Piwik PRO Marketing Suite Documentation

Release 15.3

Piwik PRO

Nov 02, 2021

Contents

PPAS version: 15.3 (see changelog)

Analytics (new)

1.1 Columns

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

Note: Each column listed in this document defines a *Scope* attribute. If you request a query that includes at least one column which requires *event* scope, the entire query will be calculated using events, instead of sessions. This might distort some custom metrics such as averages of a *session* dimension (e.g. average session time).

1.1.1 Metrics

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

Table 1: Base Metrics

Metric Name	Column ID	Scope	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

Table 1 – continued from previous page

Metric Name	Column ID	Scope	Type
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
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

1.1.2 Dimensions

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

Note: "Database type" column presents the type of source column of the dimension (in case of enum - type of the ID, in case of dynamic dimensions - not applicable).

Table 2: Base Dimensions

Dimension Name	Column ID	Scope	Type	Database Type	Nullab
Visitor ID	visitor_id	session	hex	uint64	False
User ID	user_id	session	str	string	False
Cookie ID	cookie_id	session	hex	uint64	False
Returning visitor	visitor_returning	session	[int, str]	uint8	False
Session number	visitor_session_number	session	int	uint16	False
Days since last session	visitor_days_since_last_session	session	int	uint16	True
Days since first session	visitor_days_since_first_session	session	int	uint16	True
Days since order	visitor_days_since_order	session	int	uint16	True
Events in session	session_total_events	session	int	uint16	False
Session time	session_total_time	session	int	uint32	False
Page views in session	session_total_page_views	session	int	uint16	False
Outlinks in session	session_total_outlinks	session	int	uint16	False
Downloads in session	session_total_downloads	session	int	uint16	False
Site searches in session	session_total_site_searches	session	int	uint16	False
Custom events in session	session_total_custom_events	session	int	uint16	False
Content impressions in session	session_total_content_impressions	session	int	uint16	False
Content interactions in session	session_total_content_interactions	session	int	uint16	False
Goal conversions in session	session_total_goal_conversions	session	int	uint16	False
Ecommerce conversions in session	session_total_ecommerce_conversions	session	int	uint16	False
Abandoned carts in session	session_total_abandoned_carts	session	int	uint16	False
Unique page views in session	session_unique_page_views	session	int	uint16	False
Unique outlinks in session	session_unique_outlinks	session	int	uint16	False
Unique downloads in session	session_unique_downloads	session	int	uint16	False
Unique site searches in session	session_unique_searches	session	int	uint16	False
Unique custom events in session	session_unique_custom_events	session	int	uint16	False
Unique content impressions in session	session_unique_content_impressions	session	int	uint16	False
Unique content interactions in session	session_unique_content_interactions	session	int	uint16	False
Goals converted in session	session_goals	session	array	array of int32	False
Shopping stage	session_ecommerce_status	session	[int, str]	uint8	False
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
Channel	referrer_type	session	[int, str]	uint8	False
Referrer URL	referrer_url	session	str	string	False
Campaign name	campaign_name	session	str	string	False
Campaign ID	campaign_id	session	str	string	False
Campaign content	campaign_content	session	str	string	False
Google Click ID	campaign_gclid	session	str	string	True
Operating system	operating_system	session	[str, str]	string(3)	True
Operating system version	operating_system_version	session	str	string	False
Browser engine	browser_engine	session	str	string	False
Browser name	browser_name	session	[str, str]	string(2)	True
Browser version	browser_version	session	str	string	False
Browser language	browser_language_iso639	session	[str, str]	string(2)	True
Browser fingerprint	browser_fingerprint	session	int	uint64	False
Device type	device_type	session	[int, str]	uint8	True
Device brand	device_brand	session	[str, str]	string(2)	True
Device model	device_model	session	str	string	False

Table 2 – continued from previous page

Dimension Name	Column ID	Scope	Type	Database Type	Nullab
Resolution	resolution	session	str	string	True
Resolution width	resolution_width	session	int	uint16	True
Resolution height	resolution_height	session	int	uint16	True
PDF plugin	plugin_pdf	session	int(0,1)	uint8	False
Flash plugin	plugin_flash	session	int(0,1)	uint8	False
Java plugin	plugin_java	session	int(0,1)	uint8	False
Director plugin	plugin_director	session	int(0,1)	uint8	False
QuickTime plugin	plugin_quicktime	session	int(0,1)	uint8	False
RealPlayer plugin	plugin_realplayer	session	int(0,1)	uint8	False
Windows Media Player plugin	plugin_windowsmedia	session	int(0,1)	uint8	False
Gears plugin	plugin_gears	session	int(0,1)	uint8	False
Silverlight plugin	plugin_silverlight	session	int(0,1)	uint8	False
Cookie support	plugin_cookie	session	int(0,1)	uint8	False
Continent	location_continent_iso_code	session	[str, str]	string(2)	True
Country	location_country_name	session	[str, str]	string	True
Subdivision	location_subdivision_1_name	session	[str, str]	string	True
Subdivision 2	location_subdivision_2_name	session	[str, str]	string	True
City	location_city_name	session	[int, str]	string	True
Designated market area	location_metro_code	session	[str, str]	string(3)	True
Latitude	location_latitude	session	float	float64	True
Longitude	location_longitude	session	float	float64	True
Provider	location_provider	session	str	string	False
Organization	location_organization	session	str	string	False
Session exit URL	session_exit_url	session	str	string	False
Session exit title	session_exit_title	session	str	string	False
Session entry URL	session_entry_url	session	str	string	False
Session entry title	session_entry_title	session	str	string	False
Session second URL	session_second_url	session	str	string	False
Session second title	session_second_title	session	str	string	False
Session bounce	is_bounce	session	int(0,1)	uint8	False
Event ID	event_id	event	int	uint64	False
Session ID	session_id	session	int	uint64	False
Exit view	is_exit	event	int(0,1)	uint8	False
Entry view	is_entry	event	int(0,1)	uint8	False
Event type	event_type	event	[int, str]	uint8	False
Page URL	event_url	event	str	string	False
Page title	event_title	event	str	string	False
Outlink URL	outlink_url	event	str	string	False
Download URL	download_url	event	str	string	False
Search keyword	search_keyword	event	str	string	False
Search category	search_category	event	str	string	False
Search results count	search_results_count	event	int	uint16	True
Custom event category	custom_event_category	event	str	string	False
Custom event action	custom_event_action	event	str	string	False
Custom event name	custom_event_name	event	str	string	False
Custom event value	custom_event_value	event	float	float64	True
Content name	content_name	event	str	string	False
Content piece	content_piece	event	str	string	False
Content target	content_target	event	str	string	False

Table 2 – continued from previous page

Dimension Name	Column ID	Scope	Type	Database Type	Nullab
Previous page view URL	previous_event_url	event	str	string	False
Previous page view title	previous_event_title	event	str	string	False
Next page view URL	next_event_url	event	str	string	False
Next page view title	next_event_title	event	str	string	False
Event index	event_index	event	int	uint16	False
Page view index	page_view_index	event	int	uint16	True
Time on page	time_on_page	event	int	uint32	True
Page generation time	page_generation_time	event	float	float64	True
Goal name	goal_id	event	[int, str]	int32	True
Goal revenue	goal_revenue	event	float	float64	True
Lost revenue	lost_revenue	event	float	float64	True
Order ID	order_id	event	str	string	False
Item count	item_count	event	int	uint16	True
Revenue	revenue	event	float	float64	True
Revenue (Subtotal)	revenue_subtotal	event	float	float64	True
Revenue (Tax)	revenue_tax	event	float	float64	True
Revenue (Shipping)	revenue_shipping	event	float	float64	True
Revenue (Discount)	revenue_discount	event	float	float64	True
Time until DOM is ready	timing_dom_interactive	event	int	uint32	True
Time to interact	timing_event_end	event	int	uint32	True
Consent form view source	consent_source	event	[int, str]	uint8	True
Consent form interaction type	consent_form_button	event	[int, str]	uint8	True
Consent scope	consent_scope	event	[int, str]	uint8	True
Consent action	consent_action	event	[int, str]	uint8	True
Analytics consent	consent_type_analytics	event	int(0,1)	uint8	True
AB testing personalization consent	consent_type_ab_testing_personalization	event	int(0,1)	uint8	True
Conversion tracking consent	consent_type_conversion_tracking	event	int(0,1)	uint8	True
Marketing automation consent	consent_type_marketing_automation	event	int(0,1)	uint8	True
Remarketing consent	consent_type_remarketing	event	int(0,1)	uint8	True
User feedback consent	consent_type_user_feedback	event	int(0,1)	uint8	True
Custom consent 1	consent_type_custom_1	event	int(0,1)	uint8	True
Event custom dimension 1	event_custom_dimension_1	event	str	string	False
Event custom dimension 2	event_custom_dimension_2	event	str	string	False
Event custom dimension 3	event_custom_dimension_3	event	str	string	False
Event custom dimension 4	event_custom_dimension_4	event	str	string	False
Event custom dimension 5	event_custom_dimension_5	event	str	string	False
Event custom variable key 1	event_custom_variable_key_1	event	str	string	False
Event custom variable value 1	event_custom_variable_value_1	event	str	string	False
Event custom variable key 2	event_custom_variable_key_2	event	str	string	False
Event custom variable value 2	event_custom_variable_value_2	event	str	string	False
Event custom variable key 3	event_custom_variable_key_3	event	str	string	False
Event custom variable value 3	event_custom_variable_value_3	event	str	string	False
Event custom variable key 4	event_custom_variable_key_4	event	str	string	False
Event custom variable value 4	event_custom_variable_value_4	event	str	string	False
Event custom variable key 5	event_custom_variable_key_5	event	str	string	False
Event custom variable value 5	event_custom_variable_value_5	event	str	string	False
Session custom dimension 1	session_custom_dimension_1	session	str	string	False
Session custom dimension 2	session_custom_dimension_2	session	str	string	False
Session custom dimension 3	session_custom_dimension_3	session	str	string	False

Table 2 – continued from previous page

Dimension Name	Column ID	Scope	Type	Database Type	Nullab
Session custom dimension 4	session_custom_dimension_4	session	str	string	False
Session custom dimension 5	session_custom_dimension_5	session	str	string	False
Session custom variable key 1	session_custom_variable_key_1	session	str	string	False
Session custom variable value 1	session_custom_variable_value_1	session	str	string	False
Session custom variable key 2	session_custom_variable_key_2	session	str	string	False
Session custom variable value 2	session_custom_variable_value_2	session	str	string	False
Session custom variable key 3	session_custom_variable_key_3	session	str	string	False
Session custom variable value 3	session_custom_variable_value_3	session	str	string	False
Session custom variable key 4	session_custom_variable_key_4	session	str	string	False
Session custom variable value 4	session_custom_variable_value_4	session	str	string	False
Session custom variable key 5	session_custom_variable_key_5	session	str	string	False
Session custom variable value 5	session_custom_variable_value_5	session	str	string	False
Timestamp	timestamp	session	date	not applicable	False
Local hour	local_hour	session	int	not applicable	False
Time of redirections	redirections_time	event	int	not applicable	True
Domain Lookup Time	domain_lookup_time	event	int	not applicable	True
Server Connection Time	server_connection_time	event	int	not applicable	True
Server Response Time	server_response_time	event	int	not applicable	True
Page Rendering Time	page_rendering_time	event	int	not applicable	True
IPv4 address	ipv4_address	session	ipv4	not applicable	True
IPv6 address	ipv6_address	session	ipv6	not applicable	True
Website Name	website_name	session	[str, str]	not applicable	False

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

1.1.3 Transformations

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

Table 3: Dimension To Metric Transformations

Transformation Name	Transformation ID	Source Types	Result Type
Unique Count	unique_count	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)

Table 4: Dimension To Dimension Transformations

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

1.2 Integrations

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

1.2.1 Google Ads

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

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

Metrics

The table below lists metrics provided by Google Ads integration.

Table 5: Google Ads Metrics

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

The table below lists dimensions provided by Google Ads integration.

Note: "Database type" column presents the type of source column of the dimension (in case of enum - type of the ID, in case of dynamic dimensions - not applicable).

