PRIVACY AT THE EDGE

Flutter App Developer Guide

Platforms: Android, iOS

SDK Platforms: Android, Android TV & FireOS
OS versions: Android 5.0+
Programming Languages : Java

SDK Platforms: iPhone & iPad
OS versions: iOS 9.0+ Applications
Programming Languages : Objective C + Swift

SDK version: 1.0.0-b0

SDK Download Link: https://pub.dev/packages/blotoutfluttersdk

INTRODUCTION

This document is created to help developers integrate Blotout's SDK easily with their android Applications.

Blotout's SDK offers companies all of the analytics and remarketing tools they are accustomed to, while offering best-in-class privacy preservation for the company's users. Blotout's SDK is out of the box compliant with GDPR, CCPA & COPPA. Blotout's SDK uses on-device, distributed edge computing for Analytics, Messaging and Remarketing, all without using User Personal Data, Device IDs or IP Addresses.

Pre-Requisites

- IDE: Android Studio 3.5+, Xcode
- SDK: Android SDK 21+ , iOS 8.0+
- In case Blotout's SDK is already enabled, please refer to the release notes & upgrade guide to perform upgrades based on the SDK's version's compatibility.

Setup and download

To download the android SDK, please create your account at https://<1P Container Domain>/signup or login at https://<1P Container Domain>/login. Register a new android App with the on-screen details or go directly to the "My Applications" section by clicking on your profile icon and then selecting "My Applications". Once you land on the "My Applications" page, check already registered Apps or register a new App based on your requirements. All registered Apps will be listed on this page with the Test Key and Prod Key. Based on your App environment, set the key and toggle the switch between test and production as needed.

- Copy your API Key & replace "XXXXXXXXXXXXXXX" with your stage or production key.
    a. Stage & Production key can be retrieved by login at https://<1P Container Domain>/applications

Once all the above setup is done, click on the link to download the latest version of the SDK and prepare for installation. It will download a zip file, extract the contents which will provide the following content.

1. Release Notes and Upgrade Guide
2. Sample App
3. SDK Integration Document
4. SDK Library

a. iOS Library and Helper Classes
b. Android Library and Helper Classes
c. Blotout Analytics DART package

Create a Flutter Application (**visit official link for more information**)

```
Flutter create BlotoutFlutterSample  // Replace "BlotoutFlutterSample" with you app name
```

Add Blotout's Analytics Dart Package to Flutter project

```
dependencies:
  flutter:
    sdk: flutter
  # The following adds the Cupertino Icons font to your application.
  blotoutfluttersdk: ^0.0.10
```

Please check the latest version of Blotoutfluttersdk at:
https://pub.dev/packages/blotoutfluttersdk

Setup API key and configuration

To set up API keys for test and production mode, as described in the "Setup and download" section, please get the test and production mode keys from the Blotout Dashboard.

Import BlotoutFlutterSDK Dart Package and use the below code to enable Keys and configure it.

```
import 'package:blotoutfluttersdk/blotoutfluttersdk.dart';

BlotoutAnalyticsAPI.blotoutAnalyticsInstance.initBlotoutSDK("BlotoutSDKKey","EndPointUrl");
```

