Integration and Configuration of the Android Tracking Library

Version 4.0

Webtrekk
Drive Your Profit.

Contents

1 Introduction

The Android tracking library allows you to record usage frequency and activities for your Android app in Webtrekk. This document describes how to technically integrate tracking into your Android app.

To ensure the Webtrekk analysis options are utilised optimally, please read our training material on the fundamentals of data collection during the design phase. If you do not have this document, contact your personal consultant or support@webtrekk.com at any time to obtain a copy.

The app calls and activities of the user are sent to the Webtrekk tracking system as requests through the use of the tracking API.

Depending on the user's online status, the requests are sent to Webtrekk correspondingly:

1. If the mobile device is online, the requests are sent in a definable time interval.
2. If the device is offline, the activities are buffered and sent as soon as the device is online again.

1.1 System requirements

Trackable systems: All devices with the operating system Android version 4.0 or higher.

Development environment: In principle, integration into the app is possible with any development environment. This documentation refers to Android Studio.

1.2 Integration of the tracking library

Before you start tracking for your app, the library has to be integrated into your Android project.

The steps for doing this in Android Studio with Gradle as the build system are described in the following. Here the paths refer to the Android default project layout.

1. If it does not exist already, create a folder "libs" in the main folder of the Android module you want to track.

2. Copy the file "WebtrekkSDK-release.aar" to the folder from step 1.

3. Add this folder as the local repository of the Android module in the Gradle build file by adding the following lines to the "build.gradle" file for the module:

```
repositories{
    flatDir{
        dirs 'libs'
    }
}
```

4. Add the .aar file as a dependency, also in the build.gradle file for the module.

```
dependencies {
    ...
    compile(name:'WebtrekkSDK-release', ext:'aar')
    ...
}
```

5. Adapt permissions:

The following two permissions are always needed for tracking, otherwise no requests can be transmitted.

```
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
```

If you want to track the username or other profile information from the Play Store such as the user's e-mail address, the following permissions also have to be added:

```
<uses-permission android:name="android.permission.READ_PROFILE" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
```

Element	Required for
READ_PROFILE	First name, last name
READ_CONTACTS	E-mail address
GET_ACCOUNTS	

6. Save configuration file

Copy the configuration file "webtrekk_config.xml" to the resources folder of your Android module under "**module/res/raw/**". This file contains the required configuration for sending tracking requests to your Webtrekk account (see Tracking configuration) and can also be used for defining parameters and values.

2 Tracking functionality

The tracking library lets you send page and action requests to Webtrekk. Just as in standard webpage pixilation, these requests can be enriched using further parameters.

The process of tracking an app visit is always the same.

- Initialising the Webtrekk instance
- Send as many tracking requests during the session as you wish

Two options are available for tracking activities:

Automatic tracking: Corresponding callbacks are registered in the Android app for the Android lifecycle methods. These are then called automatically and track all activity calls automatically.

Manual tracking: Here you have to implement the tracking yourself for every activity to be tracked.

Even when you activate automatic tracking, you are not forced to measure all activities of your application. You can also exclude specific activities from automatic tracking and vice versa using the options of the configuration file.

Both options require a configuration file in the XML format, see Tracking configuration.

> Webtrekk recommends activating automatic tracking and a complete configuration using the configuration file.

2.1 Tracking configuration

Use the configuration file “webtrekk_config.xml” to store all required configuration settings for measuring activities in your app.

The table that follows contains an overview of all configuration characteristics required for sending information, or that can be activated to collect additional data about the user:

Element	Description	Optional
version	Version number of the configuration file	☒
trackDomain	Your Webtrekk track domain	☒
trackId	Your Webtrekk trackId	☒
sampling	The sampling factor is used to only capture every nth user.	☑
sendDelay	Defines the interval for sending registered requests to Webtrekk.	☑
maxRequests	Specifies the maximum number of requests that are buffered on the device.	☑
autoTracked	If automatic tracking is activated, a request is generated automatically when an activity is started.	☑
autoTrackAppUpdate	Section 5.1	☑
autoTrackAdvertiserId		☑
autoTrackAppVersionName		☑
autoTrackAppVersionCode		☑
autoTrackAppPreInstalled		☑
autoTrackPlaystoreMail		☑
autoTrackPlaystoreGivenName		☑
autoTrackPlaystoreFamilyName		☑
autoTrackApiLevel		☑
autoTrackScreenOrientation		☑
autoTrackConnectionType		☑
autoTrackAdvertisementOptOut		☑
autoTrackRequestUrlStoreSize		☑
enableRemoteConfiguration	Section 9	☑
trackingConfigurationUrl		☑
resendOnStartEventTime	Defines the start time for including an interaction in a new session, for example when the user was previously called	☑

