INFOnline
Integration Guide

Platform: Android
SZM library version: 2.1.0

INFOnline GmbH
Brühler Straße 9
53119 Bonn

Tel.: +49 (0) 228 / 410 29 - 0
Fax: +49 (0) 228 / 410 29 - 66

www.INFOnline.de
info@INFOnline.de

Inhalt

1 About this document

This document provides all of the information required for technical integration and use of the SZM library in apps that use the Android operating system. **The app platform supported** here is **Android version 4.0.3 and above**.

It is divided into 5 sections:

1. The “About this document” section provides an overview of the structure and aim of this Integration Guide.
2. The “INFOnline SZM library (Android)” explains the limiting and framework conditions for the measuring instrument.
3. The “Integration of the SZM library Android” provides the technical information for installation of the measuring instrument.
4. The “Requirements for accessing the library” deals with the requirements for using the measurement library in the context of the SZM mobile applications measurement.
5. If you have any questions or suggestions, please get in touch with us. You will find all of our contact details in the final “Contact” section.

2 INFOnline SZM library (Android)

The INFOnline SZM library for Android (also referred to hereafter as “IOLib”) is a software library that records and saves information about the use of Android apps and sends it to an appropriate backend for validation and monitoring. INFOnline provides that backend.

2.1 Provision

The IOLib Android is made available by INFOnline for download. You will be sent access details by e-mail.

The download area includes

- the release notes as a text file
- the change log as a text file
- the Android library „**infonlinelib_2.1.0.aar**“ as a binary distribution
- an Android studio project „**INFOnlineLibSample“:** a sample app with integrated IOLib Android

2.2 Requirements

The SZM library for Android supports integration via the “Android Studio” development environment with Windows/MacOSX or Linux

2.2.1 Development environment

The following prerequisites must be in place to complete integration of the IOLib Android:

- Android SDK Release 24 or higher
- Android Studio 3.0 or higher
- Google Play Services 10 or higher

Android apps that use the IOLib Android must be compiled with Android 4.0.3 (API Level 15) as a minimum. The minimum SDK Version in the manifest is therefore 15.

Note:

The IOLib compiles with Android SDK r26. Android apps that compile with older Android SDK versions will possibly get the following warning using ProGuard:

Warning: de.infonline.lib.IOLWebViewClientV24: can't find referenced method 'boolean shouldOverrideUrlLoading(android.webkit.WebView,android.webkit.WebResourceRequest)' in library class android.webkit.WebViewClient

You can ignore that warning using the ProGuard directive '-dontwarn android.webkit.WebViewClient', as the missing reference has not been added until Android SDK r24.

2.2.2 Operating system

The IOLib Android requires Android 4.0.3 or higher to operate.

2.2.3 Compability

A new version of the Android operating system may require an update to the library.

It may not be possible to guarantee complete upward compatibility under some circumstances.

2.3 Function

2.3.1 When must the library be accessed?

Accessing the functions of the IOLib within the app is linked to certain events. Specifications or recommendations about the point in the app (or the user actions) at which the SZM library should be accessed and which information should be transferred have been formulated by the agof and IVW for app providers.

The requirements for accessing the IOLib within the app are described in Chapter 4 (*Requirements for accessing the library*).

2.3.2 Offline use

The IOLib Android supports the use of mobile apps without an active internet connection. The events during offline use are saved and transferred to the measurement backend at the next opportunity (as soon as there is an internet connection). A time stamp, the relevant internet connection status and other data is recorded for each event, thereby documenting the time and framework conditions of the offline use.

2.3.3 Transmission of the measurement data

In order to organize data transmission and facilitate offline use, the measurement data is not transferred to the backend directly at the time of the measurement, but is collected in a “measurement data queue”.

The dataset to be transferred is continuously concatenated with the new measurement data; as soon as a specific limit is reached in terms of size, a new dataset is created and the previous dataset is released for transmission.

The transmission of the data itself takes place asynchronously. This avoids delaying or blocking the user's interaction with the app.

2.3.4 Privacy setting, opt-out function and data protection declaration

The user of an app must be informed that the app is measuring the user's actions and communicating with the INFOnline measurement system. INFOnline provides a data protection declaration for this purpose, which can be downloaded from https://www.infonline.de/de/extra/downloads. Please include this text in the app at an appropriate point.