Table 6: Google Ads Dimensions

Dimension Name	Column ID	Scope	Type	Database Type	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

Mixed Queries

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

Note: "Database type" column presents the type of source column of the dimension (in case of enum - type of the ID, in case of dynamic dimensions - not applicable).

Table 7: Common Dimensions

Dimension Name	Column ID	Scope	Type	Database Type	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`

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

1.2.2 Google Search Console

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

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

Metrics

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

Table 8: Google Search Console Metrics

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

Dimensions

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

Note: "Database type" column presents the type of source column of the dimension (in case of enum - type of the ID, in case of dynamic dimensions - not applicable).

Table 9: Google Search Console Dimensions

Dimension Name	Column ID	Scope	Type	Database Type	Nul-lable	Notes
Source	source	ses-sion	str	string	False	
Medium	medium	ses-sion	str	string	False	
Source/Medium	source_medium	ses-sion	str	string	False	
Channel	referrer_type	ses-sion	[int, str]	uint8	False	`referrer_type.json`
Referrer URL	referrer_url	ses-sion	str	string	False	
Device type	device_type	ses-sion	[int, str]	uint8	True	`device_type.json`
Continent	loca-tion_continent_iso_code	ses-sion	[str, str]	string(2)	True	`location_continent_iso_code.json`
Country	loca-tion_country_name	ses-sion	[str, str]	string	True	ISO 3166-2 codes (e.g. "PL")
Session entry URL	session_entry_url	ses-sion	str	string	False	
Timestamp	timestamp	ses-sion	date	not appli-cable	False	by default in Raw data API
Search engine keyword	search_engine_keyword	ex-ter-nal	str	string	False	not available in Raw data API
Website Name	website_name	ses-sion	[str, str]	not appli-cable	False	website UUID

Mixed Queries

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

queries.

Note: "Database type" column presents the type of source column of the dimension (in case of enum - type of the ID, in case of dynamic dimensions - not applicable).

Table 10: Common Dimensions

Dimension Name	Column ID	Scope	Type	Database Type	Nullable	Notes
Source	source	session	str	string	False	
Medium	medium	session	str	string	False	
Source/Medium	source_medium	session	str	string	False	
Channel	referrer_type	session	[int, str]	uint8	False	`referrer_type.json`
Referrer URL	referrer_url	session	str	string	False	
Device type	device_type	session	[int, str]	uint8	True	`device_type.json`
Continent	location_continent_iso_code	session	[str, str]	string(2)	True	`location_continent_iso_code.json`
Country	location_country_name	session	[str, str]	string	True	ISO 3166-2 codes (e.g. "PL")
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

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

1.2.3 SharePoint

Once SharePoint integration is enabled, additional metrics and dimensions will become available in the *HTTP API*.

Metrics

The table below lists metrics available with SharePoint integration.

Table 11: SharePoint Metrics

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

The table below lists dimensions available with SharePoint integration.

Note: "Database type" column presents the type of source column of the dimension (in case of enum - type of the ID, in case of dynamic dimensions - not applicable).

Table 12: SharePoint Dimensions

Dimension Name	Column ID	Scope	Type	Database Type	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	

1.3 HTTP API

1.4 Metrics Mapping

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

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

1.4.1 Simple Metrics

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

Continued on next page

Table 13 – continued from previous page

Metric name	Legacy API	New API
Content interaction rate	interaction_rate	`{"column_id": "content_interaction_rate"}`
Goal conversions	nb_conversions	`{"column_id": "goal_conversions"}`
	Warning: ecommerce conversion was reported as goal conversion for goal_id 0	
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	`{"column_id": "exit_rate"}`
	Warning: definition switches depending on report	
Exit rate (events)		`{"column_id": "exit_rate_events"}`
Bounces	bounce_count	`{"column_id": "bounces"}`
Bounce rate	bounce_rate	`{"column_id": "bounce_rate"}`
	Warning: definition switches depending on report	
Bounce rate (events)		`{"column_id": "bounce_rate_events"}`
% of returning visitors	-	`{"column_id": "returning_visitors_rate"}`
Visitor IPs	-	`{"column_id": "visitor_ips"}`

Continued on next page

Table 13 – continued from previous page

Metric name	Legacy API	New API
Events per session **Warning:** does not include all event types	nb_actions_per_visit	`{"column_id": "events_per_session"}`
Unique purchases	-	`{"column_id": "unique_purchases"}`

1.4.2 Calculated Metrics

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

Metric name	Legacy API	New API
Sum of goal revenue	revenue ecommerce revenue was reported as goal revenue for goal_id 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"}`

1.4.3 Not available

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

Analytics

2.1 Reporting API (deprecated)

This API gives access to your analytics reports.

Deprecated since version 9.0: On September 20, 2020, this API will be disabled on Piwik PRO Cloud. The *tracking API* will remain as is.

We recommend you to switch to *Analytics (new) reporting API* as soon as possible. For more information, see the API migration guide.

Piwik PRO On-premises: The Piwik PRO 9.0 LTS and Piwik PRO 12.0 STS version is the last one that will use this API. You'll need to switch to the new API after the upgrade to the next version.

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

1. Login into your PPAS instance

2. Go to *Menu > Analytics settings*.
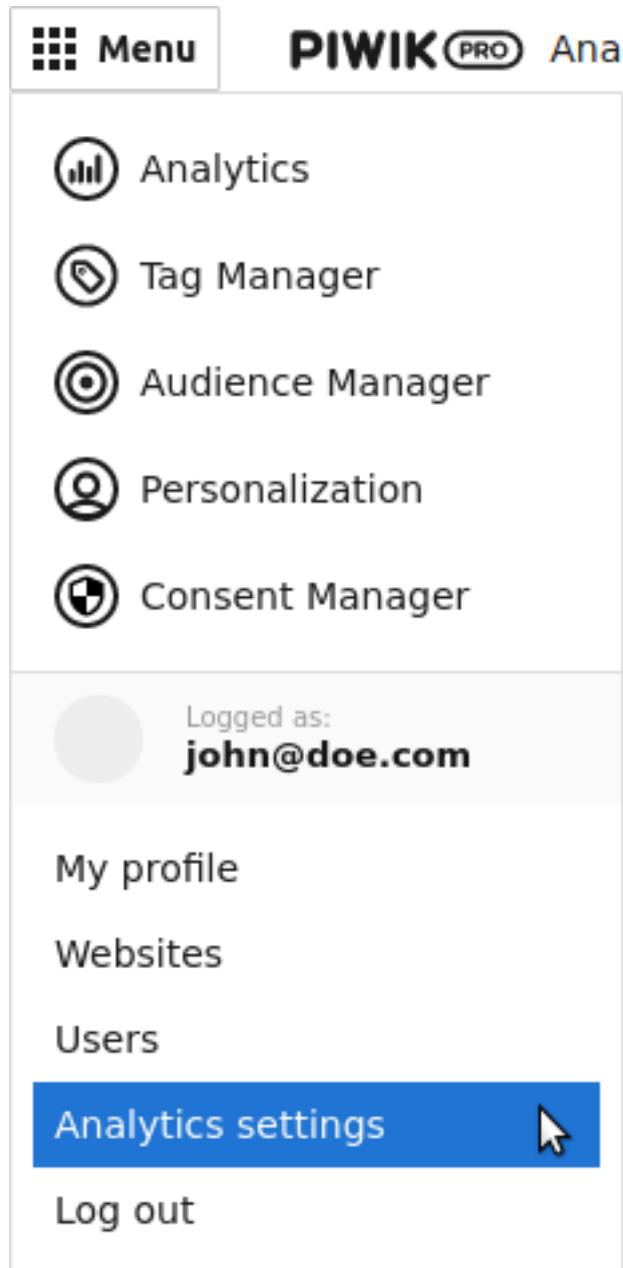
3. Select in the left menu *Platform > API*.
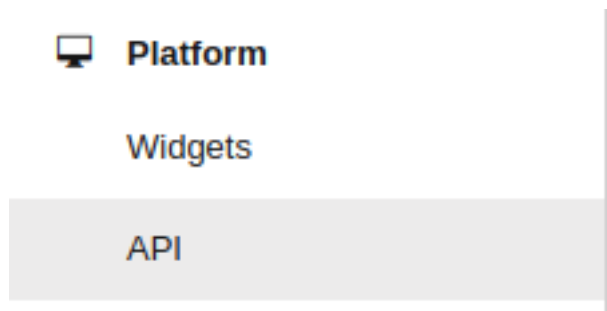
Tracker

3.1 Global reserved names used by JavaScript tracker API

The following global names are used by PPAS JavaScript tracker. Websites that will use this tracker should avoid using variables with identical 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`

3.2 JavaScript tracking API

The following API allows the user to:

- track page views

- track visits on multiple domains and subdomains

- track e-commerce events (successful orders, cart changes, product and category views)

- track content impressions

- manage custom variables to use them later

- track clicked links to external domains and download files

3.2.1 Installing Tracking code

There are two ways of installing a tracking code:

Installing tracking code via Tag Manager

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

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

Installing tracking code via code snippet.

Installing tracking code via code snippet

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

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

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

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

- String `XXX-XXX-XXX-XXX-XXX` should be replaced with *app ID* (e.g. `efcd98a5-335b-48b0-ab17-bf43f1c542be`).

- String `https://your-instance-name.piwik.pro/` should be replaced with your PPAS instance address.

```
<!-- Piwik -->
<script type="text/javascript">
  var _paq = _paq || [];
  _paq.push(["trackPageView"]);
```

(continues on next page)

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

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

This code initializes the Analytics tracker in following ways:

1. Initializes the global `_paq.push` command queue that schedules commands to be run when the Analytics tracker library is loaded.

2. Schedules basic configuration of Analytics tracker using `_paq.push`.

3. Creates a `<script>` tag that asynchronously loads the Analytics tracker library.

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

Alternative multi-domain configurations

Tracking domains and all subdomains

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

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

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

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

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

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

Tracking multiple domains as one site

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

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

domains
> **Required** `array` Domains array, with wildcards

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

Tracking subdirectories of domain as separate websites

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

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

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

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

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

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

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

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

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

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

Navigation timing page performance metrics

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

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

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

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

3.2.2 Command queue

Loading snippet creates the following API function:

`_paq.`**`push`**`(command)`
> JavaScript API interface.

> > **Arguments**

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

> > **Return type** undefined

3.2.3 Commands

Trigger tracking on demand

Trigger custom event

Trigger (custom) events bound to user actions:

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

category
> **Required** `string` Event category.

action
> **Required** `string` Event action.

name
> **Optional** `string` Event name.

value
> **Optional** `number` Event value.

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

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

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

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

Track goal conversion

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

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

goal_name
> **Required** `string` Goal Name

goal_value
> **Optional** `number` Tracked conversion value.

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

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

Example of usage:

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

Ecommerce tracking

Adding Ecommerce item

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

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

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

---

productSKU
> **Required** `string` Product stock-keeping unit.

productName
> **Optional** `string` Product name.

productCategory
> **Optional** `Array<string>|string` Product category, can be written as Array with up to 5 elements.

productPrice
> **Optional** `number` with product price.

productQuantity
> **Optional** `number` with product quantity.

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

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

Example of usage:

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

Remove Ecommerce item

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

```
_paq.push(["removeEcommerceItem", productSKU]);
```

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

productSKU
    Required `string` Product stock-keeping unit.

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

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

Example of usage:

```
_paq.push(["removeEcommerceItem", "craft-311"]);
```

Clear Ecommerce items

To clear all e-commerce items (for example to track changes in the user's cart using
`trackEcommerceCartUpdate`), use the `clearEcommerceCart` function:

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

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

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

Example of usage:

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

Tracking Ecommerce order

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

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

orderId
>   **Required** `string` Unique order ID.

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

orderSubTotal
>   **Optional** `number` Order subtotal - without shipping.

orderTax
>   **Optional** `number` Order tax amount.

orderShipping
>   **Optional** `number` Order shipping costs.

orderDiscount
>   **Optional** `number` Order discount amount.

Example of usage:

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

Updating cart

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

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

cartAmount
>   **Required** `number` Cart amount (sum of products).

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

Example of usage:

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

Tracking product / category view

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

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

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

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

productCategory
>   **Optional** `Array<string>|string` Product category, can be written as Array with up to 5 elements.

productPrice
  Optional `number` Product price.