The following example of a configuration file contains all elements that are evaluated when the Webtrekk instance is initialised and the default values that are set if the elements in the configuration file were not overwritten:

```
<?xml version="1.0" encoding="utf-8"?>
<webtrekkConfiguration>
    <version>0</version>
    <trackDomain type="text"></trackDomain>
    <trackId type="text"></trackId>
    <sampling type="text">0</sampling>
    <sendDelay type="text">60</sendDelay>
    <maxRequests type="number">5000</maxRequests>

    <autoTracked>true</autoTracked>

    <autoTrackAppUpdate>true</autoTrackAppUpdate>
    <autoTrackAdvertiserId>true</autoTrackAdvertiserId>
    <autoTrackAppVersionName>true</autoTrackAppVersionName>
    <autoTrackAppVersionCode>true</autoTrackAppVersionCode>
    <autoTrackAppPreInstalled>true</autoTrackAppPreInstalled>
    <autoTrackPlaystoreMail>false</autoTrackPlaystoreMail>
    <autoTrackPlaystoreGivenName>false</autoTrackPlaystoreGivenName>
    <autoTrackPlaystoreFamilyName>false</autoTrackPlaystoreFamilyName>
    <autoTrackApiLevel>true</autoTrackApiLevel>
    <autoTrackScreenOrientation>true</autoTrackScreenOrientation>
    <autoTrackConnectionType>true</autoTrackConnectionType>
    <autoTrackAdvertisementOptOut>true</autoTrackAdvertisementOptOut>
    <autoTrackRequestUrlStoreSize>true</autoTrackRequestUrlStoreSize>

    <enableRemoteConfiguration>false</enableRemoteConfiguration>
    <trackingConfigurationUrl></trackingConfigurationUrl>
    <resendOnStartEventTime>30</resendOnStartEventTime>
</webtrekkConfiguration>
```

Note that the configuration file has to be available in your Android module under the path "module/**res/raw/webtrekk_config.xml**".

2.2 Initialising tracking

To use the tracking classes within a source code file, you have to import them.

```
import com.webtrekk.webtrekksdk.Webtrekk;
```

The Webtrekk tracking functionality is provided via the Webtrekk class. It is implemented as singleton and has to be initialised once by the user. This is best done in the onCreate method of your MainActivity.

```
private Webtrekk webtrekk;

@Override
protected void onCreate (Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        webtrekk = Webtrekk.getInstance();
        webtrekk.initWebtrekk(getApplication());
}
```

If you have already stored the configuration file from Section 2.1 and configured your settings, you now have the functionality of the SDK available. By activating the “autoTracked” element in your configuration file, you have already implemented complete activity tracking and are now sending initial requests to your Webtrekk account.

When moving the app in the background or when closing the app directly, all requests that have not yet been sent to Webtrekk will be transmitted (if an Internet connection is available).

2.3 Starting and stopping tracking

Unlike the earlier versions of the SDK, it is no longer necessary to also transfer the activity that is called. Starting and stopping activities is automatically executed via the SDK and independently generates a contentID based on the module path and activity name.

2.4 Page tracking

In the same way as on an online platform, app tracking assumes that the application is made up of single pages, the usage of which should be recorded. Page tracking therefore lets you send certain app content such as pages or e-commerce values. The content is evaluated as page impressions and displayed on the page analysis screen of the Webtrekk user interface.

If you have deactivated the “autoTracked” element in your configuration XML, you have to execute measuring of the activity manually in order to track the activities. This is best done with the “onStart()” method of your activity by calling the “track()” method from the Webtrekk SDK.

```
// manual tracking call
@Override
public void onStart(){
   super.onStart();
   webtrekk.track();
}
```

When “autoTracked” is activated, page tracking is carried out automatically and should not be implemented manually in the onStart method.

You can also transfer additional information by calling the “track()” method, or define global parameters (see Section 3) and specific parameters for individual activities (see Section 4) using the configuration file “webtrekk_config.xml”. In order to generate a “TrackingParameter” object in the following examples, you also have to import this class.

```
import com.webtrekk.webtrekksdk.TrackingParameter;
import com.webtrekk.webtrekksdk.TrackingParameter.Parameter;
```

Subsequently you can generate a variable of the “TrackingParameter” type. Then you can assign all files to be sent to that.

```
// Example for a tracking call with a 'TrackingParameter' Object

[...]

private Webtrekk webtrekk;
private TrackingParameter tp;

@Override
protected void onCreate(Bundle savedInstanceState) {

   // get Webtrekk instance before
   [...]

   tp = new TrackingParameter();
   // tp.add(...)
}

@Override
public void onStart() {
   super.onStart();
   webtrekk.track(tp);
}

[...]
```

2.4.1 Contentgroups

Pages can be summarised with content groups, forming groups in your app that are transferred directly via the tracking library.

> If you have questions about the configuration of content groups and/or their file type, please contact support@webtrekk.com or your personal consultant.

```
// Tracking request with content groups:

private Webtrekk webtrekk;
private TrackingParameter tp;

@Override
protected void onCreate(Bundle savedInstanceState) {

   // Get Webtrekk Instance before
   [...]

   tp = new TrackingParameter();
   tp.add(Parameter.PAGE_CAT, "1", "de");
   tp.add(Parameter.PAGE_CAT, "2", "home");
}

@Override
public void onStart() {
   super.onStart();
   webtrekk.track(tp);
}
```

Content group parameters are not permitted to exceed a length of 255 characters. All additional characters are deleted.

