analytics --url=https://demo.piwik.pro --client-id=*** --client-
↪secret=*** --enable-static --enable-bots --show-progress --idsite=*** --recorders=2␣
↪sample.log
```

Python script

```
./import_logs.py --url=https://demo.piwik.pro --client-id=*** --client-secret=*** --
↪enable-static --enable-bots --show-progress --idsite=*** --recorders=2 sample.log
```

--url=https://demo.piwik.pro
: This is a mandatory parameter which points to the location of your Piwik instance

--client-id=***
: Part of API credentials. They can be obtained from PPAS (check how to do it).

--client-secret=***
: Part of API credentials. They can be obtained from PPAS (check how to do it).

--idsite=***
: Defines the Site ID of the website (eg. *99e33528-8da4-46d8-be90-a62bfb3a7bba*).

There are many other options that can be added to this script, which are described in the *Add parameters to log import*.

If you plan to import logs on a regular basis it is advised to setup a scheduled job using a tool such as CRON.

4.3.3 Exclude log lines

There are several methods allowing you to exclude particular log lines or visitors from being tracked:

- You can exclude specific IP addresses or IP ranges from being tracked. To configure excluded IPs, log into Piwik as a superuser, then click Administration > Websites.
- Excluding lines from specific IP or IP ranges - this can be done the same way as in the default tracking method in Piwik (by adding an excluded IP or IP range in the Administration -> Websites menu)
- You can exclude visitors based on their User Agent HTTP headers by using **–useragent-exclude**
- You can also provide a sole hostname that you would like to import from. This means that all the logs from other hosts will be ignored. The parameter allowing this is: **–hostname**
- It is also possible to exclude specific log lines where the URL path matches a particular URL path. See the option **–exclude-path**

If you need to add multiple paths or hostnames, you will need to add these parameters multiple times.

4.3.4 Add parameters to log import

The Web Log Analytics script does not track static files (JS, CSS, images, etc.). It also excludes all bot traffic.

Use the following commands to enable tracking of these elements:

- **–enable-bots** This enables tracking of search/spam bots via Piwik. Just add a custom variable with the bot's name. The User-agent field is examined to determine whether a log line comes from a bot or a real user.
- **–enable-static** Specifies tracking of all static files (images, JS, CSS) in Piwik PRO.
- **–enable-http-redirects** This tracks HTTP redirects as page views, with a custom title and custom variable.
- **–enable-reverse-dns** Activates reverse DNS, which is used in generating the Visitors > Providers report. NOTE: this may lead to a serious drop in performance as reverse DNS is very slow.
- **–recorders=N** Sets a specific number of threads. We recommend matching it to the number of CPU cores in the system.
- **–enable-bulk-tracking** Enables bulk tracking mode. Tracking requests will be bunched up and send using bulk request.
- **–recorder-max-payload-size=N** When importer uses the Piwik PRO bulk tracking feature in order to boost speed (option **–enable-bulk-tracking**), this option configures max number of tracking requests that bulk request can contain. Adjust the number of pageviews (or log lines) to see what generates the best performance.

More information about log import parameters can be found using the help parameter:

Python package

```
piwik_pro_log_analytics --help
```

Python script

```
./import_logs.py --help
```

4.3.5 Import data with server log analytics and standard JavaScript simultaneously

JavaScript Tracking Client and web server log file analytics can be used at the same time, on the condition that data is recorded for each method in a separate Piwik PRO website.

To avoid double counts of visits, follow these steps:

1. Create a new website in Piwik PRO with a name, for example, example.com (log files).
2. Record the website ID of this new website. The website ID will be used for importing log file data.
3. In the command line, force all requests from log files to be recorded in a specific website ID via the command –idsite=X.

4.3.6 Technical requirements

Technical requirements for running Web Log Analytics:

- Access to the server or server logs - for example via SSH
- Python 3.6+ - older versions (e.g. 2.6, 2.7 or 3.5) are not supported. Most often you'll want to import your data straight from the server where it is created. To do this, you'll need to be able to run a Python script on the machine that will send the logs to Piwik PRO.
- Log Analytics script - this is a script written in Python ensuring that logs are sent to your Piwik PRO instance, available on GitHub

Supported log formats:

- all default log formats for: Nginx, Apache, IIS, Tomcat

- all common log formats like: NCSA Common log format, Extended log format, W3C Extended log files, Nginx JSON
- log files of some popular Cloud Saas services: Amazon CloudFront logs, Amazon S3 logs
- streaming media server log files such as: Icecast
- log files with and without the virtual host will be imported

This script does not directly support importing logs from log aggregation tools, like Grafana Loki or ELK. If you'd like to import logs from one of those, you need to download them to the disk first.

4.3.7 Performance considerations & rate limiting

The script needs CPU to read and parse the log files, but it is usually Piwik PRO server itself which will limit the import speed due to network latency. To improve performance, you can use the **–recorders** option to specify the number of parallel threads which will import hits into Piwik PRO. By default we are using one recorder, but you can increase this value until you achieve satisfying speed.

If you are Piwik PRO Core user, please make sure, that you are not hitting rate limits, by using **–sleep-between-requests-ms** flag to slow down the import process.

CHAPTER 5

Other

If you need help with topics other than web API, JS API or SDKs, see this section. More help articles are also available in our help center.

See more

5.1 Content Security Policy (CSP)

5.1.1 Introduction

Specifying Content Security Policy is a common way to secure web applications. This mechanism allows specifying which scripts and styles can execute on page. It can be done either by adding a `Content-Security-Policy` header or an appropriate meta tag.

You can read about Consent Security Policy here: https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

5.1.2 Content Security Policy nonce configuration

It is common to allow only scripts and styles that were received from known domains or ones that have special nonce attribute. Nonce mechanism relies on two steps, defining nonce value in Content Security Policy and placing nonce value as an attribute in styles and scripts.

Defining nonce in Content Security Policy settings

Nonce mechanism requires additional definition in `script-src` directive of Content Security Policy:

```
script-src <your-sources> 'nonce-INSERT_VALID_NONCE_VALUE';
```

Note: **Note:** Nonce value should be generated on the server-side. Its value should be different for each request. Please note that we leave here space for your permitted sources `<your-sources>`.

Add nonce to container code

Consequently, default container code requires following modifications to work:

Asynchronous snippet: In this container code the following changes (highlighted) are required:

```
  <script type="text/javascript" nonce="INSERT_VALID_NONCE_VALUE">
      (function(window, document, dataLayerName, id) {
      window[dataLayerName]=window[dataLayerName]||[],window[dataLayerName].push(
↪{start:(new Date).getTime(),event:"stg.start"});
      var scripts=document.getElementsByTagName('script')[0],tags=document.
↪createElement('script');
      function stgCreateCookie(a,b,c){var d="";if(c){var e=new Date;e.setTime(e.
↪getTime()+24*c*60*60*1e3),d=";expires="+e.toUTCString()}document.cookie=a+"="+b+d+";
↪ path=/"}
      var isStgDebug=(window.location.href.match("stg_debug")||document.cookie.match(
↪"stg_debug"))&&!window.location.href.match("stg_disable_debug");
      stgCreateCookie("stg_debug",isStgDebug?1:"",isStgDebug?14:-1);
      var qP=[];dataLayerName!=="dataLayer"&&qP.push("data_layer_name=
↪"+dataLayerName),isStgDebug&&qP.push("stg_debug");
      var qPString=qP.length>0?("?"+qP.join("&")):"";
      tags.async=!0,tags.src="//client.containers.piwik.pro/"+id+".js"+qPString,
      scripts.parentNode.insertBefore(tags,scripts);
      !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,"ppms",["tm","cm"]);
      })(window, document, 'dataLayer', 'feacd61d-0232-40a1-96c3-7e469f7bfa7f');
  </script>
  <noscript>
      <iframe src="//client.containers.piwik.pro/feacd61d-0232-40a1-96c3-7e469f7bfa7f/
↪noscript.html" height="0" width="0" style="display:none;visibility:hidden"></iframe>
  </noscript>
```

Synchronous snippet: In this container code the following changes (highlighted) are required:

```
  <script type="text/javascript" nonce="INSERT_VALID_NONCE_VALUE">
      (function(window, document, dataLayerName, id) {
      function stgCreateCookie(a,b,c){var d="";if(c){var e=new Date;e.setTime(e.
↪getTime()+24*c*60*60*1e3),d=";expires="+e.toUTCString()}document.cookie=a+"="+b+d+";
↪ path=/"}
      var isStgDebug=(window.location.href.match("stg_debug")||document.cookie.match(
↪"stg_debug"))&&!window.location.href.match("stg_disable_debug");
      stgCreateCookie("stg_debug",isStgDebug?1:"",isStgDebug?14:-1);
      var qP=[];dataLayerName!=="dataLayer"&&qP.push("data_layer_name=
↪"+dataLayerName),isStgDebug&&qP.push("stg_debug");
      var qPString=qP.length>0?("?"+qP.join("&")):"";
      document.write('<script src="//client.containers.piwik.pro/'+id+'.sync.js' +␣
↪qPString + '" nonce="INSERT_VALID_NONCE_VALUE"></' + 'script>');
      })(window, document, 'dataLayer', 'feacd61d-0232-40a1-96c3-7e469f7bfa7f');
  </script>
```

Note: **Note:** All that is needed for Tag Manager to work is to replace `INSERT_VALID_NONCE_VALUE` with generated nonce value. It should be done twice for both asynchronous and synchronous snippet.

5.1.3 Adjust tags to work with Content Security Policy

Asynchronous tags: in most cases there should not be any change required to make asynchronous tags work. Tag Manager will automatically insert nonce attribute to all fired tags. Only exceptions is when Your tag adds other scripts/styles on page by itself - in such case, You should add nonce attribute manually.

Synchronous tags: since synchronous tags have to fire before whole page is loaded, following procedure is recommended.

This procedure is recommended:

1. Create new variable with value of nonce parameter. It is not required to create nonce variable in admin panel. Just pushing it on dataLayer before script is executed is enough.

```
window.dataLayer.push({
nonce: INSERT_VALID_NONCE_VALUE
});
```

2. Use created variable as value for nonce attribute like follows:

```
<script nonce="{{ nonce }}">
  console.log("I'm synchronous tag!");
  document.write('<p id="synchronous-tag">I was inserted by synchronous tag</p>');
</script>
```

Note: **Note:** Finally, not all 3rd party tools that are available as built-in templates are adjusted to work with Content Security Policy. This includes e.g. Google Analytics. In such cases, please refer to documentation of each respective tool (e.g. https://developers.google.com/web/fundamentals/security/csp).

5.1.4 Tag Manager debugger

To load all necessary assets from Tag Manager debugger you need to define source with `img-src`, `font-src` and `style-src` directives:

```
img-src <your-sources> client.containers.piwik.pro;
font-src <your-sources> client.containers.piwik.pro;
style-src <your-sources> client.containers.piwik.pro;
```

5.1.5 Consent Manager form assets

If your website is GDPR compliant then you need to describe `connect-src`, `style-src` and `img-src` directives:

```
connect-src <your-sources> client.piwik.pro client.containers.piwik.pro;
style-src <your-sources> 'nonce-INSERT_VALID_NONCE_VALUE';
```

Note: Please note that we define here tracking domain **client.piwik.pro** for collecting visitor consents and container domain **client.containers.piwik.pro** for fetching consent form assets.

5.1.6 Consent Manager's data subject request widget

When using a data subject request widget, you need to add a nonce attribute to its `<script>` tag.

```
<div id="ppms_cm_data_subject" class="ppms_cm_data_subject_widget__wrapper" data-
↪editor-centralize="true" data-main-container="true" data-root="true">
    <h3 id="ppms_cm_data_subject_header" class="header3">Data requests</h3>
    <p id="ppms_cm_data_subject_paragraph" class="paragraph">
        Please select below the type of data request along with any special requests␣
↪in the body of the message. (...)
    </p>
    <form id="ppms_cm_data_subject_form" class="ppms_cm_data_subject_form" data-
↪disable-select="true">
        ...
    </form>
    <script nonce="INSERT_VALID_NONCE_VALUE">
        ...
    </script>
</div>
```

5.1.7 Tracking with custom domain

If your tracking domain is custom, then you need to define it with `img-src` and `script-src` directives:

```
img-src <your-sources> your-custom-cpp-domain.com;
script-src <your-sources> your-custom-cpp-domain.com;
```

5.1.8 Example Content Security Policy definition

Following example configuration of CSP assumes:

- Client's website address: **client.com**
- Consent Manager is enabled for the website
- Client's organization name in Piwik PRO: **client**
- Client's container domain: **client.containers.piwik.pro**
- Client has Piwik PRO tag with default tracking domain: **client.piwik.pro**
- Nonce value: **nceIOfn39fn3e9h3sd**
- Configuration allows `'self'` source which is: **client.com**

```
Content-Security-Policy: default-src 'self';
                         script-src  'self' client.piwik.pro 'nonce-nceIOfn39fn3e9h3sd
↪';
                         connect-src 'self' client.containers.piwik.pro client.piwik.
↪pro;
                         img-src     'self' client.containers.piwik.pro client.piwik.
↪pro;
```

(continues on next page)

(continued from previous page)