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

Example of usage:

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

Custom Variables

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

Adding / Editing Custom Variable

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

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

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

name
  Required `string` Name of the variable

value
  Optional `string` Value of the variable limited to 200 characters.

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

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

Warning: Index is separate for each variable scope.

Example of usage:

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

Removing Custom Variable

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

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

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

---

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

Example of usage:

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

Accessing Custom Variable

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

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

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

> **Arguments**

> - **index** (*number*) – **Required** Number from 1 to 5 where variable is stored

> - **scope** (*string*) – **Optional** Scope of the variable, "visit" or "page". Default value is `"visit"`

Example of usage:

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

Custom Dimensions

Tracking Custom Dimension

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

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

customDimensionID
> **Required** `number` ID of dimension

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

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

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

Example of usage:

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

Tracking Custom Dimension (legacy)

Deprecated since version 15.3: Function *setCustomDimension()* is deprecated due to difficulty of use (passed values should be URL encoded). Please use *setCustomDimensionValue()* instead.

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

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

customDimensionID
> **Required** number ID of dimension

customDimensionValue
> **Required** string Value of Custom Dimension - limited to 255 characters. The value provided to the function should be URL encoded.

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

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

Example of usage:

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

Retrieving Custom Dimension

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

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

getCustomDimensionValue(*index*)

> **Arguments**

> > • **index** (*number*) – **Required** Index of custom dimension

Example of usage:

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

Retrieving Custom Dimension (legacy)

Deprecated since version 15.3: Function `getCustomDimension()` is deprecated due to the difficulty of use (returned values should be URL encoded). Please use `getCustomDimensionValue()` instead.

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

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

getCustomDimension(*index*)

> **Arguments**
>
> > • **index** (*number*) – **Required** Index of custom dimension

Example of usage:

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

Content Tracking

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

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

Tracking all content impressions within a page

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

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

Tracking all visible content impressions

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

Code:

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

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

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

---

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

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

Example of usage:

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

Example of usage:

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

Tracking only content impressions for specific page part

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

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

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

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

Example of usage:

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

Track interactions manually with auto detection

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

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

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

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

Example of use

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

Track impression manually

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

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

`contentName`
> **Required** `string` Name of Content Impression

`contentPiece`
> **Required** `string` Name of Content Impression Piece

`contentTarget`
> **Required** `string` URL of Content Impression Target

Example of use:

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

Track user interaction manually

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

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

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

`contentName`
> **Required** `string` Name of Content Impression

`contentPiece`
> **Required** `string` Name of Content Impression Piece

`contentTarget`
> **Required** `string` URL of Content Impression Target

Example of use:

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

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

Download and Outlink Tracking

Tracking Outlink

To enable the Download & Outlink tracking, run:

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

just after the first `trackPageView` or `trackEvent`

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

Ignoring alias domains

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

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

domains
> **Required** `array` Domains written as strings, * are accepted.

Example of usage:

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

Force Tracking using CSS class

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

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

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

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

Example of usage:

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

Force Tracking using JS function

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

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

linkAddress
> **Required** `string` Address that link points to.

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

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

Example of usage

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

Tracking Downloads

Default extensions recognized as download

The following extensions are tracked as download by default:

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

Adding extension to default extensions

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

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

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

Example of usage:

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

Replacing default extensions list

Default extensions list can be overwritten using the setDownloadExtensions function:

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

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

Example of usage:

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

Force Tracking download using CSS class

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

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

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

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

Example of usage:

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

Force Tracking download using JS function

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

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

linkAddress
> **Required** `string` Address that link points to.

dimensions
> **Optional** `object` *Custom dimension* that should be tracked with this action. Can be multiple dimensions. Written as object property using `dimension{ID}` notation.
>
> Example:
>
> ```
> {
>     dimension1: "example value",
>     dimension2: "example value"
> }
> ```

Example of usage

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

Setting link tracking delay

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

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

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

Disabling tracking

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

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

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

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

User ID Management

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

Setting UserId

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

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

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

Reset UserId

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

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

Miscellaneous

Custom page name

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

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

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

Example of usage:

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

Measuring user time spent on web page

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

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

Example of usage:

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

Note: First heart beat will be sent after 15 seconds and each heart beat following it will sent with longer and longer intervals (up to 5 minute ceiling). When page will loose focus, heart beats will be paused until focus is restored. Heart beats will stop after 30 minutes from last page view.

Tracking internal searches

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

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

keyword
> **Optional** `string` Keyword that was searched

category
> **Optional** `string` Category seleted in search engine - you can set it to false when not used.

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

dimensions
> **Optional** `object` *Custom dimension* that should be tracked with this action. Can be multiple dimensions. Written as object property using `dimension{ID}` notation.
>
> Example:
>
> ```
> {
>     dimension1: "example value",
>     dimension2: "example value"
> }
> ```

Example of usage:

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

Tracking user anonymously

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

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

isAnonymous
> **Required** `boolean` Flag that sets anonymous tracking on and off

Example of usage:

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

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

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

Send ping request on demand

Ping method sends request that will update session values without creating new event or page view. Most common use for this method is update of session custom dimensions or custom variables.

Example of usage:

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

Gathering navigation timing page performance metrics

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

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

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

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

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

Example of usage:

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

Note: In order for this setting to make effect *setTimingDataSamplingOnPageLoad()* should be used before the *trackPageView()* function

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

3.3 Tracker Object Functions

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

3.3.1 Accessing Tracker Object

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

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

>>   **Arguments**

>>>   • **`trackerUrl`** (*string*) – **Required** URL for Tracker

>>>   • **`siteId`** (*string*) – **Required** Site ID that will be linked to tracked data.

>>   **Returns** Analytics Tracker instance

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

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

>>   **Arguments**

>>>   • **`trackerUrl`** (*string*) – **Required** URL for Tracker

>>>   • **`siteId`** (*string*) – **Required** Site Id that will be linked to tracked data.

>>   **Returns** Analytics Tracker instance

3.3.2 Tracking functions

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

>>   **Arguments**

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

`trackEvent`(*category*, *action*[, *name*, *value*])
>   Tracks events that should not trigger on page loading, but only when user performs an action

>>   **Arguments**

>>>   • **`category`** (*string*) – **Required** Category of event.

>>>   • **`action`** (*string*) – **Required** Event action, for example `"link click"`.

>>>   • **`name`** (*string*) – **Optional** Event name, for example `"Cancel button"`.

>>>   • **`value`** (*number*) – **Optional** Event value.

`trackGoal`(*idGoal*[, *customRevenue*, *customData*])
>   Manually tracks goal (conversion).

>>   **Arguments**

- **idGoal** (*int/string*) – **Required** Id of goal.

- **customRevenue** (*int/float*) – **Optional** Revenue value

- **customData** (*mixed*) – **Optional** Object with *Custom dimensions*.

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

The function that tracks internal site searches.

> Arguments

- **keyword** (*string*) – **Required** String containing keyword that was searched.

- **category** (*string/boolean*) – **Optional** String with category selected in search engine, can set it to false when not used.

- **searchCount** (*number/boolean*) – **Optional** Number of results on the results page, can be set to false when not used.

enableHeartBeatTimer()

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

---

Note: First heart beat will be sent after 15 seconds and each heart beat following it will sent with longer and longer intervals (up to 5 minute ceiling). When page will loose focus, heart beats will be paused until focus is restored. Heart beats will stop after 30 minutes from last page view.

---

enableCrossDomainLinking()

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

setCrossDomainLinkingTimeout(*timeout*)

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

> Arguments

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

getCrossDomainLinkingUrlParameter()

Returns the query parameter that can be appended to link URLs so cross domain visits can be detected. If your application creates links dynamically, then you'll have to add this query parameter manually to those links (since the JavaScript tracker cannot detect when those links are added).

Eg:

```
var url = 'http://myotherdomain.com/?' + piwikTracker.
getCrossDomainLinkingUrlParameter(); $element.append('<a href="' + url
+ '"/>');
```

3.3.3 Ecommerce tracking

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

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

> Arguments

- **productSKU** (*string*) – **Required** String with product stock-keeping unit.

- **productName** (*string*) – **Optional** String with product name.

- **productCategory** (*Array<string>|string*) – **Optional** Product category, can be written as Array with up to 5 elements.

- **price** (*string*) – **Optional** String with product price.

- **quantity** (*string*) – **Optional** String with product quantity.

removeEcommerceItem (*productSKU*)
    The function that removes ecommerce item, can be used when removing items from cart.

    Arguments

- **productSKU** (*string*) – **Required** String with product stock-keeping unit.

clearEcommerceCart ()
    The function that clears all ecommerce items, can be used when cart is deleted.

getEcommerceItems ()
    The function that returns all ecommerce items, can be used to check state of tracked cart.

    Response example:

        {"craft-311":["craft-311","Unicorn Iron on Patch","Crafts & Sewing","499","3"]}

    Returns Object containing all tracked items (format: `Object<productSKU, Array[productSKU, productName, productCategory, price, quantity]>`).

trackEcommerceOrder (*orderId*, *orderGrandTotal*[, *orderSubTotal*, *orderTax*, *orderShipping*, *orderDiscount*])
    The function that tracks Ecommerce order, also tracks all items previously added.

    Arguments

- **orderId** (*string*) – **Required** Unique order ID.

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

- **orderSubTotal** (*number*) – **Optional** Order subtotal - without shipping.

- **orderTax** (*number*) – **Optional** Order tax amount.

- **orderShipping** (*number*) – **Optional** Order shipping costs.

- **orderDiscount** (*number*) – **Optional** Order discount amount.

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

    Arguments

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

setEcommerceView (*productSKU*[, *productName*, *productCategory*, *productPrice*])
    The function to track product or category page view, must be followed by the `trackPageView` function.

    Arguments

- **productSKU** (*string*) – **Required** String with product stock-keeping unit.

- **productName** (*string*) – **Optional** String with product name.

- **productCategory** (*Array<string>|string*) – **Optional** Product category, can be written as Array with up to 5 elements.

- **price** (*string*) – **Optional** String with product price.

3.3.4 Custom variables

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

setCustomVariable (*index*, *name*, *value*, *scope*)
    The function that sets a custom variable to be used later.

        Arguments

- **index** (*string*) – **Required** Number from 1 to 5 where variable is stored.

- **name** (*string*) – **Required** Name of the variable.

- **value** (*string*) – **Required** Value of the variable.

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

deleteCustomVariable (*index*, *scope*)
    The function that will delete a custom variable.

        Arguments

- **index** (*string*) – **Required** Number from 1 to 5 where variable is stored.

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

getCustomVariable (*index*, *scope*)
    The function that will return the value of custom variable.

        Arguments

- **index** (*string*) – **Required** Number from 1 to 5 where variable is stored.

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

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

3.3.5 Custom dimensions

setCustomDimensionValue (*customDimensionId*, *customDimensionValue*)
    New in version 15.3.

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

        Arguments

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

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

setCustomDimension (*customDimensionId*, *customDimensionValue*)
    Deprecated since version 15.3: Function *setCustomDimension()* is deprecated due to the difficulty of use (passed values should be URL encoded). Please use *setCustomDimensionValue()* instead.

    Function that sets a custom dimension to be used later.

        Arguments

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

- **customDimensionValue** (*string*) – **Required** Value of custom dimension (value should be URL encoded).

deleteCustomDimension (*customDimensionId*)

The function that will delete a custom dimension.

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

getCustomDimensionValue (*customDimensionId*)

New in version 15.3.

The function that will return the value of custom dimension.

> **Arguments**
>
> - **customDimensionId** (*string*) – **Required** Id of custom dimension.
>
> **Returns** Value set with *setCustomDimensionValue()*
>
> **Return type** string

getCustomDimension (*customDimensionId*)

Deprecated since version 15.3: Function *getCustomDimension()* is deprecated due to the difficulty of use (returned values should be URL encoded). Please use *getCustomDimensionValue()* instead.

The function that will return the value of custom dimension.