> Content groups are assigned one time to a page. If a page is tracked together with a content group, all subsequent page calls will also be assigned to this content group.
> When passing the content group with the tracking library, only the initial page call (contentID) is taken into consideration. Webtrekk therefore recommends implementing contentIDs and content groups while embedding the pixel.

2.4.2 Page parameters (custom parameters)

You can use “custom parameters” (depending on the package) to enrich analytical data with your own app-specific information and/or metrics.

Page parameters refer to single pages and are assigned to them directly. In contrast to content groups (see Section 2.4.1), the reference between the page and the page parameter does not have to be unique. This allows you to specify a page to be called, for instance by entering the variant or a numerical value.

```
// Tracking request with page parameter:

private Webtrekk webtrekk;
private TrackingParameter tp;

@Override
protected void onCreate(Bundle savedInstanceState) {

   // Get Webtrekk Instance before
   [...]

   tp.add(Parameter.PAGE, "1", "app");
   tp.add(Parameter.PAGE, "2", "android");
}

@Override
public void onStart() {
   super.onStart();
   webtrekk.track(tp);
}
```

2.4.3 Product tracking

The following parameters serve to measure products in detail. The products can be transmitted to Webtrekk when a product is viewed, placed in the shopping basket or when the basket is purchased. A list of tracked products appears in the Webtrekk tool under “E-Commerce > Products”.

> Do not use any thousands separator in the price details. Decimal places are separated using a point or comma. Order data are **not** processed with “action tracking”.

Product name

Saves products placed in the shopping basket. If several products appear in the shopping basket, they are each separated by a semicolon. This parameter must be entered if products are to be measured. All other parameters are optional for product tracking. A product name may not contain more than 110 characters.

```
tp.add(Parameter.PRODUCT, "name-1;name 2");
```

Product quantity (optional)

Contains the product quantity. If several products are transmitted, they are each separated by a semicolon. The standard value is “1”.

```
tp.add(Parameter.PRODUCT_COUNT, "3;1");
```

Product price (optional)

Contains the product price (“0” prices are allowed). If you pass a single product more than once (product quantity > 1), use the overall price and not the individual price. If several prices are transmitted, they are each separated by a semicolon. The standard value is “0”.

```
tp.add(Parameter.PRODUCT_COST, "10.99;2999.95");
```

Currency code (optional)

Contains the currency code of a product or order; the value must be passed to the Webtrekk pixel in line with the ISO standard. If multiple products are transmitted via a single page (e.g. on the order confirmation page if more than 1 product was purchased), only 1 currency will be applied to all products. This means that the value only needs to be set once.

```
tp.add(Parameter.CURRENCY, "EUR");
```

Note: The currency is only passed for the purpose of currency conversion. In other words, the currency will be converted to the one (if any) that is stored in the Webtrekk front-end (Configuration → System configuration: Account). Only one currency is ever displayed here.

Shopping basket status (optional)

Contains the shopping basket status. If a product is viewed (e.g. on a product’s detailed view), the status is “view”. This status should always be set if the product can be added to the shopping basket.

If the product is added to the shopping basket, the status is “add”. If the shopping basket is purchased, the status “conf” is transmitted. If no status is transmitted when tracking a product, the standard value “view,” i.e. product view, is assumed.

```
tp.add(Parameter.PRODUCT_STATUS, "view");
```

Voucher value (optional)

Contains the value of a voucher. Use this parameter if the customer places an order using a voucher. You can only pass this parameter for orders.

```
tp.add(Parameter.VOUCHER_VALUE, "35.99");
```

Product categories (optional)

Product categories allow the grouping of products. The relationship between product and product category must be unique. In other words, it is not possible to assign the product “Shoes” once to the “Ladies” product category and once again to the “Sale” product category. Such non-unique relationships can be mapped using e-commerce parameters (see Section 2.4.5).

A product category may not contain more than 110 characters.

> Product categories are only assigned one time for a product. If a product is tracked together with a category, all other products with the same name will also be assigned to this category. Therefore, if the product status “view” has to be invoked prior to the purchase of a product, it is sufficient if product categories are only passed on at that point.

```
tp.add(Parameter.PRODUCT_CAT, "1", "men");
tp.add(Parameter.PRODUCT_CAT, "2", "shoes");
```

2.4.4 Tracking orders

Webtrekk can also track orders. To do this, the order value is transmitted along with the order number. “0” values are permitted. The difference to product tracking is that, as with the total order value, information on the order is transmitted rather than information on the individual products. Besides the total of purchased products, the total order value may also contain such values as discounts and shipping and packaging costs.

> Do not use any thousands separator in the price details. Decimal places are separated using a point or comma.

The parameter “total order value” must be entered if total order values are to be tracked.

The parameter “order number” (optional) contains a unique ID, which can be assigned to the order. Use of this setting ensures that no orders are counted twice.

```
tp.add(Parameter.ORDER_NUMBER, "M-12345");
tp.add(Parameter.ORDER_TOTAL, "2996.93");
tp.add(Parameter.VOUCHER_VALUE, "35.99");
```

2.4.5 E-commerce parameters

You can use custom parameters (depending on the package) to enrich analytical data with your own website-specific information and/or metrics.