How to use BlotoutAnalytics Dart Package to send data to Server, refer sample code below:

```
void getInstance() {
  BlotoutAnalyticsAPI.blotoutAnalyticsInstance
      .initBlotoutSDK(_blotoutToken, _endPointUrl)
      .then((success) {
    setState(() {
      _resultMessage = 'Instance created with success!';
    });
  });
}

 void trackEvent() {
  Map<String, String> properties = {
    "Button Pressed": "A LogEvent button was pressed"
  };
  BlotoutAnalyticsAPI.blotoutAnalyticsInstance
      .logEvent('logEvent', properties);
  setState(() {
    _resultMessage = 'Event sent with success!';
  });
}

 void trackEventWithTime() {
  Map<String, String> properties = {
    "logEventWithTime": "A logEventWithTime button was pressed"
  };
  BlotoutAnalyticsAPI.blotoutAnalyticsInstance
      .logEventWithTime('logEventWithTime', properties, DateTime.now());
  setState(() {
    _resultMessage = 'Event sent with success!';
  });
}

 void trackStartEvent() {
  Map<String, String> properties = {
    "trackStartEvent": "A trackStartEvent button was pressed"
  };
  BlotoutAnalyticsAPI.blotoutAnalyticsInstance
      .startTimedEvent('logTimeEvent', properties);
  setState(() {
    _resultMessage = 'Event sent with success!';
  });
}

 void trackEndEvent() {
  Map<String, String> properties = {
    "trackEndEvent": "A trackEndEvent button was pressed"
  };
  BlotoutAnalyticsAPI.blotoutAnalyticsInstance
      .endTimedEvent('logTimeEvent', properties);
  setState(() {
    _resultMessage = 'Event sent with success!';
  });
```

```
}

void trackPHIEvent() {
  Map<String, String> properties = {
    "Button Pressed": "A logPHIEvent button was pressed"
  };
  BlotoutAnalyticsAPI.blotoutAnalyticsInstance
      .logPHIEvent('logPHIEvent', properties, DateTime.now());
  setState(() {
    _resultMessage = 'Event sent with success!';
  });
}

void trackPIIEvent() {
  Map<String, String> properties = {
    "Button Pressed": "A logPIIEvent button was pressed"
  };
  BlotoutAnalyticsAPI.blotoutAnalyticsInstance
      .logPIIEvent('logPIIEvent', properties, DateTime.now());
  setState(() {
    _resultMessage = 'Event sent with success!';
  });
}

void isDeviceCompromised() {
  BlotoutAnalyticsAPI.blotoutAnalyticsInstance
      .isDeviceCompromised()
      .then((success) {
    setState(() {
      if (success) {
        _resultMessage = 'Device Compromised!';
      } else {
        _resultMessage = 'Device is not compromised!';
      }
    });
  });
}
```

Additional Configuration for Native iOS and Android project

Add the required Gradle Dependency and libraries

Using Android Studio 3.5+ will automatically add the required library or frameworks into your project based on uses and dependencies. Please feel free to verify from the list in case manual operation is needed.

1. Gradle Library

```
implementation fileTree(dir: 'libs', include: ['*.jar','*.aar'])
implementation ('com.birbit:android-priority-jobqueue:2.0.1')
implementation ('com.squareup.retrofit2:retrofit:2.6.2')
implementation ('org.greenrobot:eventbus:3.0.0')
implementation group: 'com.fasterxml.jackson.core', name: 'jackson-databind', version: '2.9.8'
implementation 'com.squareup.retrofit2:converter-gson:2.3.0'
implementation group: 'com.squareup.retrofit2', name: 'converter-jackson', version: '2.6.2'
implementation 'com.squareup.okhttp3:logging-interceptor:3.10.0'
implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:1.3.72"
implementation 'com.android.installreferrer:installreferrer:2.1'
```

2. Android Manifest

```
<application>
 <receiver android:name="com.blotout.events.BODayChangedBroadcastReceiver">
      <intent-filter>
        <action android:name="android.intent.action.DATE_CHANGED"/>
        <action android:name="android.intent.action.TIMEZONE_CHANGED"/>
        <action android:name="android.intent.action.TIME_CHANGED"/>
      </intent-filter>
   </receiver>
 <receiver
         android:name="com.blotout.referrerapi.BOReferrerReceiver"
         android:enabled="true"
         android:exported="true">
         <intent-filter>
            <action android:name="com.android.vending.INSTALL_REFERRER" />
         </intent-filter>
  </receiver>
   <service android:name="com.blotout.analytics.BOClosingService" android:stopWithTask="false">
   </service>
</application>

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

Configure Logging

Blotout's SDK provides logging mechanisms. When turned on using the method mentioned here, logs related to debug information will be available in the developer console.

```
void setSDKLogEnabled(bool sdkLogEnabled)
```

EVENTs CONFIGURATION

Blotout's SDK provides various categories of events to be captured and processed. To understand in a very simplistic manner, all the data which the Blotout SDK deals with, is some kind of event. Based on the type, the processing and storage of the event may differ. Blotout's SDK treats User Information as PII data and never shares any information related to user details with any server as clean text, it's always encrypted and data governance rules are applied. So any user related data or event, which might contain user PII (Personally Identifiable Information) is treated as a non-server sync event in privacy mode and sent to the server in 1P mode with encryption and data governance turned on.

Funnel Events

Funnel Events are a specific kind of event, which carry information related to event sequence verification. In general cases it is used as navigation verification from one part of the application or webpage to another part of application or webpage, but in practice it's not restricted to that and can be used for any sequence verification.

Example:

1: Navigation Funnel Event: Home Page → Product Category Page → Product Details Page → Purchase Page → Payment Page → Order Confirmation Page

2: In the Home Page of the application, the user can perform events from 1 to 100 in any order and I want to verify the sequence as: Event 5 → Event 12 → Event 17 → Event

There can be various other forms in practice. By default, Funnel Events are enabled and in case you want to disable the funnel events or switch the state, use the below property to do the action.

Find more details on Blotout dashboard about how to create Funnel Events and test results.