> **Arguments**
>
> - **customDimensionId** (*string*) – **Required** Id of custom dimension.
>
> **Returns** Value set with *setCustomDimension()*
>
> **Return type** string

3.3.6 Content Tracking

Impressions

trackAllContentImpressions ()

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

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

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

> **Arguments**
>
> - **checkOnScroll** (*boolean*) – **Optional** Enables tracking content blocks that will be visible after scroll event.
>
> - **watchInterval** (*number*) – **Optional** Interval, in milliseconds between checking for new visible content. Periodic checks can be disabled for performance reasons by setting 0. Default value: 750.

trackContentImpressionsWithinNode (*domNode*)

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

> **Arguments**

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

trackContentImpression(*contentName*, *contentPiece*, *contentTarget*)
> The function that manually tracks content impression.

> **Arguments**

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

logAllContentBlocksOnPage()
> The function that will print all content blocks in the console for debugging purposes.

Interactions

trackContentInteractionNode(*domNode*[, *contentInteraction*])
> The function that tracks interaction within domNode. This can be used as a function inside the onClick attribute.

> **Arguments**

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

trackContentInteraction(*contentInteraction*, *contentName*, *contentPiece*, *contentTarget*)
> The function that tracks content interaction using the given data.

> **Arguments**

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

3.3.7 Download and Outlink Tracking

trackLink(*url*, *linkType*[, *customData*, *callback*])
> The function that will manually track downloads or outlinks, depending on type.

> **Arguments**

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

Tracking Outlink

enableLinkTracking(*enable*)

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

> **Arguments**
>
> > • **enable** (`boolean`) – **Required** Set it to true to track links, false to disable tracking.

setLinkClasses(*classes*)

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

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

Tracking Downloads

Default extensions recognized as download

The following extensions are tracked as download by default:

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

setDownloadClasses(*classes*)

The function that sets classes to be treated as downloads (`piwik_download` is the default one).

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

setDownloadExtensions(*extensions*)

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

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

addDownloadExtensions(*extensions*)

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

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

removeDownloadExtensions(*extensions*)

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

> **Arguments**

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

setLinkTrackingTimer(*time*)
 The function that will set delay between tracking and download;

  Arguments

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

getLinkTrackingTimer()
 The function that will return delay between tracking and download.

Disabling tracking

setIgnoreClasses(*classes*)
 The function that will set classes to be ignored in tracking download and outlinks.

  Arguments

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

3.3.8 User ID and Visitor ID Management

User ID

getUserId()
 The function that will return user ID.

setUserId(*userId*)
 The function that will set user ID to this user.

  Arguments

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

resetUserId()
 The function that will reset user ID value.

Visitor ID

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

getVisitorInfo()
 The function that will return visitor information in an array:

- new visitor flag indicating new (`1`) or returning (`0`) visitor

- visitor ID (UUID)

- first visit timestamp (Unix epoch time)

- previous visit count (`0` for first visit)

- current visit timestamp (Unix epoch time)

- last visit timestamp (Unix epoch time or `''` if N/A)

- last e-commerce order timestamp (Unix epoch time or `' '` if N/A)

3.3.9 Tracking cookies management

Cookies that are used by analytics are first party cookies.

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

enableCookies()
> The function that will enable all first party cookies. Cookies will be created on first sent tracking request.

> **Note:** Tracker has cookies enabled by default.

---

deleteCookies()
> The function that will delete existing tracking cookies after the next page view.

hasCookies()
> The function that will return `true` if cookies are enabled in this browser.

setCookieNamePrefix(*prefix*)
> The function that will set the prefix for analytics tracking cookies. Default is `"_pk_"`

> **Arguments**

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

setCookieDomain(*domain*)
> The function that will set the domain for the analytics tracking cookies.

> **Arguments**

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

setCookiePath(*path*)
> The function that will set the analytics tracking cookies path.

> **Arguments**

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

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

> **Arguments**

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

setVisitorCookieTimeout(*seconds*)
> The function that will set the expiration time of visitor cookies.

> **Arguments**

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

setReferralCookieTimeout(*seconds*)
> The function that will set the expiration time of referral cookies.

> **Arguments**

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

setSessionCookieTimeout(*seconds*)
> The function that will set the expiration time of session cookies.

> > **Arguments**

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

3.3.10 Tracker Configuration

setDocumentTitle([*title*])
> The function that will set the document tile that is being sent with tracking data.

> > **Arguments**

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

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

> > **Arguments**

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

setCustomUrl(*customUrl*)
> The function that will override tracked page URL. Tracker will use current page URL if custom URL was not set.

> > **Arguments**

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

setReferrerUrl(*url*)
> The function that will override the detected HTTP referrer.

> > **Arguments**

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

setApiUrl(*url*)
> The function that will set the Analytic's HTTP API URL endpoint. Usually the root directory of analytics.

> > **Arguments**

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

getPiwikUrl()
> The function that will return the Analytic's server URL.

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

discardHashTag(*enableFilter*)
> The function that will set tracker to include or remove URL fragment identifier from tracked URLs.

> > **Arguments**

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

setGenerationTimeMs (*generationTime*)
  The function that overrides DOM Timing API provided time needed to download the page.

  > Arguments

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

appendToTrackingUrl (*appendToUrl*)
  The function that will append a custom string to the tracking URL.

  > Arguments

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

setDoNotTrack (*enable*)
  The function that will disable tracking users who set the Do Not Track setting.

  > Arguments

  > > • **enable** (`boolean`) – **Required** When set to true tracking wont occur.

killFrame ()
  The function that will block a site from being iframed.

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

  > Arguments

  > > • **url** (`string`) – **Required** URL that should be loaded.

setCampaignNameKey (*name*)
  The function that will set campaign name parameters.

  > Arguments

  > > • **name** (`string`) – **Required** Campaign name.

setCampaignKeywordKey (*keyword*)
  The function that will set campaign keyword parameters.

  > Arguments

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

3.3.11 Anonymization

setUserIsAnonymous (*isAnonymous*)
  The function that will set user anonymous tracking.

  > Arguments

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

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

3.3.12 Advanced Usage

ping ()
  Ping method sends request that will update session values without creating new event or page view. Most common use for this method is update of session custom dimensions or custom variables.

addListener(*domElement*)

   The function will add a click listener to link element.

   > Arguments

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

setRequestMethod(*method*)

   The function that will set the request method.

   > Arguments

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

setCustomRequestProcessing(*function*)

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

setRequestContentType(*contentType*)

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

   > Arguments

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

customCrossDomainLinkDecorator(*urlDecorator*)

   The function that set custom cross domain decorator used on links to pass visitor ID via URL (used by *enableCrossDomainLinking()*). It will be later parsed by *customCrossDomainLinkVisitorIdGetter()*.

   > Arguments

   > > • **urlDecorator** (*function*) – function that will decorate URL with visitor ID passed by URL

urlDecorator(*url*, *value*, *name*)

   Decorator function accepts link URL and visitor ID value and parameter name and returns URL containing visitor ID data.

   > Arguments

   > > • **url** (*string*) – **Required** Link URL

   > > • **value** (*string*) – **Required** Value of visitor ID that should be passed via URL

   > > • **name** (*string*) – **Required** Name of visitor ID parameter used by tracker (can be set)

   > Returns Decorated URL or `null` (no change in URL)

   > Return type string|null

Note: Usage example: value send via URL query parameter (equivalent of default implementation).

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

---

customCrossDomainLinkVisitorIdGetter(*urlParser*)
>   The function that set custom cross domain URL parser (decorated by function set via *customCrossDomainLinkDecorator()*). It returns value of visitor ID parsed from page URL (used by *enableCrossDomainLinking()*).

>>   **Arguments**

>>>   - **urlParser** (*function*) – function that will parse URL and return value of visitor ID passed through it.

urlParser(*url*, *name*)
>   Parser function that accepts page URL and visitor ID parameter name and returns visitor ID value.

>>   **Arguments**

>>>   - **url** (*string*) – **Required** page URL

>>>   - **name** (*string*) – **Required** name of visitor ID param used by tracker (can be set)

>>   **Returns**   Visitor ID value (parsed from URL)

>>   **Return type**   string

Note: Usage example: value send via URL query parameter (equivalent of default implementation).

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

---

enableJSErrorTracking()
>   Enables tracking of unhandled JavaScript errors

Note: Browsers may limit information about error details if it occurs in script loaded from different origin (see details).

3.4 HTTP API

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

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

3.5 Web Log Analytics

3.5.1 Set up log import

This step requires a little more familiarity with Bash, and around 4 to 10 hours of time depending on the volume of data.

You need to run the Log Importer tool with the correct parameters. Some of them must be present, while others are optional.

Sample command:

```
import_logs.py --url=https://demo.piwik.pro --token-
↪auth=XXXXXXXXXXXXXXXXXXXXXXXXXXXXX --enable-static --enable-bots --show-progress --
↪idsite=X --recorders=2 --recorder-max-payload-size=50 sample.log
```

--url=https://demo.piwik.pro
   This is a mandatory parameter which points to the location of your Piwik instance

--token-auth=XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   Authentication token with superuser rights

--idsite=X
   Defines the Site ID of the website. It can be either integer (eg. *1*) or UUID (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*.

Once the log importer tool finishes parsing and uploading logs to your Piwik instance, you will have to wait for the archiving process to populate the Piwik reports with new data.

The time needed for this process depends on the amount of the data you've uploaded and in rare cases may even take a couple of days (for example, uploading years of historical data), but usually it is a matter of around an hour.

If you plan to import logs on a regular basis it is advised to setup a scheduled job using a tool such as CRON.

3.5.2 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.

3.5.3 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.

- **–recorder-max-payload-size=N** The importer uses the Piwik PRO bulk tracking feature in order to boost speed. 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:

```
import_logs.py --help
```

3.5.4 Import data with server log analytics and standard JavaScript simultaneously

JavaScript Tracking 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.

3.5.5 Reprocess reports after the log import

Note: Information in this section doesn't apply to Piwik PRO cloud, only to the on-premises Piwik PRO web analytics stack.

Your first run of Log Analytics will potentially import a very large amount of historical data, even months or years worth.

After this first process is completed, run this command to archive all historical reporting data:

```
./console core:archive --force-all-websites --force-all-periods=31557600 --force-date-
→last-n=1000 --piwik-domain=demo.piwik.pro
```

Next, place the following command into a cron to process archives of logs imported at hourly or daily intervals:

```
./console core:archive --piwik-domain=demo.piwik.pro
```

If you are planning to process a very large volume of initial data in your first run, please contact us at support@piwik.pro for help.

3.5.6 Technical requirements

Technical requirements for running Web Log Analytics:

- Access to the server or server logs – for example via SSH

- Python 2.6 or 2.7 – versions 3.x 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 Importer tool – this is a script written in Python ensuring that logs are sent to your Piwik instance.

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

CHAPTER 4

Audience Manager

4.1 Profile data

key	type	description

id	uuid	ID of profile.
		Example:
		
		`"d9a614a1-1234-11ea-a72c-`
		`↪0202c0f2d936"`

website_id	uuid	ID of the website.
		Example:
		
		`"5dff7262-731e-291d-ad23-`
		`↪d1aea83ecd51"`

user_id	string	Value of user id from the Analytics. Example: `"ff1063df11"`
email	string	Email address of the user (detected from submitted form or imported from e.g. CSV). Example: `"test@example.com"`
analytics_visitor_id	string	Analytics ID of the user. Value of cookie analytics_visitor_id. Example: `"b3d31070825871e1"`
analytics_visitor_ids	list	List of analytics_visitor_ids. Example: `["d40bb72cc59e9ef3",`

device_ids	list	List of device IDs. Example:

4.2 JavaScript API

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

4.2.1 Loading snippet

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

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

Configuration:

- String `XXX-XXX-XXX-XXX-XXX` should be replaced with *app ID* (e.g. `efcd98a5-335b-48b0-ab17-bf43f1c542be`).

- String `https://your-instance-name.piwik.pro/` should be replaced with your PPAS instance address. (please note that it's used in 2 places in the snippet).

Code:

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

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

This code initializes the API interface in the following ways:

1. Creates a `<script>` tag that asynchronously loads the Audience Manager API library.

2. Initializes the global `ppms.am.api` command queue that schedules commands to be run when the API library is loaded.

3. Schedules `create` command on `ppms.am.api` to initialize the API object with a basic PPAS configuration.

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

4.2.2 Command queue

Executing the snippet creates the following global function:

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

Audience Manager API command queue.