```
                        font-src    'self' client.containers.piwik.pro;
                        style-src   'self' client.containers.piwik.pro 'nonce-
↪nceIOfn39fn3e9h3sd';
```

5.2 Event processing

Here's the order in which our tracker detects event types. The detection process is completed with the first match. After that it stops and no further matching is performed.

Step 1: Goal conversion. First step in the event type detection process is a check for conversion signs in the tracked event. If your tracked event had an `idgoal` parameter with the ID value other than `0` then it will be marked as `GoalConversion` type.

Step 2: Ping. Then the `ping` parameter is analyzed. Depending on it's value one of the following ping types will be assigned:

- values `1`, `2` and `3` result in `Heartbeat` ping type
- value of `4` results in `Deanonymization` ping type
- value of `5` results in `PagePerformance` ping type
- value of `6` results in `CustomPing` ping type

Values outside of that range will cause an error which results in `BrokenEvent`.

Step 3: Download. Event will be categorized as `Download` when `download` parameter of the tracked event is provided.

Step 4: Outlink. Event will be categorized as `Outlink` when `link` parameter of the tracked event is provided.

Step 5: Consent form impression. Event will be categorized as `ConsentFormImpression` when `e_c` parameter of the tracked event has value `consent_form_impression` assigned.

Step 6: Consent form click. Event will be categorized as `ConsentFormClick` when `e_c` parameter of the tracked event has value `consent_form_click` assigned.

Step 7: Consent decision. Event will be categorized as `ConsentDecision` when `e_c` parameter of the tracked event has value `consent_decision` assigned.

Step 8: SharePoint SharePoint event type detection is a bit more complicated. For the event to be categorized as `SharePoint` type two things must occur:

- Custom Variable 1 key has to be equal to `ppas.sharepoint.plugin`
- `e_c` parameter has to be equal to either `download` or `search`

Step 9: Custom event. Event will be categorized as `Custom` when `e_c` and `e_a` parameters of the tracked event are provided.

Step 10: Content interaction. Event will be categorized as `ContentInteraction` when `c_i` and `c_n` parameters of the tracked event are provided.

Step 11: Content impression. Event will be categorized as `ContentImpression` when only `c_n` parameter of the tracked event is provided (and `c_i` is not).

Step 12: Cart update. Event will be categorized as `CartUpdated` when `idgoal` parameter of the tracked event is equal to `0` and `ec_id` parameter is NOT provided.

Step 13: Order completed. Event will be categorized as `OrderCompleted` when `idgoal` parameter of the tracked event is equal to `0` but also `ec_id` parameter is provided.

Step 14: Site search. Event will be categorized as `Search` when either `search` parameter of the tracked event is provided or a search term was detected in the tracked url (provided as the `url` parameter).

Step 15: Page view. When every other detection step failed then your event will be categoried as a simple `PageView`.

5.2.1 Other events

Here are events that aren't detected in the described way. They are a result of the post-processing of an event or a session.

Abandoned cart: When a session did not track a `OrderCompleted` event, the last event of that type will be converted to `AbandonedCart`.

Excluded event: There are several ways of excluding an event (e.g. by blacklisting source IP or User-Agent header matching). If an event matches given criteria it will be excluded from the reports but is still tracked and receives `ExcludedEvent` type. If you experien any report abnormalities you may check Tracker Debugger if any of the legitimate traffic is not excluded by mistake.

Broken event: The last type is assigned to the tracked event when any error occurs during the processing (e.g. you provided incorrect value in the `idgoal` parameter, provided `idsite` does not exist, etc). That way you can still check it in the tracker debugger and attached error message will tell you what is wrong with it.

5.3 Skip link tracking with a data-disable-delay attribute

5.3.1 Introduction

As per the MDN definition:

The <a> HTML element (or anchor element), with its href attribute, creates a hyperlink to web pages, files, email addresses, locations in the same page, or anything else a URL can address.

If you wish to trigger tags, when the anchor element is clicked, they need time to execute before the redirect happens. That is why our container is equipped with a delay mechanism.

5.3.2 Delay mechanism

Each app or site you create has the option to *Delay loading the next page* in its settings. You can adjust this value in *Data collection -> Other options* section in your app settings under *Administration -> Sites & apps*. The default value for each app is `500ms`. Once you assign a trigger and a tag to an anchor element on your page, this mechanism will ensure that the tag fires and has time to execute, before the visitor is redirected to the desired page.

However, not every anchor element is supposed to perform a redirect. One such example can be SPA pages where `a` elements can serve as buttons. In such case, the action performed inside the container can break the functionality of the page. That is where the `data-disable-delay` attribute comes in.

5.3.3 data-disable-delay attribute

`data-disable-dalay` is special custom attribute that is recognized by the container. Once the anchor element is clicked and the aforementioned attribute is detected on the element, it tells the container to skip the execution of the logic responsible for delaying the default action. Listeners attached to the element are executed immediatly after clicking.

Example

1. Let's assume that your Tag Manager setup includes a *Custom code (async)* tag (the contents of the tag does not matter in this case) and a basic *Click trigger* assigned to the said tag.
2. On your page, the following code is present:

```
<a
    id='link-id'
    href="/"
>
    link
</a>
<script>
    window.setTimeout(function() {
        document.addEventListener('click', function(event) {
            if(event.target.id === 'link-id') {
                event.preventDefault()
            }
        })
    }, 1000)
</script>
```

3. Once the visitor clicks the link, a redirect happens. This is not desired, since the listener performs a *preventDefault* action.
4. Now let's modify the anchor element to look like this:

```
<a
    id='link-id'
    href="/"
    data-disable-delay
>
    link
</a>
```

5. After the modification is done, clicking the link no longer performs a redirect and fires the click listener immediately.

5.4 Custom data layer name

A data layer is a data structure on your site or app where you can store data and access it with Tag Manager. The default data layer name is `dataLayer`, but you can change it to a custom one.

5.4.1 Choose a unique data layer name

Use a unique data layer name. Make sure that it's not used by other tools installed on your site or app. If the names are the same, the tools can interfere with each other.

To check if the data layer name is used on your site or app, follow these steps:

1. Pick your new data layer name. Example: `customDataLayer`.
2. In a browser's console, run the following script with the picked name:

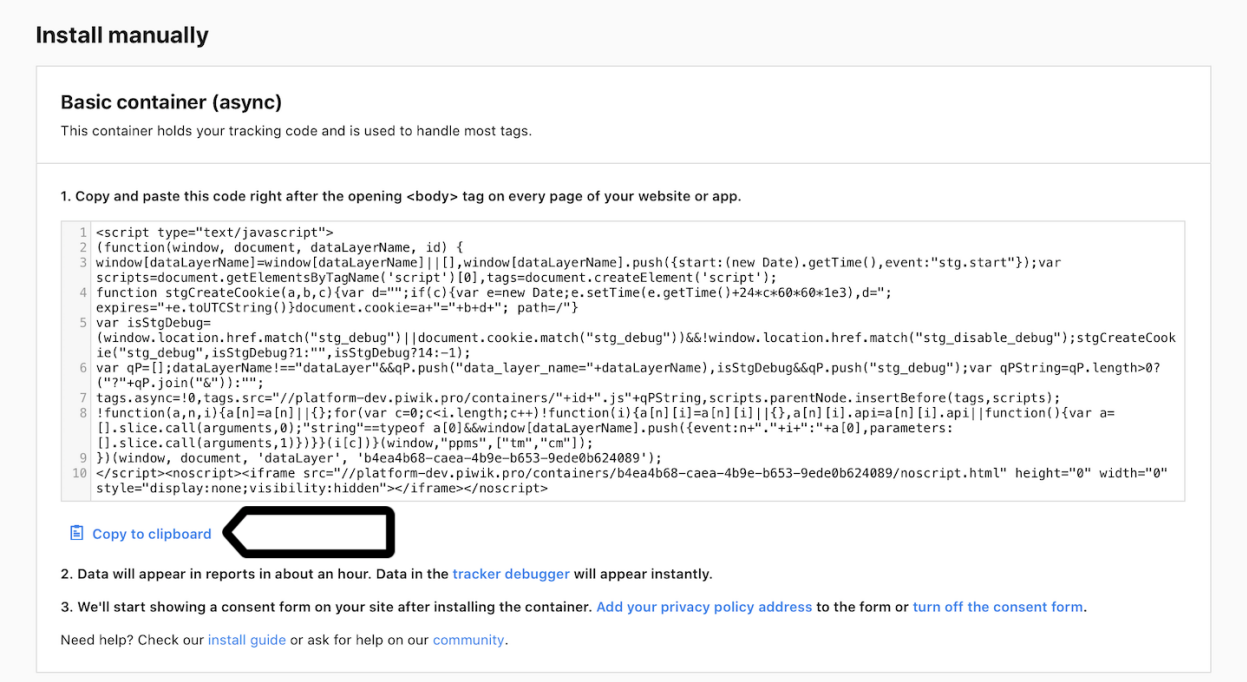
```
var dataLayerName = "customDataLayer";
!window.hasOwnProperty(dataLayerName);
```

3. If the return statement is `true`, you can use this name safely. It means that no other tool is using this name.

5.4.2 Rename your data layer

To rename the data layer, follow these steps:

1. Log in to Piwik PRO.
2. Go to **Menu > Administration**.
3. Navigate to **Sites & apps**.
4. Select the site or app for which you want to rename the data layer.
5. Navigate to **Installation**.
6. Copy the basic container's code. You'll modify this code in the next steps.

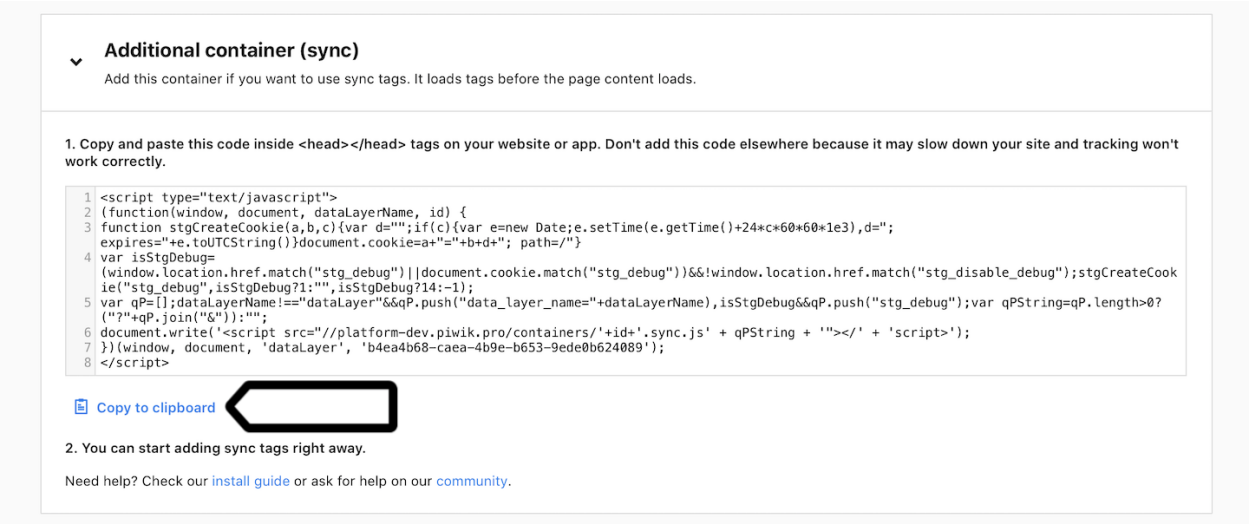


7. In the copied code, change `dataLayer` to a custom name.