```
void setFunnelEventsEnabled(bool funnelEventsEnabled);
```

Segment Events

Segment events are another type of events which are generally used to categorise users and create different sets of user buckets.

Example:

1: Age 30+ High Income Group: This segment tells us about users, who are above 30 years of age and in a high income group as per global or country norms.

2: Age 30+ iPhone 11: This segment tells us about users, who are above 30 years of age and use an iPhone 11 as an active device.

There can be n number of segments and are generally created dynamically based on need using Blotout dashboard.

Segment Events are enabled by default, but can be toggled using the below property.

Find more details on Blotout's dashboard about how to create Segment Events and test results.

```
void setSegmentEventsEnabled(bool segmentEventsEnabled);
```

Non-Timed Events

Non-Timed events are generally events which are not time bound and do not contain duration information. For example, the Home Page loaded is non-timed but Home page loading started and home page loading ended, when grouped together, can be a timed event.

These events are categorized under two main categories in Blotout's SDK

1: SystemEvents:

System events are those which the Blotout SDK captures automatically like App Launch, App Terminated etc. These kinds of non-timed events do not require any developer intervention except enable or disable.

2: Developer Events:

Developer Events are those which developers codify in the Application code with the help of Blotout's SDK and SDK sync with Blotout's server, like "iPhone added to cart".

```
void logEvent(String eventName,  Map<String, dynamic> eventInfo);

void logEvent(String eventName, Map<String, dynamic> eventInfo, DateTime eventTime);
```

There are two methods mentioned above:

1st method logs the developer codified non-timed event considering current time as event occurrence time. The 2nd method also logs the developer codified non-timed event considering event time as passed under the happenedAt parameter. In the case that the developer passes happenedAt param as nil or null, then both methods behave similarly.

Timed Events

Timed events are generally events which are time bound and contain duration information as explained above. Timed events w.r.t Blotout's SDK are developer codified events only.

When developers want to log an event along with duration, then they can use the below mentioned APIs to log a timed event.

```
void startTimedEvent(String eventName, Map<String, dynamic> startEventInfo);

void endTimedEvent(String eventName, Map<String, dynamic> endEventInfo);
```

There are two methods mentioned above:

- startTimedEvent : This method will start the timer for the event name mentioned in the method call.
- endTimedEvent : This method will end the timer for the event name mentioned in the method call.

PII & PHI Events

PII (Personal Identifiable Information) events are like developer codified events that carry sensitive information related to User.

PHI (Protected Health information) events are like PII but carries user's private health information

In Blotout managed or deployed Infrastructure, PII and PHI events data is encrypted using asymmetric encryption algorithms and provides access to authenticated users only.

Below methods can be used to log PII and PHI information.

```
void logPIIEvent(String eventName,  Map<String, dynamic> eventInfo, DateTime eventTime);

void logPHIEvent(String eventName, Map<String, dynamic> eventInfo, DateTime eventTime);
```

Location Events

Blotout's SDK doesn't provide any specific event for location, rather the Blotout SDK associates events with location meta information for user analytics.

To turn ON the sensor

To turn ON Blotout's SDK or for turning on a specific sensor for a specific task, use the below mentioned APIs. When the Blotout SDK is integrated, then by default all the sensors are enabled.

```
// Default Value is YES, only set to NO when you want to disable SDK
// Once you disable SDK, SDK won't collect any further information but already collected information,
// will be sent to server as per Blotout Contract
void setEnabled(bool enabled);

//Individual Module enable or disable control
//System Events, which SDK detect automatically
void setSystemEventsEnabled(bool systemEventsEnabled);

//Retention Events, which SDK detect for retention tracking like DAU, MAU
void setRetentionEventsEnabled(bool retentionEventsEnabled);

//Funnel Events, which SDK process for funnel analysis
public void setFunnelEventsEnabled(bool funnelEventsEnabled);

//Segments Events, which SDK process for segment analysis
void setSegmentEventsEnabled(bool segmentEventsEnabled);

//Developer Codified Events, which SDK collects when developer send some events
void setDeveloperEventsEnabled(bool developerEventsEnabled);
```

To turn OFF the sensor

To turn OFF Blotout's SDK or for turning off a specific sensor for a specific task, use the below mentioned APIs. When the Blotout SDK is integrated then by default all the sensors are enabled.