> **Arguments**
>
> > - **`command`** (*string*) – Command name.
> >
> > - **`args`** – Command arguments. The number of arguments and their function depend on command.
>
> **Returns** Commands are expected to be run asynchronously and return no value.

Return type undefined

4.2.3 Commands

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

Get list of audiences user belongs to

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

Code:

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

onFulfilled(*audience_list*)
    The fulfilment handler callback (called with result).

> **Arguments**
>
> > • **audience_list** (*Array<string>*) – **Required** Array of *audience* IDs the *user* belongs to.
> >
> > Example:
> >
> > ```
> > ["e8c6e873-955c-4771-9fd5-92c94577e9d9", "756e5920-422f-4d13-b73a-
> > →917f696ca288"]
> > ```

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

> **Arguments**
>
> > • **error_code** (*string*) – **Required** Error code.
> >
> > Example:
> >
> > ```
> > "server_error"
> > ```

Check user membership in the audience

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

Code:

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

audience_id
    Required string ID of the checked *audience*.

> Example:
>
> ```
> "52073260-5861-4a56-be5e-6628794722ee"
> ```

onFulfilled(*in_audience*)
    The fulfilment handler callback (called with result).

> **Arguments**

- **in_audience** (`boolean`) – **Required** *True* when *user* is part of the *audience*, *false* otherwise.

    Example:

    ```
    true
    ```

onRejected(*error_code*)

   The rejection handler callback (called with error code).

   Arguments

- **error_code** (`string`) – **Required** Error code.

    Example:

    ```
    "server_error"
    ```

Get user attributes

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

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

Code:

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

onFulfilled(*attributes*)

   The fulfilment handler callback (called with result).

   Arguments

- **attributes** (`Object<string,Object<string, (string|number|Array<string>)>>`) – **Required** Object containing *user attributes* divided by source.

    – *analytics* - `Object<string,string>` Contains *analytics attributes* about the *user* (e.g. browser name, browser version, country).

    – *attributes* - `Object<string,(string|number|Array<string>)>` Contains *custom attributes* about the *user* (e.g. first name, last name, email).

    Example:

    ```
    {
        "analytics": {
            "browser_name": "chrome",
            "country": "us"
        },
        "attributes": {
            "favourite_brands": ["Alfa Romeo", "Aston Martin"],
            "age": 32,
            "first_name": "James",
    ```

    (continues on next page)

```
            "last_name": "Bond"
    }
}
```

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

　　　　Arguments

　　　　　　• **error_code** (*string*) – **Required** Error code.

　　　　　　Example:

```
"server_error"
```

Update user attributes

Creates or updates *user custom attributes*.

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

Code:

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

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

　　　　• key (string) - *attribute* name

　　　　• value (string|number|Array<string>|object) - Value of the *attribute*. System will process it differently depending on its type:

　　　　　　– string - overwrite the *attribute* value with the new value. If the *attribute* was not used before - creates new text *attribute*.

　　　　　　– number - overwrite the *attribute* value with the new value. If the *attribute* was not used before - creates new numeric *attribute*.

　　　　　　– Array<string> - overwrite the *attribute* value with the new set of values. If the *attribute* was not used before - creates new text *attribute* with a list of values.

　　　　　　– object - ModificationAction using following format: {action: string, value: (string|number)}. It allows to manipulate *attribute* value using one of the following ModificationAction action values:

　　　　　　　　* "set" - overwrite *attribute* value using the ModificationAction value. Works identically to the shorter versions using string, number or Array<string> types.

　　　　　　　　* "add" - add the ModificationAction value (or 1, if not specified) to the *attribute* value.

　　　　　　　　Note:

　　　　　　　　　　· Works only on numeric *attributes*.

· `ModificationAction value` can be any `number` (including negative and fractional numbers).

· If the *attribute* was not used before - creates new `numeric` *attribute* and sets its value to `0` before performing action.

* `"list-add"` - add the `ModificationAction value` to the list of *attribute* values or extend single value *attribute* to a list of values. New value will be a list containing previous value(s) in addition to the added value.

Note:

· Only string values are allowed on the list or can be extended to a list.

· List values are unique. Adding value that already was on the list will not modify the list.

* `"list-remove"` - remove the `ModificationAction value` from the list of *attribute* values or delete single value *attribute*. New value will be a list containing previous value(s) without the removed value.

Note:

· Only string values are allowed on the list.

Example:

```
{
    "favourite_color": "black",
    "drink": "Martini",
    "code_number": 7,
    "aliases": ["Peter", "Conrad", "Patrick", "Bill"],
    "kill_count": {
        "action": "add",
        "value": 3,
    },
    "favourite_brands": {
        "action": "list-add",
        "value": "Land Rover",
    },
    "current_missions": {
        "action": "list-remove",
        "value": "Casino Royale",
    },
}
```

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

Example:

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

```
    "onRejected": onRejected
}
```

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

> Example:

```
"jbond"
```

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

> Example:

```
"1234567890ABCDEF"
```

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

> Example:

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

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

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

> > **Arguments**

> > > • **error_code** (*string*) – **Required** Error code.

> > > > Example:

```
"server_error"
```

4.3 Form Tracker

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

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

4.3.1 Supported browsers

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

4.3.2 Privacy by design

PPAS follows "Privacy by design" approach to system engineering.

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

Private information

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

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

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

Configuration

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

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

4.3.3 Installation

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

Using Tag Manager

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

Manual installation

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

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

- String `XXX-XXX-XXX-XXX-XXX` should be replaced with *app ID* (e.g. `efcd98a5-335b-48b0-ab17-bf43f1c542be`).

- String `https://your-instance-name.piwik.pro//` should be replaced with your PPAS instance address (please note that it's used in 3 places in the snippet).

Changed in version 10.0.

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

New in version 6.3.

`forms_config`
> **Required** `Object<string,(boolean|{type: string, fields: Array<string>})>`
> Configuration of tracked forms. Default configuration requires that all tracked forms are specified in this object as keys. Each key is another form ID.
>
> Value of each key can be specified in 2 ways:
>
> - `true` - All fields in form using this ID will be tracked (this behavior can be changed using `trackingType` option).
>
> - `Object` - Specifies which fields will be included or excluded from the form.
>
>   **`type`**
>   > **Required** `"whitelist"|"blacklist"` Defines type of form fields filter.
>
>   **`fields`**
>   > **Required** `Array<string>` Lists field names used by the filter. Default configuration identifies fields by input `name` attribute, but `useLabels` option can change this behavior.
>
> Example:

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

New in version 6.3.

options
> Optional `object` Options that change behavior of the tracker.

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

> Deprecated since version 6.3.

> **trackingType**
> > Optional `"whitelist"|"blacklist"` Defines what is default strategy of form configuration. Default value: `"whitelist"`.
> >
> > - `"whitelist"` - All form IDs that are not set in `forms_config` are ignored by the tracker.
> >
> > - `"blacklist"` - All form IDs that are set in `forms_config` and use `true` value are ignored by the tracker. Forms defining filtered fields are tracked according to specified fields filter. All other forms are tracked as a whole.
> >
> > ---
> > **Note:** This option is intended for backward compatibility and is planned to be removed in the future.
> > ---

> Example:
>
> ```
> {
>     useLabels: true,
> }
> ```

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

1. Creates a `<script>` tag that asynchronously loads Audience Manager Form Tracker library.

2. Initializes global `ppms.am.form` command queue that schedules commands to be run when Form Tracker library is loaded.

3. Schedules creation of Form Tracker instance (using `ppms.am.form` function).

4.4 Public HTTP API

4.5 Authorized HTTP API

CHAPTER 5

Consent Manager

5.1 Custom consent form

5.1.1 Enable custom consent form

Consent Manager's JS API enables you to build a custom consent form in place of the default one.

To turn on Custom consent form mode:

1. Go to Administration module

2. Go to "Websites & apps" tab

3. In "Settings" section, find "Privacy" settings

4. Ensure that "Ask visitors for consent" is checked

5. Enable "Use a custom consent form" option

Then you can build a form using the JavaScript API.

5.1.2 Example implementation

Visit Piwik PRO - Custom consent form example page to discover a live demo of Custom consent form implementation.

5.2 JavaScript API

5.2.1 Introduction

Consent Manager provides a JavaScript API that allows the user to:

- Get compliance types

- Get new compliance types

- Set initial compliance settings

- Set compliance settings

- Get compliance settings

- Send data subject request

- New in version 12.0: Open consent form

- New in version 15.3: Track consent stats

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

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

> **Arguments**
>
> - **`command`** (*string*) – Command name
>
> - **`args`** – Command arguments. The number of arguments and their function depend on command.
>
> **Returns** Commands are expected to be run asynchronously and return no value
>
> **Return type** undefined

Consent Manager is fully integrated with Tag Manager. If you already have asynchronous snippet installed, then you are able use Consent Manager's JavaScript API.

5.2.2 Commands

All commands work in the context of the current visitor and website. Additionally, they sometimes require communication with a PPAS server and are asynchronous. Callback functions are used to provide response value or information about errors. `onSuccess(...args)` callback is 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.

Get compliance types

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.

Code:

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

`onFulfilled`(*types*)
> **required** The fulfillment handler callback (called with result)
>
> **Arguments**
>
> - **`types`** (*Array<string>*) – **Required** Array of consent types
>
>   Example:
>
>   ```
>   ["remarketing", "analytics"]
>   ```

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

>> **Arguments**

>>> • **error** (*string|object*) – **Required** Error code or exception

Get new compliance types

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

Code:

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

onFulfilled(*types*)
> **required** The fulfillment handler callback (called with result)

>> **Arguments**

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

>>> Example:

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

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

>> **Arguments**

>>> • **error** (*string|object*) – **Required** Error code or exception

Set initial compliance settings

Sets initial compliance settings (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.

Code:

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

settings
> **required** The consent settings object

>> Example:

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

>> or

>> Example:

```
['analytics']
```

onFulfilled()
> **required** The fulfillment handler callback

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

> > **Arguments**

> > > • **error** (*string|object*) – **Required** Error code or exception

Set compliance settings

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.

Code:

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

settings
> **required** The consent settings object

> > Example:

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

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

> > • `0` - user has rejected the consent

> > • `1` - user has approved the consent

onFulfilled()
> **required** The fulfillment handler callback

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

> > **Arguments**

> > > • **error** (*string|object*) – **Required** Error code or exception

Get compliance settings

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.

Code:

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

settings
> > **required** The consent settings object

> > > Example:

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

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

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

- `0` - user reject consent

- `1` - user approve consent

onFulfilled(*settings*)
>   **required** The 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.
>
>   >   **Arguments**
>
>   >   >   - **error** (*string|object*) – **Required** Error code or exception

Send data subject request

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

Code:

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

request
>   **required** The subject data request.
>
>   >   Example:
>
>    >    ```
>    >    {content: 'user input', email: 'example@example.org', type: 'delete_data'}
>    >    ```
>
>   >   Where `type` is request type, and can be one of:
>
>   >   - `change_data` for data alteration request
>
>   >   - `view_data` for view data request
>
>   >   - `delete_data` for delete data request

onFulfilled()
>   **required** The fulfillment handler callback

onRejected(*error*)
>   The rejection handler callback (called with error code). If not specified, the exception will be thrown in the main
>   stack trace.
>
>   >   **Arguments**
>
>   >   >   - **error** (*string|object*) – **Required** Error code or exception

Open consent form

New in version 12.0.

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.

Code:

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

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

　　　　Arguments

　　　　　　• **popupId** (*string*) – Id of the consent popup

　　　　　　　Example:

```
"ppms_cm_consent_popup_30a851b6-6bf4-45f9-9a53-583401bb5d60"
```

　　　　　　• **consentTypes** (*array<string>*) – Array of consent types

　　　　　　　Example:

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

　　　　　　• **consents** (*array<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.

　　　　Arguments

　　　　　　• **error** (*string|object*) – **Required** Error code or exception

Track Main Form view

New in version 15.3.

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

Code:

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

onFulfilled()
　　The fulfillment handler callback

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

　　　　Arguments

　　　　　　• **error** (*string|object*) – **Required** Error code or exception

Track Reminder Widget view

New in version 15.3.