Parameter references

E-commerce parameters are used to transmit additional product information (e.g. size, colour). In the case of several products, the number of single parameter values must match the number of products. The values are separated by a semicolon.

E-commerce parameters can also be used to transmit information about an order, e.g. payment or shipping type. In these cases, order tracking must be used. It is enough to transmit this parameter once per order. It applies equally to all products in the shopping basket.

The reference (product or order) is selected when configuring the tracking library. If “individual value” is selected, the parameter refers to the order. If “multiple values” has been selected, the parameter can refer to the product or the order.

> Note: Since website targets in Webtrekk must always be entered as e-commerce parameters, it is also possible to transfer e-commerce parameters that are not linked to orders or products.

```
tp.add(Parameter.ECOM, "1", "Levis");
tp.add(Parameter.ECOM, "2", "30");
tp.add(Parameter.ECOM, "3", "32");
```

2.4.6 Campaigns

Campaign tracking is configured in the Webtrekk tool (Configuration > Campaign Configuration). Without this configuration, no campaign information such as campaign clicks will be collected. Visits to certain pages via an app or the entry of defined links can be tracked as campaign clicks.

You have the option of setting your own campaign ID in the configuration segment. A campaign ID consists of a media code name and its value, separated by "%3D".

Additional information can also be provided for campaigns using campaign parameters.

```
tp.add(Parameter.ADVERTISEMENT, "mc%3Dnewsletter_2016_1");
```

Campaign parameter

You can use "custom parameters" to enrich analytical data with your own app-specific information and/or metrics.

Campaign parameters always refer to an advertising medium (the smallest subunit of a campaign in Webtrekk).

```
tp.add(Parameter.AD, "1", "personalized");
```

Webtrekk provides the ability to track Google campaigns automatically, if they were used to download the app from the App Store.

The referrer parameter can be used to determine a user's path to a specific app in the Play Store.

The broadcast receiver of the library must be added to the application's manifest first of all. To do this, open the XML view of the manifest and insert the following code within the <application> tag:

```
<receiver
   android:name="com.webtrekk.webtrekksdk.ReferrerReceiver"
   android:exported="true">
   <intent-filter>
      <action android:name="com.android.vending.INSTALL_REFERRER" />
   </intent-filter>
</receiver>
```

Documentation explaining the configuration of links to the Play Store, together with all information on campaign tracking, can be found on the following website: https://developers.google.com/analytics/devguides/collection/android/v2/campaigns. Apart from "gclid", all of the parameters listed there are supported.

The campaign tracked by us is made up as follows:

- Media Code (wt_mc): utm_source.utm_medium.utm_content.utm_campaign
- Kampagnen Keyword (wt_kw): utm_term

```
https://play.google.com/store/apps/details?id=com.example.app
&referrer=utm_source%3Dgoogle
%26utm_medium%3Dcpc
%26utm_term%3Drunning%252Bshoes
%26utm_content%3DdisplayAd1
%26utm_campaign%3Dshoe%252Bcampaign
```

As in the example above, the campaign code has to be URL-encoded in the “referrer” parameter in the link to the Play Store.

2.4.7 Session parameters

Session parameters always refer to a session, in other words a visit. If the value for the parameter is transmitted during a visit several times, only the last value is evaluated.

For example, the status indicating whether a user was logged in during the visit is transmitted. By default, each visit at the start of a session parameter is indicated as “not logged in”. A login is transmitted by the same parameter and thus overwrites the first value.

> In contrast to a page parameter, it is not possible to evaluate the page that a session parameter was set on. In addition, a page parameter allows the evaluation of any value set during a visit.

```
tp.add(Parameter.SESSION, "1", "logged.in");
```

2.4.8 Internal search

Analyse the search terms entered in your app by visitors by including them in tracking. Enter the search term used in the configuration parameter dynamically.

```
tp.add(Parameter.INTERN_SEARCH, "searchterm");
```

2.4.9 Custom visitor IDs

To improve visitor identification, you can use custom visitor IDs instead of the Webtrekk Ever-ID.

To use custom visitor IDs, include a unique identifier from your system with the tracking library. If you do not use unique visitor IDs in your app, you can use the visitor's e-mail address as an alternative unique identifier. In this case, you should encrypt the address to comply with data protection requirements and ensure the e-mail address is unreadable (for instance with MD5 hash).

```
tp.add(Parameter.CUSTOMER_ID, "372d1a04d003eebc09e17330d5d3117c");
```

With the optional visitor categories, you can also categorise the visitors. These URM categories first have to be created in the tool. In the example below, the family status is assigned to the visitor.

```
tp.add(Parameter.USER_CAT, "1", "single");
```

2.5 Action tracking

Action tracking can be used to track specific events in the app, such as clicks on a button or the ticking of checkboxes.

```
TrackingParameter buttonParameter = new TrackingParameter();

buttonParameter
   .add(Parameter.ACTION_NAME, "orderButton")
   .add(Parameter.CURRENCY, "EUR")
   .add(Parameter.PRODUCT, "product-a")
   .add(Parameter.PRODUCT_COUNT, "1")
   .add(Parameter.PRODUCT_STATUS, "add");

webtrekk.track(buttonParameter);
```

Similar to page tracking, you can add further parameters such as the position or the colour of a button to each request.

```
tp.add(Parameter.ACTION, "1", "green ");
tp.add(Parameter.ACTION, "2", "top");
```

2.6 Media tracking

Webtrekk provides the ability to track media usage with your app. Tracking is performed similar to the examples shown so far.