```
(window, document, 'dataLayer', '69bc995f-c40a-42ae-b756-b8b9fbc16508');
```

```
<script type="text/javascript">
(function(window, document, dataLayerName, id) {
window[dataLayerName]=window[dataLayerName]||[],window[dataLayerName].push({start:(new Date).getTime(),event:"stg.start"});var
scripts=document.getElementsByTagName('script')[0],tags=document.createElement('script');
function stgCreateCookie(a,b,c){var d="";if(c){var e=new Date;e.setTime(e.getTime()+24*c*60*60*1e3),d=";
expires="+e.toUTCString()}document.cookie=a+"="+b+d+"; path=/"}
var isStgDebug=
(window.location.href.match("stg_debug")||document.cookie.match("stg_debug"))&&!window.location.href.match("stg_disable_debug");stgCreateCooki
e("stg_debug",isStgDebug?1:"",isStgDebug?14:-1);
var qP=[];dataLayerName!=="dataLayer"&&qP.push("data_layer_name="+dataLayerName),isStgDebug&&qP.push("stg_debug");var qPString=qP.length>0?
("?"+qP.join("&")):"";
tags.async=!0,tags.src="https://clearbank.containers.piwik.pro/"+id+".js"+qPString,scripts.parentNode.insertBefore(tags,scripts);
!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,"ppms",["tm","cm"]);
})(window, document, 'dataLayer', '69bc995f-c40a-42ae-b756-b8b9fbc16508');
</script><noscript><iframe src="https://clearbank.containers.piwik.pro/69bc995f-c40a-42ae-b756-b8b9fbc16508/noscript.html" height="0"
width="0" style="display:none;visibility:hidden"></iframe></noscript>
```

8. Paste the code right after the opening **<body>** tag on every page of your website or app.
9. Optionally, copy the additional container's code. You'll modify this code in the next steps.



10. In the copied code, change `dataLayer` to a custom name.