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

Code:

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

onFulfilled()
:   The fulfillment handler callback

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

    Arguments

    • **error** (*string*|*object*) – **Required** Error code or exception

Track Privacy Policy Link view

New in version 15.3.

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

Code:

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

onFulfilled()
:   The fulfillment handler callback

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

    Arguments

    • **error** (*string*|*object*) – **Required** Error code or exception

Track *Agree to all* click

New in version 15.3.

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

Code:

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

onFulfilled()
:   The fulfillment handler callback

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

    Arguments

    • **error** (*string*|*object*) – **Required** Error code or exception

Track *Reject all* click

New in version 15.3.

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

Code:

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

onFulfilled()
> The fulfillment handler callback

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

> > **Arguments**

> > > • **error** (*string|object*) – **Required** Error code or exception

Track *Save choices* click

New in version 15.3.

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

Code:

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

onFulfilled()
> The fulfillment handler callback

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

> > **Arguments**

> > > • **error** (*string|object*) – **Required** Error code or exception

Track close button click

New in version 15.3.

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

Code:

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

onFulfilled()
> The fulfillment handler callback

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

> > **Arguments**

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

Tag Manager

6.1 Authorized HTTP API

6.1.1 Tags

6.1.2 Triggers

6.1.3 Variables

6.1.4 Versions

6.1.5 Changelog

6.1.6 Operations

New in version 10.0.

6.2 Custom data layer name

6.2.1 Introduction

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

6.2.2 Renaming the data layer

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

1. Log into your PPAS instance.

2. Head to *Menu > Administration*.

3. Select the website that you want to set the new data layer name for and then go to the *Installation* page.

4. From here you should copy the code of the snippet, that you want to use - asynchronous or synchronous - and change the *dataLayer* value highlighted on the screens below:

    a. for the asynchronous snippet

    b. for the synchronous snippet

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

Note:

- The snippet code was changed compared to PPAS versions pre 10.0, to accommodate for the possibility of changing the data layer name. The new snippet, released alongside version 10.0 is backward compatible, however, to use the functionality described in this document, whole snippet code needs to be replaced on your website.

- If you are using both synchronous and asynchronous snippets, it is recommended to use the same data layer name in both. Nonadherence to this rule may cause unexpected and unwanted behavior.

- We strongly advise making sure that the chosen name is not used in any other external software present on your website as well as inside your own website's code. Data layer name should be unique and reserved only for use inside the PPAS container. For additional guidelines regarding the data layer naming process, please refer to *Data layer name guidelines*.

6.2.3 Snippet migration

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

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

will become:

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

6.2.4 Data layer name guidelines

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

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

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

6.3 Content Security Policy (CSP)

6.3.1 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

6.3.2 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: 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>**.

Adding nonce to container code

Consequently, default container code requires following modifications to work:

- **asynchronous snippet** - given container code following changes (highlighted) are required:

```html
<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,
    tags.nonce="INSERT_VALID_NONCE_VALUE",
    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** - following changes (highlighted) are required:

```html
<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: 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.

6.3.3 Adjusting 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:

  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: 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).

6.3.4 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;
```

6.3.5 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 tracker domain **client.piwik.pro** for collecting visitor consents and container domain **client.containers.piwik.pro** for fetching consent form assets.

6.3.6 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>
```

6.3.7 Tracker with custom domain

If your domain for tracker is custom, then you need to define it with `img-src` and `script-src` directives:

```
img-src <your-sources> your-custom-tracker-domain.com;
script-src <your-sources> your-custom-tracker-domain.com;
```

6.3.8 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 tracker 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;
                         font-src    'self' client.containers.piwik.pro;
                         style-src   'self' client.containers.piwik.pro 'nonce-
→nceIOfn39fn3e9h3sd';
```

New in version 10.1.

6.4 Custom popup template implementation examples

6.4.1 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.

6.4.2 Example 1

Preview:
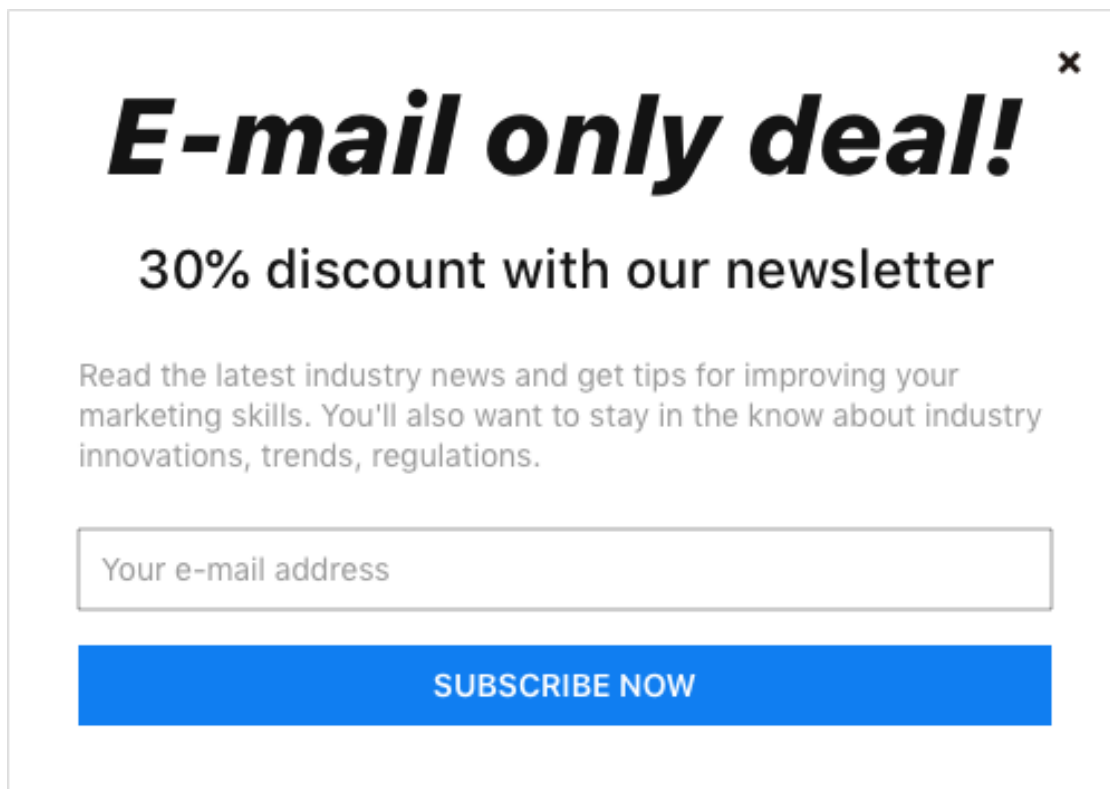
Note: Handling of the close button is provided out of the box, as long as the class name `ppms-popup-close-button` is unchanged. Your own JavaScript code to handle *Subscribe now* button needs to be provided.

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">
```

```html
        <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>
    </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;
  }
```

```css
.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;
}

.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;
      min-height: unset;
    }
  }
</style>
```

6.4.3 Example 2

Preview:
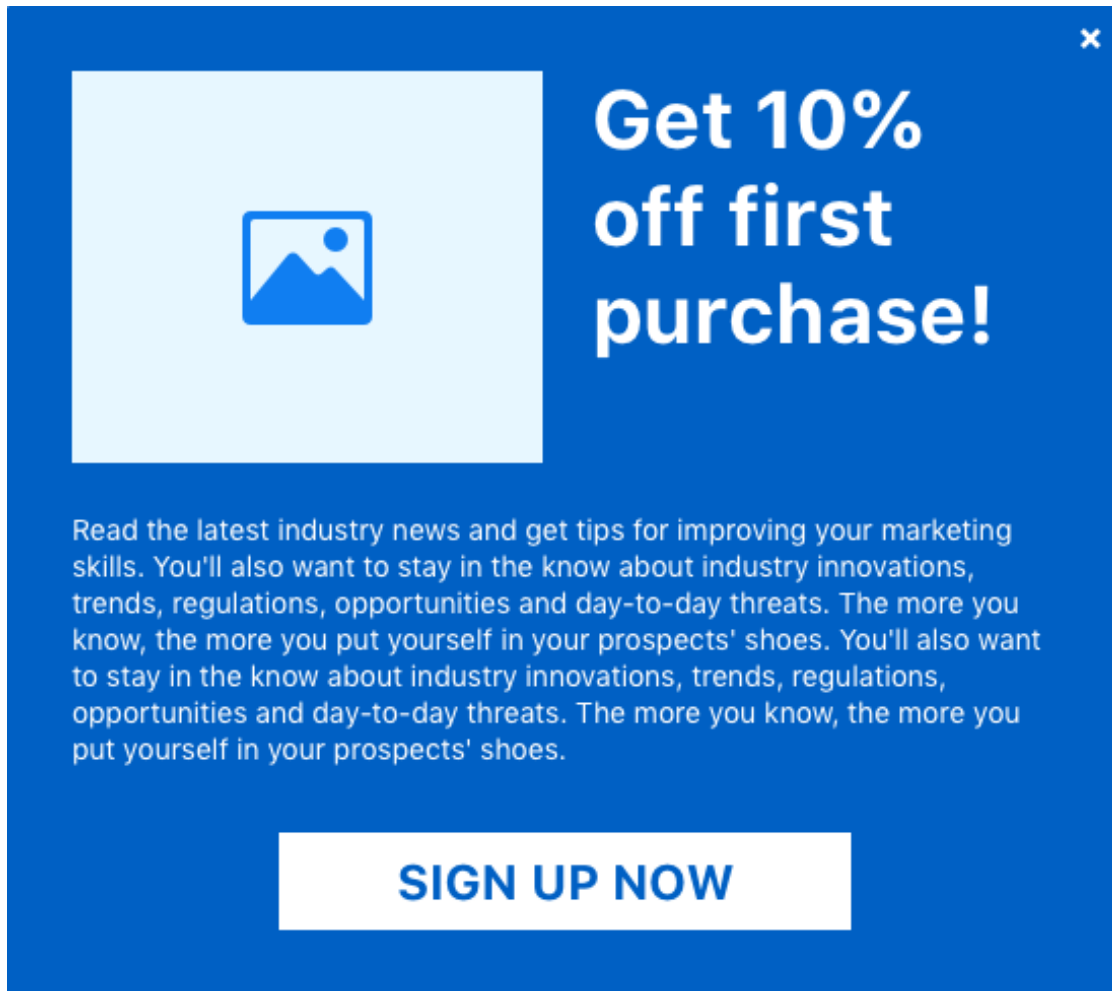
Note: Handling of the close button is provided out of the box, as long as the class name `ppms-popup-close-button` is unchanged. Your own JavaScript code to handle *Sign up now* button needs to be provided.