To track a media file, first set up the corresponding tracking parameters where you can configure the file name, length, bandwidth and other details.

Then you can send a tracking request in the corresponding action methods such as Play, Pause, Stop and Search. The following example shows an extension of the Android VideoPlayer class which supplements the methods of the basic class with tracking.

```
public class TrackedVideoView extends VideoView {

private Webtrekk webtrekk;
TrackingParameter tp;

    public TrackedVideoView(Context context) {
        super(context);
        initMediaFile();
    }

    public TrackedVideoView(Context context, AttributeSet attrs) {
        super(context, attrs);
        initMediaFile();
    }

    public TrackedVideoView(Context context,
AttributeSet attrs, int defStyle) {

        super(context, attrs, defStyle);
        initMediaFile();
    }

    public void initMediaFile() {
        tp = new TrackingParameter();

        tp.add(Parameter.MEDIA_FILE,
                getResources().getResourceEntryName(R.raw.marv3));
        tp.add(Parameter.MEDIA_LENGTH,
                String.valueOf(getDuration()));
        tp.add(Parameter.MEDIA_POS,
                String.valueOf(getCurrentPosition()));
        tp.add(Parameter.MEDIA_CAT, "1", "mp3");
        tp.add(Parameter.MEDIA_CAT, "1", "example");
    }

    @Override
    public void pause() {
        super.pause();
        tp.add(Parameter.MEDIA_ACTION, "pause");
        tp.add(Parameter.MEDIA_POS,
                String.valueOf(getCurrentPosition()));
        webtrekk.track(tp);
    }

    @Override
    public void start() {
        super.start();

        tp.add(Parameter.MEDIA_ACTION, "start");
        tp.add(Parameter.MEDIA_POS,
                String.valueOf(getCurrentPosition()));

        webtrekk.track(tp);
    }

[...]
```

```
[...]

    @Override
    public void seekTo(int msec) {
        super.seekTo(msec);

        tp.add(Parameter.MEDIA_ACTION, "seek");
        tp.add(Parameter.MEDIA_POS, String.valueOf(msec));

        webtrekk.track(tp);
    }

    @Override
    public void stopPlayback() {
        super.stopPlayback();

        tp.add(Parameter.MEDIA_POS,
                String.valueOf(getCurrentPosition()));
        tp.add(Parameter.MEDIA_ACTION, "stop");

        webtrekk.track(tp);
    }

    public Webtrekk getWebtrekk() {
        return webtrekk;
    }

    public void setWebtrekk(Webtrekk webtrekk) {
        this.webtrekk = webtrekk;
    }
}
```

A media session is started as soon as the video can be seen by the user. This means the video is measured as soon as it is displayed (even if it is not played immediately).

Every media session is closed as soon as the action “stop” has been recorded. If further events are to be tracked after this, a new media session will need to be started.

2.6.1 Play

The Play event must be sent when the user starts watching the video or restarts after pressing pause.

```
MediaActivity.this.tp.add(Parameter.MEDIA_ACTION, "init");
MediaActivity.this.tp.add(Parameter.MEDIA_ACTION, "play");
```

2.6.2 Position

The Position event should be sent every 30 seconds while a video is being watched (played). This ensures video viewing can be measured accurately.

```
MediaActivity.this.tp.add(Parameter.MEDIA_ACTION, "pos");
```

2.6.3 Pause

The Pause event is sent when the user pauses the video.

```
MediaActivity.this.tp.add(Parameter.MEDIA_ACTION, "pause");
```

2.6.4 Seek

The Seek event has to be send one time when the user begins changing the position of the video. When the slide control of the media player is released again, another event has to be sent to end the spooling process. This second event depends on the current mode of the media player. During playback, “play” is sent as the second event. If the player is currently in pause mode, the end of the spooling process is confirmed by a “pause” event.

```
MediaActivity.this.tp.add(Parameter.MEDIA_ACTION, "seek");
```

2.6.5 Stop

The Stop event must be sent when the user stops playing the video, the end of the video is reached or the video is removed (for instance when the activity is closed or another one is started).

```
MediaActivity.this.tp.add(Parameter.MEDIA_ACTION, "stop");
```

The Stop event ends the media session. If you wish to measure further events (for instance if the video is watched again), a new media session must be started.