```
(window, document, 'dataLayer', '69bc995f-c40a-42ae-b756-b8b9fbc16508');
```

```
<script type="text/javascript">
(function(window, document, dataLayerName, id) {
function stgCreateCookie(a,b,c){var d="";if(c){var e=new Date;e.setTime(e.getTime()+24*c*60*60*1e3),d=";
expires="+e.toUTCString()}document.cookie=a+"="+b+d+"; path=/"}
var isStgDebug=
(window.location.href.match("stg_debug")||document.cookie.match("stg_debug"))&&!window.location.href.match("stg_disable_debug");stgCreateCooki
e("stg_debug",isStgDebug?1:"",isStgDebug?14:-1);
var qP=[];dataLayerName!=="dataLayer"&&qP.push("data_layer_name="+dataLayerName),isStgDebug&&qP.push("stg_debug");var qPString=qP.length>0?
("?"+qP.join("&")):"";
document.write('<script src="https://clearbank.containers.piwik.pro/'+id+'.sync.js' + qPString + '"></' + 'script>');
})(window, document, 'dataLayer', '69bc995f-c40a-42ae-b756-b8b9fbc16508');
</script>
```

Note: **Note:** If you're using both containers, use the same data layer name in each container. Otherwise, things can break.

11. Paste the code inside **<head></head>** tags on your website or app. Don't add this code elsewhere because it may slow down your site and tracking won't work correctly.
12. Replace all existing references to the old data layer name. For example, if you use

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

replace it with

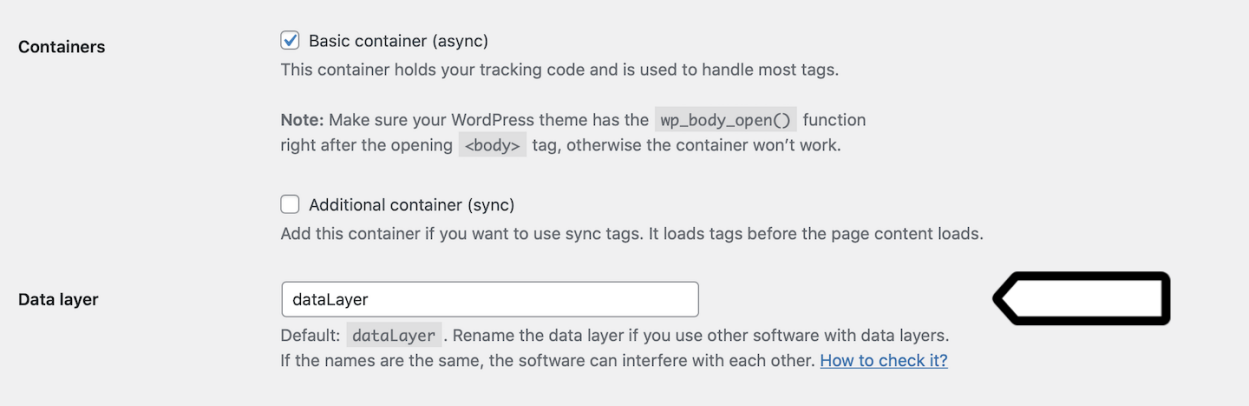
```
customDataLayer.push({event: "test-event"});
```

5.4.3 WordPress plugin: rename your data layer

If you installed our containers with the WordPress plugin, you can quickly rename the data layer in the plugin settings.

To rename the data layer in our WordPress plugin, follow these steps:

1. In your WordPress admin panel, go to **Settings > Piwik PRO**.
2. In **Data layer**, change the name to a custom one.



3. Click **Save changes**.
4. Replace all existing references to the old data layer name. For example, if you use

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

replace it with

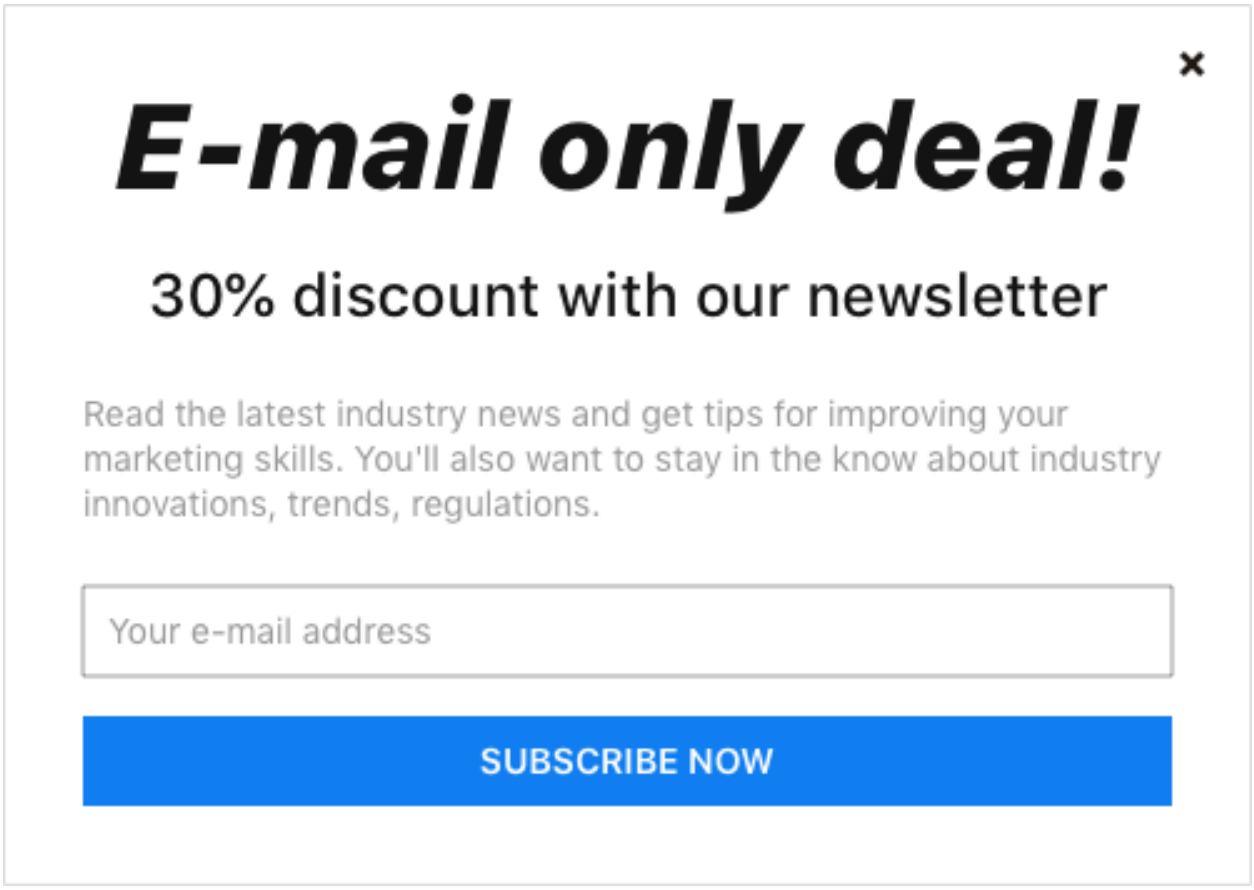
```
customDataLayer.push({event: "test-event"});
```

5.5 Custom popup examples

5.5.1 Introduction

Since version 10.1 of PPAS there is a possiblity of creating a *Custom popup* tag template. To add one, head to *Tag Manager* and while on *Tags* tab, choose + *Crate new tag*. From there you can select *Custom popup* template. Once added, you will be greated by default template code which consists of overlay, popup box and close button. To highlight what can be created with the use of this template, we decided to share some example implementations that can be further modified and expanded.

5.5.2 Example 1



Example code:

```
<div class="ppms-popup-overlay">
  <div class="ppms-popup-box">
    <span class="ppms-popup-close-button"> <!-- classname must stay as it is,␣
↪otherwise close button will not work -->
      <svg width="16px" height="16px" viewBox="0 0 16 16" version="1.1" xmlns="http://
↪www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
        <g>
            <path d="M11.125,3 L13,4.875 L9.874,7.999 L13,11.125 L11.125,13 L7.999,9.
↪874 L4.875,13 L3,11.125 L6.125,7.999 L3,4.87L4.875,3 L7.999,6.125 L11.125,3 Z" />
        </g>
      </svg>
    </span>
    <div class="ppms-popup-content">
      <h1 class="ppms-popup-header">E-mail only deal!</h1>
      <h2 class="ppms-popup-subheader">30% discount with our newsletter</h2>
      <p class="ppms-popup-paragraph">
        Read the latest industry news and get tips for improving your marketing␣
↪skills.
        You'll also want to stay in the know about industry innovations, trends,␣
↪regulations.
      </p>
      <input class="ppms-popup-input" type="email" placeholder="Your e-mail address">
      <button class="ppms-popup-button">Subscribe now</button>
```

(continues on next page)

(continued from previous page)

```
    </div>
  </div>
</div>


<style type="text/css">
  .ppms-popup-overlay {
    z-index: 10000;
    width: 100%;
    height: 100%;
    top: 0;
    left: 0;
    position: fixed;
    background-color: rgba(0, 0, 0, 0.8);
    display: flex;
    justify-content: center;
    align-items: center;
  }

  .ppms-popup-box {
    max-width: 500px;
    min-height: 350px;
    box-sizing: border-box;
    position: relative;
    background-color: #fff;
    border: 1px solid #ddd;
    padding: 28px 32px 32px 32px;
  }

  .ppms-popup-close-button {
    z-index: 1000;
    right: 16px;
    top: 16px;
    position: absolute;
    cursor: pointer;
    box-sizing: content-box;
    fill: #000;
  }

  .ppms-popup-close-button:hover {
    fill: #999;
  }

  .ppms-popup-content {
    font-family: "BlinkMacSystemFont", -apple-system, "Roboto", "Oxygen-Sans", "Ubuntu
↪", "Cantarell", "Helvetica Neue", sans-serif;
  }

  .ppms-popup-header {
    text-align: center;
    font-style: italic;
    font-size: 48px;
    line-height: 58px;
    color: #131313;
    font-weight: 700;
    margin: 0;
  }
```

(continues on next page)

(continued from previous page)

```
.ppms-popup-subheader {
  color: #131313;
  font-size: 24px;
  font-weight: 500;
  line-height: 29px;
  text-align: center;
  margin-top: 16px;
}

.ppms-popup-paragraph {
  color: #999999;
  font-size: 14px;
  line-height: 18px;
  margin-top: 24px;
}

.ppms-popup-input {
  display: block;
  width: 100%;
  box-sizing: border-box;
  height: 36px;
  border: 1px solid #999999;
  background-color: #FFFFFF;
  color: #999999;
  font-size: 14px;
  line-height: 16px;
  margin-top: 24px;
  padding: 0 10px;
}