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 -
↪-->
```

(continues on next page)

```
    <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>
      </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>
```

```css
    </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;
  }

  .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;
```

```css
  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;
}

@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;
```

```css
    }
  }

  @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>
```

6.4.4 Example 3

Preview:
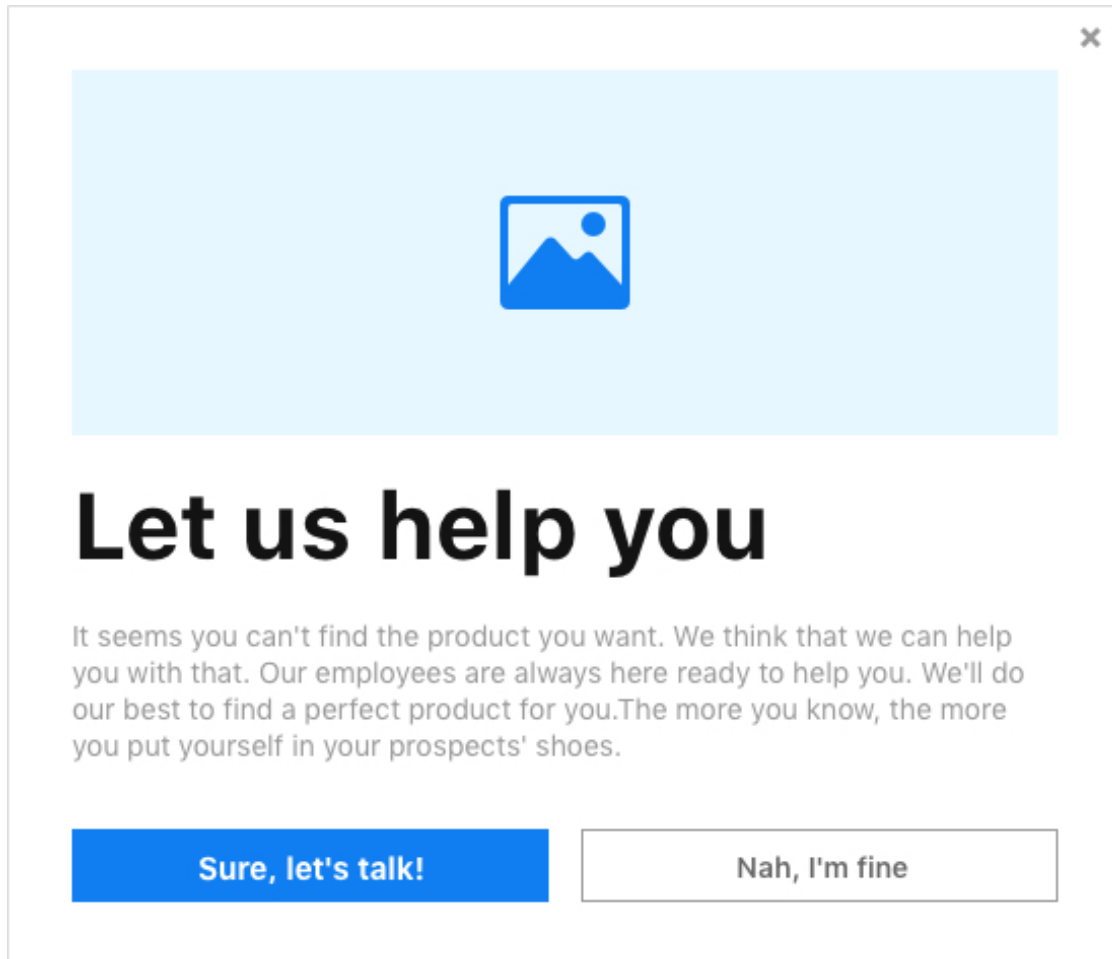
Note: Handling of the close button is provided out of the box, as long as the class name `ppms-popup-close-button` is unchanged. Your own JavaScript code to handle *Sure, let's talk* and *Nah, I'm fine* buttons needs to be provided.

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"
          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" />
```
(continues on next page)

```
          </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">
  .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 {
```

```css
    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;
  }

  .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;
  }
```

```css
.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;
  }
}
</style>
```

New in version 14.0.

6.5 Skip link tracking with *data-disable-delay* attribute

6.5.1 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.

6.5.2 Delay mechanism

Each website or app you create has the *Page redirect delay* option in its settings. You can adjust this value in *Settings -> General settings* section in your app settings under *Administration -> Websites & 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.

6.5.3 *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.

Platform

7.1 Authorized API guide

7.1.1 Introduction

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

Obtaining token

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

Generate API Credentials

- Login to your account using your email and password.

- Go to your profile (`Menu` then `User panel`).

- On this page click on `API Credentials` tab. This page allows you to manage all your API credentials.

- Click `Generate new credentials` which will result in new popup. Fill in your custom `credentials name`. Name must contains at least 3 characters.

- Copy your newly generated `CLIENT ID` and `CLIENT SECRET` because they **won't be available for you after dismissing this window.**

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

Create access token

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

Piwik PRO API tokens use JWT format.

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

Response example:

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

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

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

Deleting API Credentials

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

7.1.2 API usage example

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

API usage example with curl

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

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

Generate your access token

Request example:

```
POST /auth/token
```

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

(continues on next page)

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

Response example:

```
{
    "token_type":"Bearer",
    "expires_in":1800,
    "access_token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.
↪eyJpc3MiOiJwcG1zIiwiYXVkIjoiaHR0cHM6XC9cL3Rlc3RpbmcucG13aWsucHJvXC9zZXR5LCJzdWIiOiJkNmNkZGMxMS1iZDA
↪Nec2mYFRv6manzXjq0sHQxINZvu-fbDYT8AedVHBKYvu1F9hYKaFReY8rNgfsMANw2OX8-
↪IKpTrQb1DyRkG4nxpIEbob528_
↪lPd7roho5mtKlE8sfS9WZE1piYOwaNDySDEUwUowgj2xBiJqSODjxBI6qVhLkynGEEeNBVh-
↪lrUmlcjpYqUc3saHvX72L-rqbIHa_1dzGarR-dcPyns-RpKjZEILzUSYOHdM09KDti-xsG-
↪nbKHGdP8fVEEJPyupnAfJPOLHQg_j1c5IvJSvTKVF3j4_
↪zo6Zw5g8YkaheT9Iwph5BGHFRneXatcmbwKI8JzSDFi6CinzI-okYKRPbg"
}
```

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

Create an app

Request example:

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

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

Note, that you have to replace:

- `<domain>` with your PPAS instance URL,

- `<your_access_token>` with your generated access token

Response example:

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

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

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

        ],
        "excludedUrlParams":[

        ],
        "excludedUserAgents":[

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

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

Get an app

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

Request example:

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

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

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

Response example:

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

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

            ],
            "excludedUrlParams":[

            ],
            "excludedUserAgents":[

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

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

Update app

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

Request example:

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

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

This request changed app name from `AppName` to `NewAppName`.

Notice three things:

- `-X PATCH` before URL. It means that this request is available using `HTTP PATCH method`

- you have to specify also `data/id` - it's a JSON API requirement

- also `data/type` is required. For example, when you want to work with app resource, specify it's type as `ppms/app`

- you can set only parameters you want to update. For more apps attributes go to App edit reference

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

Delete an app

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

Request example:

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

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

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

API usage example with Postman

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

- Access control

- Apps

- Meta Sites

- Users

- User Groups

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

- replace your domain in url

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

7.1.3 FAQ

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

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

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

API returns `JWT not found`

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

API returns `Expired JWT Token`

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

API returns `access token not authorized`

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

7.2 Access Control API

7.3 Apps API

7.4 Audit log API

7.5 Meta Sites API

7.6 Modules API

7.7 Tracker Settings API

7.8 Users API

7.9 User Groups API

Integrations

8.1 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.

8.1.1 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 JS tracker. Check *how to get cookie hash* section for detailed information.

```html
<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>"
    }
  }
  </script>
</amp-analytics>
```

8.1.2 How to get tracker cookie hash

If there is no non-AMP page tracked by traditional JS tracker, 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 JS tracker. Each JS tracker generates unique cookie name based on its configuration. Follow these instructions to get hash from cookie generated by JS tracker:

- Setup JS tracker 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 JS tracker (in the example its value is `cdef`). Copy this part and replace `<TRACKER_HASH>` with it.

Here you can see how to look for tracker cookie in Google Chrome developer tools:
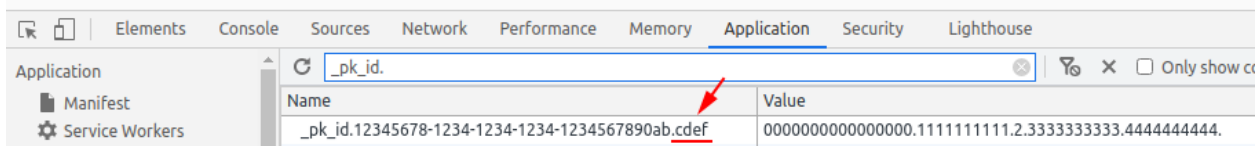
8.1.3 Tracking custom events

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",
          "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

8.1.4 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:

```html
<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

8.1.5 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:

```html
<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

8.1.6 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",
        "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

8.1.7 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

8.1.8 Complete page example

This example shows complete AMP page with 2 buttons. It will send page view, custom event and goal conversion.

```html
<!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",
                            "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>
```

8.2 Piwik PRO SDK for Android

8.2.1 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

Including the library

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

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

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

Configuration

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

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

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

```
    }
}
```

2. Manage the `Tracker` on your own. To configure the `Tracker` you will need a server address and website ID (you can find it in "Administration > Websites & apps > Installation"):

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

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

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

The application is ready to use Piwik PRO SDK.

8.2.2 Using Piwik PRO SDK

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

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

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

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

2. You manage the `Tracker` yourself

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

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

Tracking screen views

Requires Analytics

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

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

- A path (required) – each screen should be mapped to the URL path

- A title (optional) – the title of the action being tracked. It is possible to use slashes (/) to set one or several categories for this action.

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

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

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

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

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

Tracking custom events

Requires Analytics

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

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

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

- A category (required) – this String defines the event category. You may define event categories based on the class of user actions (e.g. clicks, gestures, voice commands), or you may define them based on the features available in your application (e.g. play, pause, fast forward, etc.).

- An action (required) – this String defines the specific event action within the category specified. In the example, we are effectively saying that the category of the event is user clicks, and the action is a button click.

- A name (optional) – this String defines a label associated with the event. For example, if you have multiple button controls on a screen, you may use the label to specify the specific view control identifier that was clicked.

- A value (optional) – this Float defines a numerical value associated with the event. For example, if you were tracking "Buy" button clicks, you may log the number of items being purchased or their total cost.

- A path (optional) – the path under which this event occurred.

For more resources, please visit:

- Custom Events Overview

- Ultimate guide to event tracking.

Tracking exceptions

Requires Analytics

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

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

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

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

Tracking social interactions

Requires Analytics

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

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

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

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

Tracking downloads and app installs

Requires Analytics

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

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

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

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

```java
        DownloadTracker downloadTracker = new DownloadTracker(getTracker());
        DownloadTracker.Extra extra = new DownloadTracker.Extra.Custom() {
            @Override
            public boolean isIntensiveWork() {
                return false;
            }

            @Nullable
            @Override
            public String buildExtraIdentifier() {
                return "Demo Android download";
            }
        };

        TrackHelper.track().download(downloadTracker).identifier(extra).force().
→version("1.0").with(getTracker());
```

- isIntensiveWork() - return true if this should be run async and on a separate thread.

- buildExtraIdentifier() - return a String that will be used as extra identifier or null.

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

Tracking outlinks

Requires Analytics

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

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

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

Tracking search operations

Requires Analytics

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

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

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

- A category (optional) – specify a search category.

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

Tracking content impressions and interactions

Requires Analytics

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

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

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

Tracking goals

Requires Analytics

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

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

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

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

Tracking ecommerce transactions

Requires Analytics

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

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

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

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

- `orderId` (required) – a unique String identifying the order
- `grandTotal` (required) – Total amount of the order, in cents
- `subTotal` (optional) – the subTotal (net price) for the order, in cents

- `tax` (optional) – the tax for the order, in cents
- `shipping` (optional) – the shipping for the order, in cents
- `discount` (optional) – the discount for the order, in cents
- `items` (optional) – the items included in the order, use the `EcommerceItems` class to instantiate items

Tracking campaigns

Requires Analytics

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

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

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

Tracking custom variables

Requires Analytics

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

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

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

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

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

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

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

Custom variable is defined by three parameters:

- An index (required) – a given custom variable name must always be stored in the same "index" per session. For example, if you choose to store the variable name = "Gender" in index = 1 and you record another custom variable in index = 1, then the "Gender" variable will be deleted and replaced with a new custom variable stored in index 1.

- A name (required) – this String defines the name of a specific Custom Variable such as "User type" (Limited to 200 characters).

- A value (required) – this String defines the value of a specific Custom Variable such as "Customer" (Limited to 200 characters).

Tracking custom dimensions

Requires Analytics

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

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

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

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

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

Tracking user profile attributes

Requires Audience Manager

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

For sending profile attributes use `audienceManagerSetProfileAttribute` method:

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

- A name (required) – defines the profile attribute name (non-null string).

- A value (required) – defines the profile attribute value (non null string).

- An `add` (chain method) – used to specify more attributes to the user within the same event.