3 Global tracking parameters

Global tracking parameters are available across activities and are therefore privileged for sending constant values or desired information from the standard customer parameters for each individual activity. The GlobalTrackingParameter object is available to you in your app for this purpose. In this TrackingParameter instance, you can set global values which are sent with each request:

```
// 'apiLevel' is a keyword of custom tracking parameter
// each request contains the api level of the device
// in session parameter 6
webtrekk.getGlobalTrackingParameter().add(
   Parameter.SESSION,
   "6",
    webtrekk.getCustomParameter().get("apiLevel")
);
```

Alternatively you can map the same scenario in your configuration file. Note that global tracking parameters set in the source code are overwritten by those in your configuration file.

```
<globalTrackingParameter>
   <sessionParameter>
      <parameter id="6" key="apiLevel" value=""></parameter>
   </sessionParameter>
</globalTrackingParameter>
```

You can set any parameter available to you in the TrackingParameter object as a global parameter and therefore send it with each request.

> Requests with action names are not affected by this. Further information about action tracking is found in the section Action tracking.

You will find an overview of the configuration of global tracking parameters for your configuration file below:

```
<globalTrackingParameter>

   <!-- define global tracking parameter which are send with every
request, the key has to match a valid parameter name!
   entries made here are available as default parameters in the
trackingparameter instance -->
   <parameter id="PRODUCT">product-a</parameter>
   <parameter id="PRODUCT_COST">0.99</parameter>

   <!-- define the global page parameter, the key is the index -->
   <pageParameter>
      <parameter id="1">pageparam1</parameter>
      <parameter id="2">pageparam2</parameter>
      <parameter id="3">pageparam3</parameter>
   </pageParameter>

   <sessionParameter>
      <parameter id="1">sessionparam1</parameter>
   </sessionParameter>

   <ecomParameter>
      <parameter id="1">ecomparam1</parameter>
   </ecomParameter>

   <userCategories>
      <parameter id="1">usercategory1</parameter>
   </userCategories>

   <pageCategories>
      <parameter id="1">pagecategory1</parameter>
   </pageCategories>

   <adParameter>
      <parameter id="1">adparam1</parameter>
   </adParameter>

   <actionParameter>
      <parameter id="1">actionparam1</parameter>
   </actionParameter>

   <productCategories>
      <parameter id="1">productcategory1</parameter>
   </productCategories>

   <mediaCategories>
      <parameter id="1">mediacategory1</parameter>
   </mediaCategories>

</globalTrackingParameter>
```

4 Activity parameters

Unlike the global tracking parameters, the activity parameters are only available to you within the scope of the activity. All parameters assigned to a TrackingParameter object within an activity in the source code of your app are automatically activity parameters.

Since activity parameters are more closely related to the activity, they overwrite the configuration for global tracking parameters. This is the case for global tracking parameters set in the source code and in the configuration file. As with the global tracking parameters, configurations defined in the source code for each activity parameter are overwritten by the configuration in the configuration file.

Activity parameters are set in the source code, as shown in the examples in Section 2.4. Another brief example follows:

```
// Example for a tracking call with an object TrackingParameter

private Webtrekk webtrekk;
private TrackingParameter tp;

[...]

@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);

   webtrekk = Webtrekk.getInstance();

   tp = new TrackingParameter();
   tp.add(Parameter.PAGE_CAT, "1", "de");
   tp.add(Parameter.PAGE_CAT, "2", "home");
}

@Override
public void onStart() {
   super.onStart();
   // no track call, because autoTracked is enabled
   // webtrekk.track(tp);
}

[...]
```

Alternatively you can model the same scenario in your configuration file:

```
<activity>
   <classname type="text">myapp.test.MainActivity</classname>
   <mappingname type="text">Landingpage</mappingname>
   <autoTracked>true</autoTracked>

   <!--activity tracking parameter -->
   <activityTrackingParameter>
      <pageCategories>
         <parameter id="1">de</parameter>
         <parameter id="2">home</parameter>
      </pageCategories>
   </activityTrackingParameter>
</activity>
```

Requests with action names are not affected by this. Further information about action tracking is found in the section Action tracking.

An “<activity></activity>” element is defined in the configuration file for each activity that will be adapted/extended using the configuration file. There are additional setting options for this element using the following configuration elements:

- **<classname></classname>**: Contains the activity name for which the following settings are valid
- **<mappingname></mappingname>**: With this element you can specify and alternative content ID for this activity
- **<autoTracked></autoTracked>**: Activates/deactivates automatic tracking separately for this activity

You will find an overview of the configuration of global tracking parameters for your configuration file below:

```
<activity>
   <classname type="text">myapp.test.MainActivity</classname>
   <mappingname type="text">Landingpage</mappingname>
   <autoTracked>true</autoTracked>

   <!--activity tracking parameter -->
   <activityTrackingParameter>
      <pageParameter>
         <parameter id="1">pageParam1</parameter>
         <parameter id="2">pageParam2</parameter>
         <parameter id="3">pageParam3</parameter>
      </pageParameter>
      <sessionParameter>
         <parameter id="1">sessionParam1</parameter>
      </sessionParameter>
      <ecomParameter>
         <parameter id="1">ecomParam1</parameter>
      </ecomParameter>
      <userCategories>
         <parameter id="1">userCat1</parameter>
      </userCategories>
      <pageCategories>
         <parameter id="1">pageCat1</parameter>
      </pageCategories>
      <adParameter>
         <parameter id="1">adParam1</parameter>
      </adParameter>
      <actionParameter>
         <parameter id="1">actionParam1</parameter>
      </actionParameter>
      <productCategories>
         <parameter id="1">productCat1</parameter>
      </productCategories>
      <mediaCategories>
         <parameter id="1">mediaCat1</parameter>
      </mediaCategories>

   </activityTrackingParameter>
</activity>
```