In accordance with the EU General Protection Regulation (GDPR), the measurement is based on the legal basis of the legitimate interest, which is transmitted to INFOnline via a configurable variable (privacy setting).

Privacy setting when initializing the library (see chapter 3.6.2):

LIN = "Legitimate Interest"

Complete measurement and use of unique identifiers per device or app.

If you cannot carry out the measurement on the legal basis of legitimate interest according to EU GDPR, please contact our Customer Service.

NOTE

Please note that a complete measurement, after the EU GDPR has come into force from 25 May, and a participation in the agof study daily digital facts according to today's requirements, is only possible on the legal basis of the legitimate interest.
This also applies to the use of the library for in-app survey, which may only be initialized if the measurement is based on the legal basis of the legitimate interest (see Integration Guide for the in-app survey library).

The user must also be given an opt-out function. Implementation of this is the responsibility of the app developer. On integration of the function, users of the app can activate and deactivate the opt-out. When the opt-out is activated, no counter impulse is triggered.

The technical details for integrating the Opt-Out function are listed in chapter 3.6.9 (*Integration of the opt-out function*).

3 Integration of the SZM library Android

This section describes the technical integration of the SZM library Android into an Android project structure.

3.1 Thread-safe

The IOLib for Android is implemented completely thread-safe.

3.2 IOLib files

The INFOnline SZM library for Android comprises the following files/directories

- **RELEASE_NOTES.txt**

This file includes information about the releases of the IOLib.

- **CHANGELOG.txt**

This file includes a history of the changes over the individual releases of the IOLib.

- **“infonlinelib_2.1.0.aar“ file**

The IOLibrary for recording the usage data of an Android app as a binary distribution

- **“INFOnlineLibrarySample“ directory**

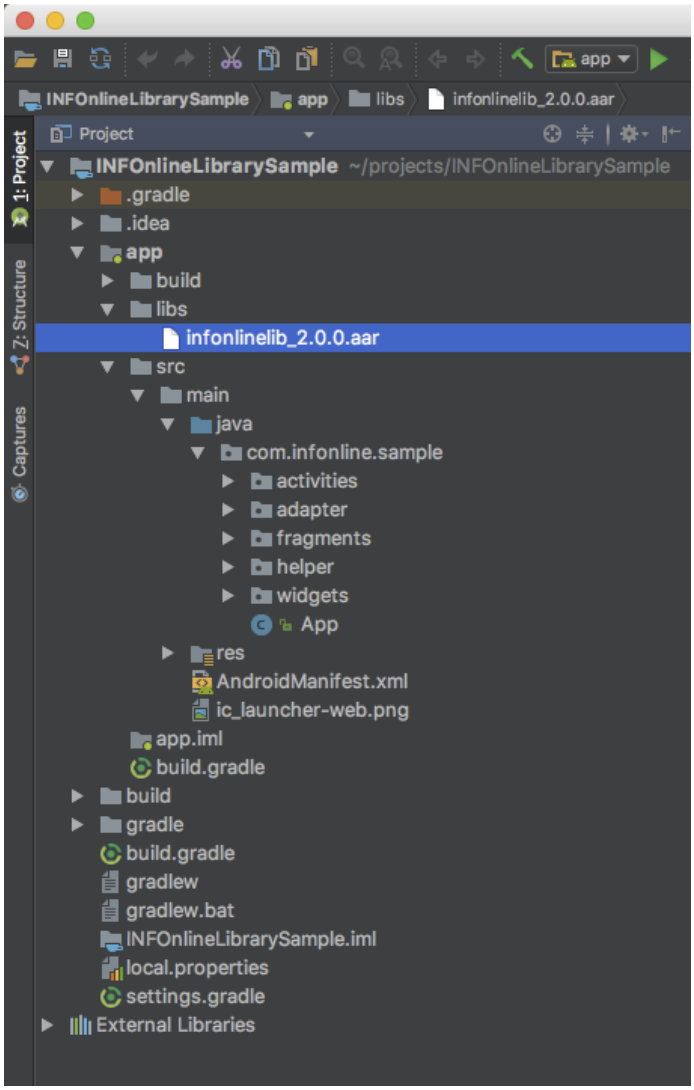
A sample project for Android Studio that demonstrates use of the IOLibrary for Android

3.3 Integration of the IOLib measurement library

Integration of the IOLib into an Android app project is described below.