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

- WEBSITE_ID - always sent,

- USER_ID - if it is set. *Read more* about the User ID,

- EMAIL - if it is set. *Read more* about the email,

- VISITOR_ID - always sent, ID of the mobile application user, generated by SDK

- DEVICE_ID - an Advertising ID that, by default, is fetched automatically when the tracker instance is created. To turn off automatic fetch, use the `setTrackDeviceId(boolean isTracked)` method:

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

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

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

Reading user profile attributes

Requires Audience Manager

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

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

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

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

- An attributes (output) – dictionary of key-value pairs, where each pair represents the attribute name (key) and value.

- An errorData (output) – in case of error, only this method will be called. The method passes the error string.

Checking audience membership

Requires Audience Manager

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

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

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

- An audienceId (required) – ID of the audience (Audience Manager -> Audiences tab)

- An OnCheckAudienceMembership (required) – callback to handle request result (call is asynchronous), has two methods `void onChecked(boolean isMember)` and `void onError(String errorData)`

- An isMember (output) – a boolean value that indicates if user belongs to audience with given ID

- An errorData (output) – in case of error, only this method will be called. The method passes the error string.

8.2.3 Advanced usage

User ID

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

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

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

User email address

Used only by Audience Manager

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

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

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

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

Visitor ID

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

---

manually:

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

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

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

Sessions

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

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

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

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

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

Dispatching

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

- setDispatchInterval(0) - incoming events will be dispatched immediately

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

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

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

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

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

  ...
}
```

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

- lua_zlib

- lua-nginx-module

- inflate.lua samples

Custom queries

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

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

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

Default custom variables

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

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

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

Local storage limits

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

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

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

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

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

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

- A limit (required) – size in bytes after which events are deleted, 0 = unlimited. By default, the limit is set to 4 * 1024 * 1024 bytes = 4 Mb.

Opt out

You can enable an app-level opt-out flag that will disable Piwik PRO tracking across the entire app. Note that this flag must be set each time the app starts up and will default to `false`. To set the app-level opt-out, use:

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

Dry run

The SDK provides a dryRun flag that, when set, prevents any data from being sent to Piwik. The dryRun flag should be set whenever you are testing or debugging an implementation and do not want test data to appear in your Piwik reports. To set the dry run flag, use:

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

- A dryRunTarget (required) – a data structure the data should be passed into `List<Packet>` type. Set it to null to disable dry run.

8.2.4 License

Piwik PRO Android SDK is released under the BSD-3 Clause license.

Copyright 2018 Piwik PRO team

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- Neither the name of Piwik team nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, IN-CIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSI-NESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CON-TRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAM-AGE.

8.3 Piwik PRO SDK for iOS

8.3.1 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. `'~> 1.0.0'`.

Then run `pod install`. In every file you wish to use the PiwikPROSDK, don't forget to import it.

Configuration

To configure the tracker you will need the URL address of your tracking server and website ID (you can find it in *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;
}
```

8.3.2 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.

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

Requires Analytics

To track custom name-value pairs assigned to your users or screen views, you can use custom variables. A custom variable can have a visit scope, which means that they are assigned to the whole visit of the user or action scope meaning that they are assigned only to the next tracked action such as screen view. You can find more information about custom variables here:

It is required for names and values to be encoded in UTF-8.

```
[[PiwikTracker sharedInstance] setCustomVariableForIndex:1 name:@"filter" value:@"lcd
→" scope:CustomVariableScopeAction];
```

- An index (required) – a given custom variable name must always be stored in the same "index" per session. For example, if you choose to store the variable name = "Gender" in index = 1 and you record another custom variable in index = 1, then the "Gender" variable will be deleted and replaced with new custom variable stored in index 1. Please note that some of the indexes are already reserved. See *Default custom variables* section for details.

- A name (required) – this String defines the name of a specific Custom Variable such as "User type". Limited to 200 characters.

- A value (required) – this String defines the value of a specific Custom Variable such as "Customer". Limited to 200 characters.

- A scope (required) – this String allows the specification of the tracking event type - "visit", "action", etc. The scope is the value from the enum `CustomVariableScope` and could be `CustomVariableScopeVisit` or `CustomVariableScopeAction`.

Tracking with custom dimensions

Requires Analytics

You can also use custom dimensions to track custom values as below. Custom dimensions can also can have a visit or action scope but first have to be defined on the server in your web panel. More details about custom dimensions can be found in the documentation:

```
[[PiwikTracker sharedInstance] setCustomDimensionForIndex:1 value:@"english"␣
→scope:CustomDimensionScopeVisit];
```

- An index (required) – a given custom dimension must always be stored in the same "index" per session, similar to custom variables. In example `1` is our dimension slot.

- A value (required) – this String defines the value of a specific custom dimension such as "English". Limited to 200 characters.

- A scope (required) – this String allows the specification of the tracking event type - "visit", "action", etc. Scope is the value from enum `CustomDimensionScope` and could be `CustomDimensionScopeVisit` or `CustomDimensionScopeAction`.

Tracking profile attributes

Requires Audience Manager

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

Audience Manager) and, afterwards started using a mobile application, the existing profile will be enriched only if the email was set. Otherwise, a new profile will be created.

For sending profile attributes use the `sendProfileAttributeWithName` method:

```
[[PiwikTracker sharedInstance] sendProfileAttributeWithName:@"food" value:@"chips"];
```

- A name (required) – defines profile attribute name (non-null string).

- A value (required) – defines profile attribute value (non-null string).

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

- WEBSITE_ID - always sent,

- USER_ID - if It is set. *Read more* about the User ID,

- EMAIL - if It is set. *Read more* about the email,

- VISITOR_ID - always sent, ID of the mobile application user, generated by SDK

- DEVICE_ID - it is a device IDFA, which is not set by default (due to platform limitations). In order to set device ID see *Device ID* section below.

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

Reading user profile attributes

Requires Audience Manager

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

```
[[PiwikTracker sharedInstance] audienceManagerGetProfileAttributes:^(NSDictionary
→*profileAttributes, NSError * _Nullable error) {
    // do something with attributes list
}];
```

- `completionBlock` (required) – callback to handle request result (call is asynchronous)

- `profileAttributes` (output) – dictionary of key-value pairs, where each pair represent attribute name (key) and value.

- `errorData` (output) – in case of error only, this method will be called. This method passes the error string.

Checking audience membership

Requires Audience Manager

Audience check allows one to check if the user belongs to a specific group of users defined in the audience manger panel based on analytics data and audience manager profile attributes. You can check if the user belongs to a given audience, for example, to display him/her some type of special offer like in the example below:

```
[[PiwikTracker sharedInstance] checkMembershipWithAudienceID:@"12345678-90ab-cdef-
→1234-567890abcdef" completionBlock:^(BOOL isMember, NSError * _Nullable error) {
    // do something if is member or handle error
}];
```

- `audienceId` (required) – ID of the audience (Audience Manager -> Audiences tab)

- `completionBlock` (required) – callback to handle request result (call is asynchronous)

- `isMember` (output) – a boolean value that indicates if the user belongs to an audience with a given ID

- `error` (output) – in case of error only, this method will be called. Method pass the error string.

8.3.3 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. 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;
```

8.3.4 License

Piwik PRO iOS SDK is available under the MIT license.

Copyright 2018 Piwik PRO team

All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Glossary

Application Website or application tracked by PPAS.

App ID PPAS *application* identificator (previously **website ID**, **site ID** or **idSite**).

User Visitor on tracked *application*.

Analytics ID ID assigned to *user* by *Analytics* for the duration of *Analytics* session. It is stored in browser cookie.

User ID Permanent ID assigned to *user* by *application* (e.g. username). You can read more about it here.

Device ID Device ID (device identification) is a distinctive number associated with a smartphone or similar handheld device. Device IDs are separate from hardware serial numbers.

Identifier Unique *user* ID (e.g. *analytics ID*, *user ID*, *device ID* or email).

Visit Period of continuous *user* activity on *application*. It ends in the following situations:

- after a period of inactivity (option set to 30 minutes by default)
- at midnight (option enabled by default)
- on campaign change (option enabled by default)
- when HTTP referrer points to different website (option disabled by default)

Audience Named set of *attribute* conditions used to define a group of *users* matching them.

Attribute Named value assigned to *user* profile.

Attribute whitelist List of *user attributes* that are publicly available via Audience Manager API.

> **Note:** It is still necessary to identify the *user* with his *analytics ID* to access this information.

PII Personally Identifiable Information.

Analytics attribute *Attribute* generated from value provided by *Analytics* (e.g. browser and device data, location data, etc.). You can read more about *attribute* sources here.

Note: If *custom attribute* uses the same name - it will be represented as a separate *attribute*.

Custom attribute *Attribute* generated from value provided by source other than *Analytics* (e.g. *Form Tracker*, sdk/index). You can read more about *attribute* sources here.

Warning: *Custom attribute* will store only latest value provided by any custom source.

Note: If *analytics attribute* uses the same name - it will be represented as a separate *attribute*.

Analytics PPAS component gathering statistics about each *user* of the *application* (previously **Piwik**).

Symbols

A

C

D

E

F

G

H

I

K

killFrame() (*built-in function*), 53

L

logAllContentBlocksOnPage() (*built-in function*), 48

O

onFulfilled() (*built-in function*), 62, 63, 66, 72–78
onRejected() (*built-in function*), 62–64, 66, 72–78

P

PII, **139**
ping() (*built-in function*), 53
Piwik.getAsyncTracker() (*Piwik method*), 43
Piwik.getTracker() (*Piwik method*), 43
ppms.am.api() (*ppms.am method*), 61
ppms.cm.api() (*ppms.cm method*), 72

R

redirectFile() (*built-in function*), 53
removeDownloadExtensions() (*built-in function*), 49
removeEcommerceItem() (*built-in function*), 45
resetUserId() (*built-in function*), 50

S

setApiUrl() (*built-in function*), 52
setCampaignKeywordKey() (*built-in function*), 53
setCampaignNameKey() (*built-in function*), 53
setCookieDomain() (*built-in function*), 51
setCookieNamePrefix() (*built-in function*), 51
setCookiePath() (*built-in function*), 51
setCrossDomainLinkingTimeout() (*built-in function*), 44
setCustomDimension() (*built-in function*), 46
setCustomDimensionValue() (*built-in function*), 46
setCustomRequestProcessing() (*built-in function*), 54
setCustomUrl() (*built-in function*), 52
setCustomVariable() (*built-in function*), 46
setDocumentTitle() (*built-in function*), 52
setDomains() (*built-in function*), 52
setDoNotTrack() (*built-in function*), 53
setDownloadClasses() (*built-in function*), 49
setDownloadExtensions() (*built-in function*), 49
setEcommerceView() (*built-in function*), 45
setGenerationTimeMs() (*built-in function*), 52
setIgnoreClasses() (*built-in function*), 50
setLinkClasses() (*built-in function*), 49
setLinkTrackingTimer() (*built-in function*), 50

setReferralCookieTimeout() (*built-in function*), 51
setReferrerUrl() (*built-in function*), 52
setRequestContentType() (*built-in function*), 54
setRequestMethod() (*built-in function*), 54
setSecureCookie() (*built-in function*), 51
setSessionCookieTimeout() (*built-in function*), 52
setUserId() (*built-in function*), 50
setUserIsAnonymous() (*built-in function*), 53
setVisitorCookieTimeout() (*built-in function*), 51
storeCustomVariablesInCookie() (*built-in function*), 46

T

trackAllContentImpressions() (*built-in function*), 47
trackContentImpression() (*built-in function*), 48
trackContentImpressionsWithinNode() (*built-in function*), 47
trackContentInteraction() (*built-in function*), 48
trackContentInteractionNode() (*built-in function*), 48
trackEcommerceCartUpdate() (*built-in function*), 45
trackEcommerceOrder() (*built-in function*), 45
trackEvent() (*built-in function*), 43
trackGoal() (*built-in function*), 43
trackingType (*None attribute*), 69
trackLink() (*built-in function*), 48
trackPageView() (*built-in function*), 43
trackSiteSearch() (*built-in function*), 44
trackVisibleContentImpressions() (*built-in function*), 47
type (*None attribute*), 68

U

urlDecorator() (*built-in function*), 54
urlParser() (*built-in function*), 55
useLabels (*None attribute*), 69
User, **139**
User ID, **139**
user_id (*None attribute*), 66

V

Visit, **139**