5 Custom parameters/placeholders

Custom parameters give you access to all properties activated in the configuration XML; see Tracking configuration. You can also write user-defined keys to the custom parameters and use their values in freely selectable parameters within the scope of the activity.

```
// set key 'example_product_name'with value to custom parameter
webtrekk.getCustomParameter().put("my_product_name", "Product A");

// set the value of custom parameter 'example_product_name' to the
// Webtrekk Product name
tp.add(
   TrackingParameter.Parameter.PRODUCT,
   webtrekk.getCustomParameter().get("my_product_name")
);
```

If you want to assign defined custom parameters to a tracking parameter through your configuration file, use the "key" attribute of every element to send the value of the custom parameter with the parameter being tracked:

```
// set the value of custom parameter 'example_product_name' to the
// Webtrekk Product name by using the 'key' attribute
<parameter id="PRODUCT" key="my_product_name"></parameter>
```

Make sure that custom parameters are initialised before they are used.

5.1 Standard custom parameters

Webtrekk offers the possibility of reading various predefined properties from the tracking library and sending them in a freely selectable parameter. An overview of the standard parameters follows:

Element (key)	Description	Values
appPreinstalled	Contains a value specifying whether the app was already preinstalled on the device.	true || false
appVersion	Contains the Android app version.	Example: 2.1
appUpdated	Contains a value specifying whether the app has been updated.	0 || 1
apiLevel	Contains the Android API level.	Example: 19
appVersionCode	Whenever you publish a new version of your app in the Play Store, it is assigned a unique integer for identification. This is incremental and makes it possible to identify updates.	Example: 3
screenOrientation	Contains the current orientation of the device.	portrait || landscape
connectionType	Specifies the current connection type of the device	3G || WiFi || Offline
requestUrlStoreSize	Contains the current number of buffered requests and thereby makes it possible for you to determine whether the value stored in the configuration file is optimised for the URL store, or whether it is for example configured too small	Example: 173
playstoreMail	Contains the e-mail address associated with the first Google account for the device.	
playstoreGivenname	Contains the first name of the user of the first Google account for the device.	
playstoreFamilyname	Contains the last name of the user of the first Google account for the device.	
advertiserId	The advertiser ID is a unique user ID assigned by Google to each user via the Play Store. It allows the user to be identified even across various devices, as long as the user logs on with the same Google account.	
advertisingOptOut	Makes it possible for the app user to not see any advertising.	

6 Action tracking

Action tracking allows you to also track interactions within an activity. For example, you can track the clicks on a button by transferring an action request when the desired button state occurs.

What is an action request? Every request for which an “ACTION_NAME” is transferred so that the “ct-Parameter” is populated in the tracking request is an action request.

Actions can only be tracked manually and therefore has to be implemented in the source code for your app. Unlike normal data capture, only the data transferred as parameters to the “track” method can be captured with action tracking. This means that no global tracking parameters are sent with an action request unless they are explicitly passed to the “track” method.

```
// track an action by adding an 'ACTION_NAME'
// only the given parameters will set to the tracking request -
// no global or activity parameters
public void onAddToBasketClicked(View view) {
TrackingParameter buttonParameter = new TrackingParameter();

   buttonParameter
      .add(Parameter.ACTION_NAME, "addToBasket")
      .add(Parameter.CURRENCY, "EUR")
      .add(Parameter.PRODUCT, "product-a")
      .add(Parameter.PRODUCT_COUNT, "1")
      .add(Parameter.PRODUCT_STATUS, "add");

   webtrekk.track(buttonParameter);
}
```

7 Tracking parameter object

Through the instance of a TrackingParameter object, you can set custom parameters and also pass standard parameters in a clear and uniform manner:

Key			URL parameter
ACTION_NAME	Action name		ct
VOUCHER_VALUE	Voucher value		cb563
ORDER_TOTAL	Order value		ov
ORDER_NUMBER	Order number		oi
PRODUCT	Product name(s)		ba
PRODUCT_COST	Production costs		co
CURRENCY	Currency		cr
PRODUCT_COUNT	Product count		qn
PRODUCT_STATUS	Product status		st
CUSTOMER_ID	Customer ID		cd
EMAIL	User e-mail		uc700
EMAIL_RID			uc701
NEWSLETTER			uc702
GNAME	First name of the user		uc703
SNAME	Last name of the user		uc704
PHONE	Telephone		uc705
GENDER	Gender		uc705
BIRTHDAY	Date of birth		uc707
CITY	City		uc708
COUNTRY	Country		uc709
ZIP	Postal/zip code		uc710
STREET	Street		uc711
STREETNUMBER	House number		uc712
INTERN_SEARCH	Search term		is
ACTION	Action parameters		e.g. ck1
AD	Campaign parameters		e.g. cc1
ECOM	E-commerce parameters		e.g. cb1
PAGE	Page parameters		e.g. cp1
SESSION	Session parameters		e.g. cs1
MEDIA_CAT	Media categories		e.g. mg1
PAGE_CAT	Page categories		e.g. cg1
PRODUCT_CAT	Product categories		e.g. ca1
USER_CAT	User categories		e.g. uc1

8 Webtrekk Ever-ID

The Webtrekk Ever-ID is used to identify users. On standard websites this is set and transmitted automatically.