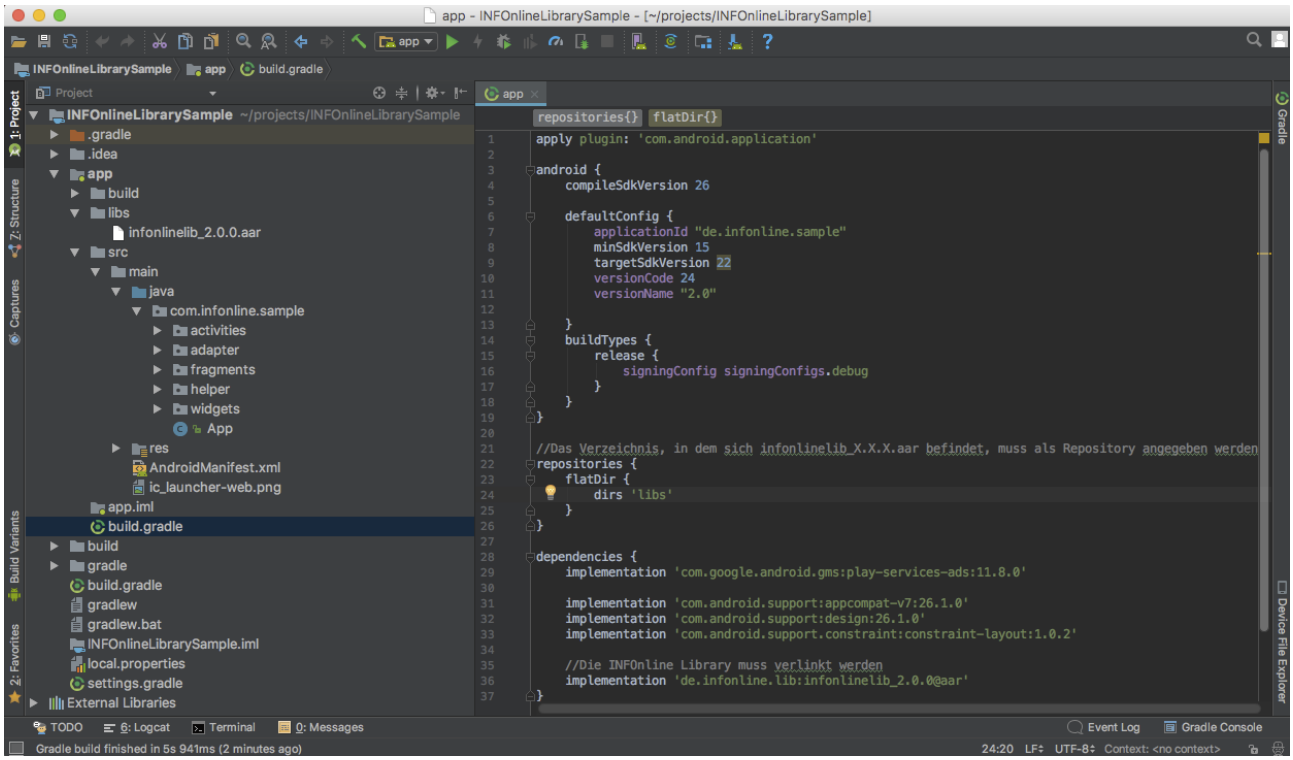
1. In Android Studio: Integration of AAR file

- Copy the file “infonlinelib_X.X.X.aar“ to the project to be measured (e.g. to <MyApp>/app/libs)


app
INFOnlineLibrarySample
app
libs
infonlinelib_2.0.0.aar
Project
1: Project
7: Structure
Captures
INFOnlineLibrarySample ~/projects/INFOnlineLibrarySample
.gradle
.idea
app
build
libs
infonlinelib_2.0.0.aar
src
main
java
com.infonline.sample
activities
adapter
fragments
helper
widgets
App
res
AndroidManifest.xml
ic_launcher-web.png
app.iml
build.gradle
build
gradle
build.gradle
gradlew
gradlew.bat
INFOnlineLibrarySample.iml
local.properties
settings.gradle
External Libraries