.ppms-popup-input::placeholder {
  color: #999999;
}

.ppms-popup-button {
  height: 36px;
  background-color: #107ef1;
  color: #ffffff;
  width: 100%;
  text-transform: uppercase;
  border: none;
  font-size: 14px;
  font-weight: 600;
  line-height: 16px;
  text-align: center;
  margin-top: 16px;
  cursor: pointer;
}

.ppms-popup-button:hover {
  background-color: #338dee;
}

@media (max-height: 360px) {
  .ppms-popup-box {
    padding: 20px;
```

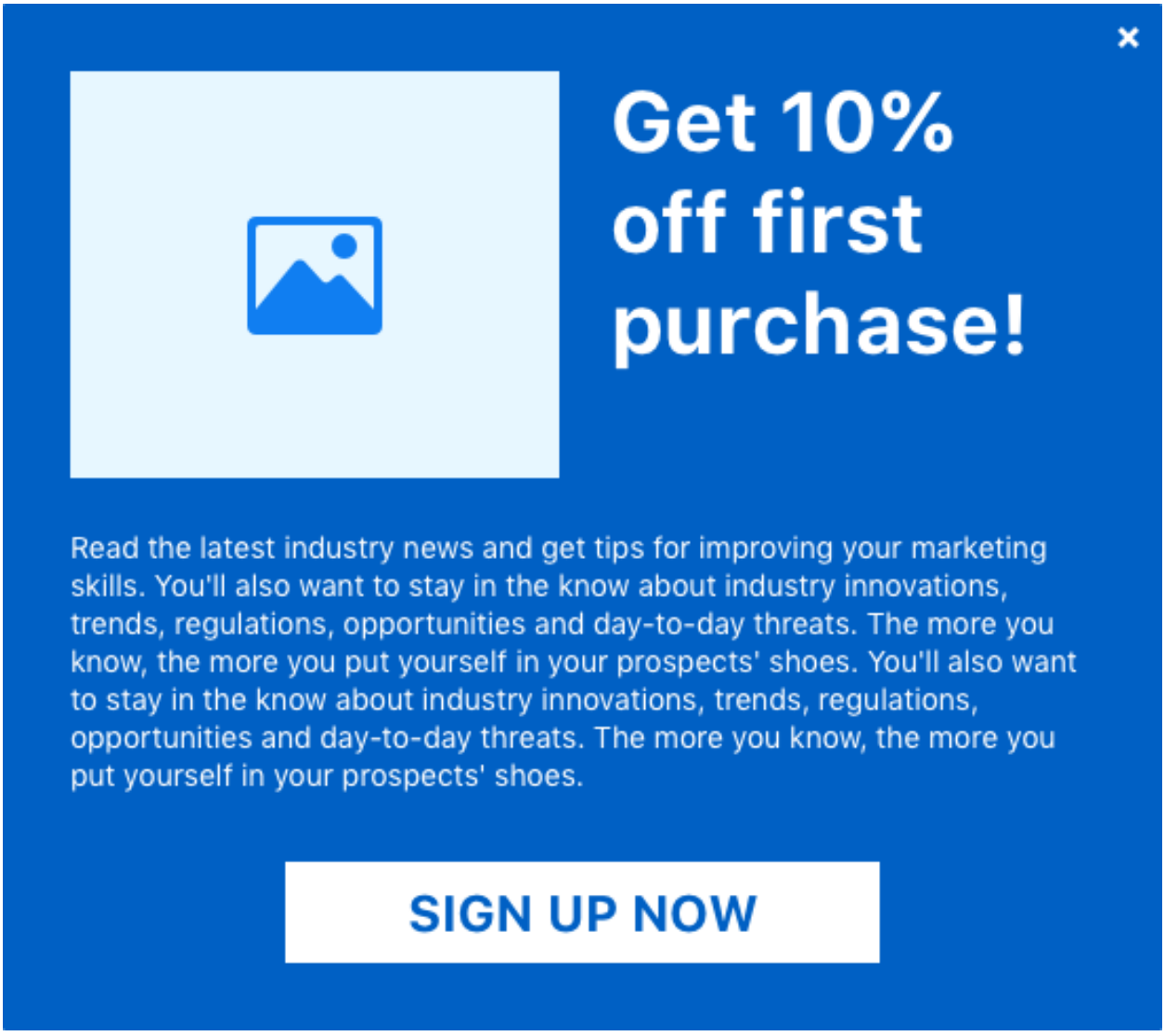
(continues on next page)

(continued from previous page)

```
      min-height: unset;
    }
  }
</style>
```

Note: Handling of the close button is provided out of the box, as long as the class name `ppms-popup-close-button` is unchanged. Your own JavaScript code to handle *Subscribe now* button needs to be provided.

5.5.3 Example 2



Example code:

```
<div class="ppms-popup-overlay">
  <div class="ppms-popup-box">
    <span class="ppms-popup-close-button"> <!-- classname must stay as it is,␣
↪otherwise close button will not work -->
      <svg width="16px" height="16px" viewBox="0 0 16 16" version="1.1" xmlns="http://
↪www.w3.org/2000/svg"
       xmlns:xlink="http://www.w3.org/1999/xlink">
        <g>
          <path d="M11.125,3 L13,4.875 L9.874,7.999 L13,11.125 L11.125,13 L7.999,9.
↪874 L4.875,13 L3,11.125 L6.125,7.999 L3,4.87L4.875,3 L7.999,6.125 L11.125,3 Z" />
        </g>
      </svg>
    </span>
    <!-- classname must stay as it is, otherwise close button will not work -->
    <div class="ppms-popup-content">
      <div class="ppms-popup-top-wrapper">
        <div class="ppms-popup-image">
          <svg width="64px" height="56px" viewBox="0 0 64 56" version="1.1" xmlns=
↪"http://www.w3.org/2000/svg"
           xmlns:xlink="http://www.w3.org/1999/xlink">
            <g transform="translate(-869.000000, -538.000000)">
              <g transform="translate(48.000000, 538.000000)">
                <path d="M871.25,18.25 C870.083328,19.416672 868.666672,20 867,20␣
↪C865.333328,20 863.916672,19.416672
                 862.75,18.25 C861.583328,17.083328 861,15.666672 861,14 C861,12.
↪333328 861.583328,10.916672 862.75,9.75
                 C863.916672,8.583328 865.333328,8 867,8 C868.666672,8 870.083328,8.
↪583328 871.25,9.75 C872.416672,
                 10.916672 873,12.333328 873,14 C873,15.666672 872.416672,17.083328␣
↪871.25,18.25 Z M881,0 C882.142866,0
                 883.095232,0.388882667 883.857143,1.16666667 C884.619054,1.94445067␣
↪885,2.91665733 885,4.08333333 L885,
                 51.9166667 C885,53.0833389 884.619054,54.0555521 883.857143,54.
↪8333333 C883.095232,55.611115 882.142866,
                 56 881,56 L825,56 C823.857137,56 822.904765,55.611115 822.142857,54.
↪8333333 C821.380949,54.0555521 821,
                 53.0833389 821,51.9166667 L821,4.08333333 C821,2.91665733 821.
↪380949,1.94445067 822.142857,1.16666667
                 C822.904765,0.388882667 823.857137,0 825,0 L881,0 Z M866.5625,28.
↪4117647 L881,44 L881,5.76470588 C881,
                 4.58822588 880.368059,4 879.104167,4 L826.895833,4 C825.826384,4␣
↪825.194445,4.58822588 825,5.76470588
                 L825,44 L843.375,21.6470588 C844.152784,20.8627388 844.979167,20.
↪4705882 845.854167,20.4705882
                 C846.923617,20.4705882 847.75,20.8137224 848.333333,21.5 L856.
↪208333,30.1764706 L856.791667,30.7647059
                 C857.375,31.1568659 857.909716,31.3529412 858.395833,31.3529412␣
↪C858.881951,31.3529412 859.465275,
                 31.1078494 860.145833,30.6176471 L862.770833,28.2647059 C863.451392,
↪27.7745035 864.083333,27.5294118
                 864.666667,27.5294118 C865.444451,27.5294118 866.076383,27.8235294␣
↪866.5625,28.4117647 Z" />
              </g>
            </g>
          </svg>
        </div>
        <h1 class="ppms-popup-header">Get 10% off first purchase!</h1>
```

(continues on next page)

(continued from previous page)