Since the technical possibilities for identifying users of mobile devices are limited, you have the possibility to retrieve and send this ID manually whenever the user switches from the app to a mobile browser, for example. This enables the user to retain the Ever-ID provided by the app during the website visit, thereby enabling end-to-end tracking of the user session.

You can use the method described below to read the Ever-ID.

```
String eid = webtrekk.getEverId();
```

Transmission to the browser is then realised by appending two parameters to the target link (URL).

- “wt_eid”: The Ever-ID
- “wt_t”: Current Unix timestamp in milliseconds. This is needed in order to ensure the Ever-ID is valid for no longer than 15 minutes in the URL. This minimises the probability of such a link being posted and, for example, evaluated multiple times with the same ID (otherwise, all of the visitors would receive the same Ever-ID and therefore be counted as one visitor!).

```
// Example for an URL with Ever-ID and timestamp:
http://new.domain.com/page?wt_eid=2135817235100536325&wt_t=135841437
8580
```

When the link is clicked, the web browser opens and the user is taken to the new page. There the user is assigned the same Ever-ID it also had in the app.

In addition to this, you can also provide your own referrer in order to ensure the visit to your website is reported as a referral and not as direct access. To simulate your own referrer, simply add the additional URL parameter “wt_ref” to the link URL. The parameter “wt_t”, which is valid for a period of 15 minutes, is also used here again.

```
// Example URL with Ever-ID, timestamp and referrer:
http://new.domain.com/start.html?wt_eid=2135817235100536325&wt_ref=h
ttp%3A%2F%2Fwww.webtrekk.com%2Fen%2Fhome.html&wt_t=1358414378580
```

Please note that the referrer URL must be encoded.

Note: If this feature is not used correctly, it can happen that multiple users are assigned the same Ever-ID. This would have a significant impact on the quality of your data. Therefore please ensure you use this feature correctly!

Pixel version 3.2.3 or higher is needed on the website in order to use this feature.

9 Opt-out functionality

Pursuant to article 15 of the Telemedia Act (TMG), website visitors can opt out of having their data stored anonymously so they are no longer recorded in the future. The “setOptedOut” method has to be used to enable this objection.

```
// exclude user from tracking
webtrekk.setOptout(true);
```

By invoking “optedOut”, you can check whether the user is willing to have their visit recorded or not.

```
// get information if user has optout status
boolean optOut = webtrekk.isOptedOut();
```

10 External tracking configuration

In order to avoid having to upload a complete release to the Google Play Store in case of an incorrect tracking configuration or if it is amended, you can store a path in the configuration XML. The application attempts to load a new configuration script from this path every time it starts. This reloaded file fully overwrites the previous configuration.

To activate reloading an external configuration file, you merely have to configure the following 2 elements in your configuration file:

```
<enableRemoteConfiguration>true</enableRemoteConfiguration>
<trackingConfigurationUrl>https://app.domain.tld/tracking_config.xml
</trackingConfigurationUrl>
```

If you want to reload an external configuration file, it has to be valid which means that all mandatory configurations listed in Section 2.1 also have to be stored in the external configuration file. Otherwise the reloaded configuration file is discarded and the existing configuration file is used.

> In addition, the version number in the configuration file being reloaded must have a higher value then the current configuration file. Otherwise the file cannot be interpreted as an update.

Please note the following configuration points for reloading an external configuration file:

1. "enableRemoteConfiguration" is activated in the current configuration.
2. The file stored under "trackingConfigurationUrl" in the current configuration is available; please also note the protocol you are using.
3. The new configuration file has correct information under:
    a. TrackID
    b. Trackdomain
    c. "sampling", "sendDelay", "maxRequests", "autoTracked"
    d. "enableRemoteConfiguration"
    e. "trackingConfigurationUrl"
4. Increment version number in a new configuration file

> If your application is already available in the Play Store, it is not possible to activate reloading an external configuration file without an update of your application in the Play Store.

11 Contact

Please do not hesitate to contact us if you have setup-related questions.
Webtrekk offers various support and consulting packages for priority support and comprehensive advice. Talk to us – we are happy to prepare an individual offer for you.

Webtrekk GmbH
Robert-Koch-Platz 4
10115 Berlin
Germany

Telephone 030 - 755 415 - 0
Fax 030 - 755 415 - 100
support@webtrekk.com

www.webtrekk.com