2. In Android Studio: adapt build.gradle

- Adapt the file <MyApp>/app/build.gradle:




```
//The directory in which infonlinelib_X.X.X.aar is located must be specified as the
repository

repositories{
          flatDir{
                 dirs 'libs'
                 }
}

dependencies {
    //INFOnline Library requires the mobile ads API of Google Play Services Library
    implementation 'com.google.android.gms:play-services-ads:11.8.0'

     //Link the INFOnline library
    implementation 'de.infonline.lib:infonlinelib_X.X.X@aar'
}
```

3. In Android Studio: Initialization of the IOLib

- Create a new class or use an existing class, derive from application
- Generate IOLib (described below)

NOTE The IOLib should be initialized in the onCreate() method of the application!

The IOLLib offers two different options for initialization:

1) The initialization of the app with implicit session start:

```
import de.infonline.lib.IOLSession;
import de.infonline.lib.IOLSessionType;
import de.infonline.lib.IOLSessionPrivacySetting;
import android.app.Application;

public class SampleApplication extends Application {

    @Override
    public void onCreate() {
        super.onCreate();
        IOLSession.getSessionForType(IOLSessionType.SZM)      // Session Type SZM
                  .initIOLSession(this,                        // Application Context
                                  "OfferIdentifier",           // Offer Identifier
                                  BuildConfig.DEBUG,           // Debug mode on/off
                                  IOLSessionPrivacySetting.LIN);  // Privacy Setting


    }
```

ALTERNATIVELY you can initialize the lib in 2 steps and start the session.

2) Pure initialization of the IOLLib:

```
import de.infonline.lib.IOLSession;
import android.app.Application;

public class SampleApplication extends Application {

    @Override
    public void onCreate() {
        super.onCreate();
        IOLSession.init(this)               // Application Context
    }
```

At runtime, the initialization of the session:

```
public void initIOLSession() {

   IOLSession.getSessionForType(IOLSessionType.SZM)          // Session Type SZM
             .initIOLSession("OfferIdentifier",              // Offer Identifier
                             BuildConfig.DEBUG,              // Debug mode on/off
                             IOLSessionPrivacySetting.LIN); // Privacy Setting

    }
```

4. In Android Studio: Code

- IOLib can be stopped and started explicitly

```
   IOLSession.getSessionForType(IOLSessionType.SZM).startSession();
   IOLSession.getSessionForType(IOLSessionType.SZM).terminateSession();
```

HINWEIS This works only if the IOL session was initialized in the onCreate() method of the application! The method is described in section 3.3 / point 3.

5. In Android Studio: Each activity

- Events are logged via the IOLSession
- Events can be logged in the activities of the app, e.g. the call of a ViewAppeared

```
    // Tracking View Appeared
    IOLSession.getSessionForType(IOLSessionType.SZM)
              .logEvent(new IOLViewEvent(IOLViewEventType.appeared));
```

3.4 Integration of the Google Play Services Library

As of August 2014, polices of the Advertising Identifier must be used in relation to audience measurement in accordance with the provisions of the Google Play Developer Program:

https://play.google.com/about/monetization-ads/ads/ad-id/

In order to record the Advertising Identifier, the Google Play Services library must be integrated into the project, but only the Google Mobile Ads API:

```
dependencies {
    implementation 'com.google.android.gms:play-services-ads:11.8.0'
}
```

However, the Advertising Identifier can only be read on devices that have Google Play installed.

Google provides a detailed guide that describes step by step how the library is integrated into the project concerned:

http://developer.android.com/google/play-services/setup.html

NOTE Integration of the Google Play Services library must also be carried out for apps that are not published in the Google Play Store!

3.5 Parallel measurement SZM & ÖWA

The IOLib Android supports the parallel operation of sessions of different measurement systems. In the following it will be shown how the measurement can be operated simultaneously for both systems.

Please note that the implementation of the library described here is only necessary if you are also a member of "Österreichische Webanalyse" (ÖWA) and would like to initiate a parallel measurement in this context.

It is absolutely necessary that you have a site ID of the INFOnline and an ID of the ÖWA (in the format: "at_x_example").