```
      </div>
      <p class="ppms-popup-paragraph">
        Read the latest industry news and get tips for improving your marketing␣
↪skills.
        You'll also want to stay in the know about industry innovations, trends,␣
↪regulations, opportunities and
        day-to-day threats. The more you know, the more you put yourself in your␣
↪prospects' shoes. You'll also want to
        stay in the know about industry innovations, trends, regulations,␣
↪opportunities and day-to-day threats. The more
        you know, the more you put yourself in your prospects' shoes.
      </p>
      <button class="ppms-popup-button">Sign up now</button>
    </div>
  </div>
</div>


<style type="text/css">
  .ppms-popup-overlay {
    z-index: 10000;
    width: 100%;
    height: 100%;
    top: 0;
    left: 0;
    position: fixed;
    background-color: rgba(0, 0, 0, 0.8);
    display: flex;
    justify-content: center;
    align-items: center;
  }

  .ppms-popup-box {
    width: 550px;
    min-height: 487px;
    box-sizing: border-box;
    position: relative;
    background-color: #0060c4;
    padding: 32px;
  }

  .ppms-popup-close-button {
    z-index: 1000;
    right: 8px;
    top: 8px;
    position: absolute;
    cursor: pointer;
    box-sizing: content-box;
    fill: #fff;
  }

  .ppms-popup-close-button:hover {
    fill: #aaa;
  }

  .ppms-popup-content {
    font-family: "BlinkMacSystemFont", -apple-system, "Roboto", "Oxygen-Sans", "Ubuntu
↪", "Cantarell", "Helvetica Neue", sans-serif;
```

(continues on next page)

(continued from previous page)

```
}

.ppms-popup-top-wrapper {
  display: flex;
  flex-wrap: wrap;
  align-items: top;
  margin: -12px;
}

.ppms-popup-image {
  flex: 1 1 232px;
  height: 193px;
  background-color: #e6f7ff;
  display: flex;
  justify-content: center;
  align-items: center;
  margin: 12px;
  fill: #107EF1;
}

.ppms-popup-header {
  flex: 1 1 230px;
  text-align: left;
  color: #fff;
  font-size: 40px;
  font-weight: bold;
  line-height: 48px;
  margin: 12px;
}

.ppms-popup-paragraph {
  color: #fff;
  font-size: 14px;
  line-height: 18px;
  margin-top: 24px;
}

.ppms-popup-button {
  display: block;
  width: 282px;
  height: 48px;
  background-color: #fff;
  color: #0060C4;
  font-size: 24px;
  font-weight: bold;
  line-height: 29px;
  text-align: center;
  text-transform: uppercase;
  border: none;
  margin: 32px auto 0 auto;
  cursor: pointer;
}

.ppms-popup-button:hover {
  background-color: #aaa;
}
```

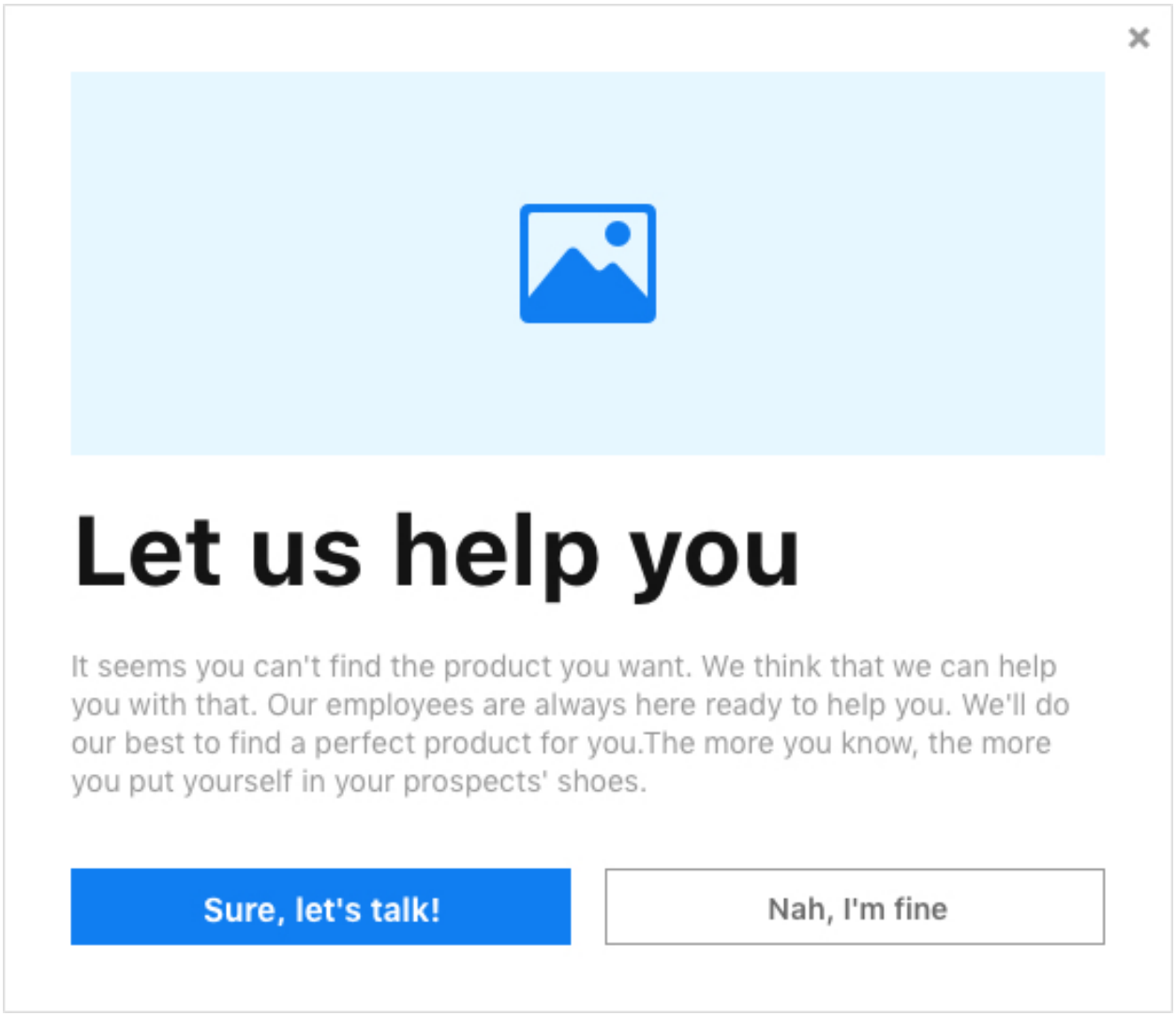
(continues on next page)

(continued from previous page)

```
  @media (max-width: 560px) {
    .ppms-popup-image {
      display: none;
    }