```
// Default Value is YES, only set to NO when you want to disable SDK
// Once you disable SDK, SDK won't collect any further information but already collected information,
// will be sent to server as per Blotout Contract
void setEnabled(bool enabled);

//Individual Module enable or disable control
//System Events, which SDK detect automatically
void setSystemEventsEnabled(bool systemEventsEnabled);

//Retention Events, which SDK detect for retention tracking like DAU, MAU
void setRetentionEventsEnabled(bool retentionEventsEnabled);

//Funnel Events, which SDK process for funnel analysis
void setFunnelEventsEnabled(bool funnelEventsEnabled);

//Segments Events, which SDK process for segment analysis
void setSegmentEventsEnabled(bool segmentEventsEnabled);

//Developer Codified Events, which SDK collects when developer send some events
void setDeveloperEventsEnabled(bool developerEventsEnabled);
```

REPUTATION AS A SERVICE (RaaS)

Device & Environment Reputation

Blotout's SDK provides an initial version of Reputation as a Service along with Blotout's Analytics SDK, which provides some very helpful information about the devices which are sending events back to the server or interacting with client servers in any other way.

```
//RaaS information methods

//Share information about device, whether compromise or not
Future<bool> isDeviceCompromised();
//Share information about App, whether App is compromised or not
Future<bool> isAppCompromised();
//Share information about whether App Network is being proxied or not
Future<bool> isNetworkProxied();
//Share information about App platform & OS, whether App is running under simulated environment or actual device
Future<bool> isSimulator();
//Share information about App platform & OS, whether App is running under Virtual Machine or actual device
Future<bool> bool isRunningOnVM();
//Share information about the complete environment considering the above mentioned a whole set.
Future<bool> isEnvironmentSecure();
```

There are several methods mentioned above, each method serves a specific purpose in sharing information about device, network & environment details.

END USER DATA CONFIGURATION

End user data configuration can be done in two ways under the current version of Blotout's SDK. End users are actual Device users who use Blotout clients' App for a specific purpose.

Turn OFF user data collection

To turn OFF data collection, so that SDK does not even collect data events while still enabled, please use the below mentioned API.

```
void setDataCollectionEnabled(bool dataCollectionEnabled);
```

Turn ON user data collection

To turn ON data collection, so that SDK collects data events & process, please use the below mentioned API.

```
void setDataCollectionEnabled(bool dataCollectionEnabled);
```

Disable event data sync with server

To disable event data sync with the server, while SDK can still collect and store events data locally, please use the below mentioned API.

```
void setNetworkSyncEnabled(bool networkSyncEnabled)
```

Enable event data sync with servers

To enable event data sync with the server, so that the SDK can sync collected event data with the server, please use the below mentioned API. Remember this will sync all the event data collected so far and has not been synced with the server due to any reason.

```
void setNetworkSyncEnabled(bool networkSyncEnabled)
```

TROUBLESHOOTING

If you face challenges integrating Blotout even after following integration instructions in this document, enable debug mode and check for error messages. If issues persist, please email us at devehelp@blotout.io

FAQ’S

There are certain questions asked by different clients & we have documented them for finding answers here in case they are helpful.

Question: Do I have to re-register my App in staging once registered in production?

Answer: Yes, you can use the same configuration but re-registration is required, with the same account.

Question: Can I use Blotout’s SDK for funnel qualification?

Answer: Yes, funnel data should be created in the format acceptable by Blotout’s SDK & Server.

Question: Can I use Blotout’s SDK for segment qualification?

Answer: Yes, segment data should be created in the format acceptable by Blotout SDK & Server.

Question: Can I use Blotout’s SDK for campaign execution?

Answer: No, current Blotout’s SDK does not support but we are working on it & will be available soon.

Question: Can I use Blotout’s SDK for any kind of developer codified events?

Answer: Yes, any developer codified event can be sent to Blotout’s SDK and finally to server but SDK only accepts event name as string and meta information as key/value pair which can contain values in the data type format of String, INT, Float.