    .ppms-popup-box {
      display: flex;
      align-items: center;
    }

    .ppms-popup-button {
      padding: 0 25px;
      width: auto;
    }
  }

  @media (max-height: 490px) {
    .ppms-popup-image {
      display: none;
    }

    .ppms-popup-box {
      width: 100%;
      display: flex;
      align-items: center;
      min-height: unset;
      padding: 20px;
    }
  }
</style>
```

Note: Handling of the close button is provided out of the box, as long as the class name `ppms-popup-close-button` is unchanged. Your own JavaScript code to handle *Sign up now* button needs to be provided.

5.5.4 Example 3



Example code:

```
<div class="ppms-popup-overlay">
  <div class="ppms-popup-box">
    <span class="ppms-popup-close-button"> <!-- classname must stay as it is,␣
↪otherwise close button will not work -->
      <svg width="16px" height="16px" viewBox="0 0 16 16" version="1.1" xmlns="http://
↪www.w3.org/2000/svg"
        xmlns:xlink="http://www.w3.org/1999/xlink">
        <g>
          <path d="M11.125,3 L13,4.875 L9.874,7.999 L13,11.125 L11.125,13 L7.999,9.
↪874 L4.875,13 L3,11.125 L6.125,7.999 L3,4.87L4.875,3 L7.999,6.125 L11.125,3 Z" />
        </g>
      </svg>
    </span>
    <div class="ppms-popup-content">
      <div class="ppms-popup-image">
        <svg width="64px" height="56px" viewBox="0 0 64 56" version="1.1" xmlns=
↪"http://www.w3.org/2000/svg"
```

(continues on next page)

(continued from previous page)

```
          xmlns:xlink="http://www.w3.org/1999/xlink">
          <g transform="translate(-869.000000, -538.000000)">
            <g transform="translate(48.000000, 538.000000)">
              <path d="M871.25,18.25 C870.083328,19.416672 868.666672,20 867,20 C865.
↪333328,20 863.916672,19.416672
                862.75,18.25 C861.583328,17.083328 861,15.666672 861,14 C861,12.
↪333328 861.583328,10.916672 862.75,9.75
                C863.916672,8.583328 865.333328,8 867,8 C868.666672,8 870.083328,8.
↪583328 871.25,9.75 C872.416672,
                10.916672 873,12.333328 873,14 C873,15.666672 872.416672,17.083328␣
↪871.25,18.25 Z M881,0 C882.142866,0
                883.095232,0.388882667 883.857143,1.16666667 C884.619054,1.94445067␣
↪885,2.91665733 885,4.08333333 L885,
                51.9166667 C885,53.0833389 884.619054,54.0555521 883.857143,54.
↪8333333 C883.095232,55.611115 882.142866,
                56 881,56 L825,56 C823.857137,56 822.904765,55.611115 822.142857,54.
↪8333333 C821.380949,54.0555521 821,
                53.0833389 821,51.9166667 L821,4.08333333 C821,2.91665733 821.380949,
↪1.94445067 822.142857,1.16666667
                C822.904765,0.388882667 823.857137,0 825,0 L881,0 Z M866.5625,28.
↪4117647 L881,44 L881,5.76470588 C881,
                4.58822588 880.368059,4 879.104167,4 L826.895833,4 C825.826384,4 825.
↪194445,4.58822588 825,5.76470588
                L825,44 L843.375,21.6470588 C844.152784,20.8627388 844.979167,20.
↪4705882 845.854167,20.4705882
                C846.923617,20.4705882 847.75,20.8137224 848.333333,21.5 L856.208333,
↪30.1764706 L856.791667,30.7647059
                C857.375,31.1568659 857.909716,31.3529412 858.395833,31.3529412 C858.
↪881951,31.3529412 859.465275,
                31.1078494 860.145833,30.6176471 L862.770833,28.2647059 C863.451392,
↪27.7745035 864.083333,27.5294118
                864.666667,27.5294118 C865.444451,27.5294118 866.076383,27.8235294␣
↪866.5625,28.4117647 Z" />
            </g>
          </g>
        </svg>
      </div>
      <h1 class="ppms-popup-header">Let us help you</h1>
      <p class="ppms-popup-paragraph">
        It seems you can't find the product you want. We think that we can help you␣
↪with that. Our employees are always
        here ready to help you. We'll do our best to find a perfect product for you.
↪The more you know, the more you put
        yourself in your prospects' shoes.
      </p>
      <div class="ppms-popup-button-wrapper">
        <button class="ppms-popup-button ppms-popup-button-accept">Sure, let's talk!</
↪button>
        <button class="ppms-popup-button ppms-popup-button-reject">Nah, I'm fine</
↪button>
      </div>
    </div>
  </div>
</div>


<style type="text/css">
```

(continues on next page)

(continued from previous page)

```
  .ppms-popup-overlay {
    z-index: 10000;
    width: 100%;
    height: 100%;
    top: 0;
    left: 0;
    position: fixed;
    background-color: rgba(0, 0, 0, 0.8);
    display: flex;
    justify-content: center;
    align-items: center;
  }

  .ppms-popup-box {
    width: 550px;
    min-height: 474px;
    box-sizing: border-box;
    position: relative;
    background-color: #fff;
    padding: 32px;
  }

  .ppms-popup-close-button {
    z-index: 1000;
    right: 8px;
    top: 8px;
    position: absolute;
    cursor: pointer;
    box-sizing: content-box;
    fill: #999;
  }

  .ppms-popup-close-button:hover {
    fill: #666;
  }

  .ppms-popup-content {
    font-family: "BlinkMacSystemFont", -apple-system, "Roboto", "Oxygen-Sans", "Ubuntu
↪", "Cantarell", "Helvetica Neue", sans-serif;
  }

  .ppms-popup-image {
    width: 100%;
    height: 180px;
    background-color: #e6f7ff;
    display: flex;
    justify-content: center;
    align-items: center;
    fill: #107EF1;
  }

  .ppms-popup-header {
    text-align: left;
    color: #000;
    font-size: 46px;
    font-weight: bold;
    margin: 16px 0;
```

(continues on next page)

(continued from previous page)

```
}

.ppms-popup-paragraph {
  color: #999;
  font-size: 14px;
  line-height: 18px;
  margin-bottom: 32px;
}

.ppms-popup-button-wrapper {
  display: flex;
  flex-wrap: wrap;
  margin: -8px;
}

.ppms-popup-button {
  height: 36px;
  flex: 1 1 235px;
  font-size: 15px;
  font-weight: bold;
  line-height: 18px;
  text-align: center;
  padding: 0px;
  margin: 8px;
  cursor: pointer;
}

.ppms-popup-button-accept {
  background-color: #1c80eb;
  color: #fff;
  border: none;
}

.ppms-popup-button-accept:hover {
  background-color: #338dee;
}

.ppms-popup-button-reject {
  background-color: #fff;
  color: #666;
  border: 1px solid #999;
}

.ppms-popup-button-reject:hover {
  background-color: #eee;
}

@media (max-width: 560px), (max-height: 480px) {
  .ppms-popup-image {
    display: none;
  }

  .ppms-popup-box {
    display: flex;
    align-items: center;
    min-height: unset;
  }
```

(continues on next page)

(continued from previous page)

```
  }
</style>
```

Note: Handling of the close button is provided out of the box, as long as the class name `ppms-popup-close-button` is unchanged. Your own JavaScript code to handle *Sure, let's talk* and *Nah, I'm fine* buttons needs to be provided.

CHAPTER 6

Changelog

We update our product bi-weekly, except for on-premises accounts that are updated once every year. To see recent changes, see our changelog.

Index

Symbols

C