Prerequisite is an integration of the IOLib Android according to chapter 3.3. Points 3-5 are to be executed as follows:

3. In Android Studio: Initialization of the IOLib:

The IOLLib offers two different options for initialization:

1) The initialization of the app with implicit session start::

```
import de.infonline.lib.IOLSession;
import de.infonline.lib.IOLSessionType;
import de.infonline.lib.IOLSessionPrivacySetting;
import android.app.Application;

public class SampleApplication extends Application {

    @Override
    public void onCreate() {
        super.onCreate();
        IOLSession.getSessionForType(IOLSessionType.SZM)       // Session Type SZM
                  .initIOLSession(this,                       // Application Context
                                  "OfferIdentifier-SZM",      // Offer Identifier SZM
                                  BuildConfig.DEBUG,          // Debug mode on/off
                                  IOLSessionPrivacySetting.LIN);  // Privacy Setting

        IOLSession.getSessionForType(IOLSessionType.OEWA)       // Session Type ÖWA
                  .initIOLSession(this,                         // Application Context
                                  "OfferIdentifier-OEWA",      // Offer Identifier ÖWA
                                  BuildConfig.DEBUG,           // Debug mode on/off
                                  IOLSessionPrivacySetting.LIN);  // Privacy Setting
    }
```

ALTERNATIVELY you can initialize the lib in 2 steps:

2) Pure initialization of the IOLLib:

```
import de.infonline.lib.IOLSession;
import android.app.Application;

public class SampleApplication extends Application {

    @Override
    public void onCreate() {
        super.onCreate();
        IOLSession.init(this)               // Application Context
    }
```

At runtime, the initialization of the session for both measurement systems:

```
public void initIOLSession() {

    IOLSession.getSessionForType(IOLSessionType.SZM)          // Session Type SZM
              .initIOLSession("OfferIdentifier-SZM",          // Offer Identifier SZM
                              BuildConfig.DEBUG,              // Debug mode on/off
                              IOLSessionPrivacySetting.LIN); // Privacy Setting

    IOLSession.getSessionForType(IOLSessionType.OEWA)         // Session Type ÖWA
              .initIOLSession("OfferIdentifier-OEWA",         // Offer Identifier ÖWA
                              BuildConfig.DEBUG,              // Debug mode on/off
                              IOLSessionPrivacySetting.LIN); // Privacy Setting

     }
```

4. In Android Studio: Code

- Starting IOLib explicitly for both cctive sessions

```
    IOLSession.getSessionForType(IOLSessionType.SZM).startSession();
    IOLSession.getSessionForType(IOLSessionType.OEWA).startSession();
```

NOTE This works only if the IOL session was initialised in the onCreate() method of the application! Method 1) is described in chapter 3.3 / point 3.

5. In Android Studio: Each Activity

- Events are logged into the selected measuring system via the IOLSession.
- Events can be logged in the activities of the app, e.g. the call of a ViewAppeared

```
    // Tracking View Appeared
    IOLSession.getSessionForType(IOLSessionType.SZM)
              .logEvent(new IOLViewEvent(IOLViewEventType.appeared));

    IOLSession.getSessionForType(IOLSessionType.OEWA)
              .logEvent(new IOLViewEvent(IOLViewEventType.appeared));
```

3.6 IO library functions

The IOLib for Android offers the functions described below:

3.6.1 Using the IOLib in SZM Mode

IOLSession **getSessionForType(IOLSessionType iolSessionType)**

Functions of the IOLib must be called on a concrete IOLSession. To do this, pass the corresponding IOLSessionType.

Parameter:

- **IOLSessionType (mandatory)**
  The desired IOLSessionType.

Example:

```
IOLSession iolSession = IOLSession.getSessionForType(IOLSessionType.SZM);
```

3.6.2 Initialization

initIOLSession(String offerIdentifier, boolean debugModeEnabled, IOLSessionPrivacySetting privacySetting)

initIOLSession(Context context, String offerIdentifier, boolean debugModeEnabled, IOLSessionPrivacySetting privacySetting)

NOTE The IOLib must be initialized before recording of the events. In doing so, the site ID of the app must be transferred as a parameter.

Parameter:

- **Application Context (optional)**
  The Application Context of the Android app must be transferred here, if not already done via IOLSession.init(Context context).

- **Site ID (mandatory)**
  The unique ID for the service of the relevant app. The unique site ID is allocated by INFOnline for each app and each operating system.

- **debugModeEnabled (optional)**
  When debug mode is activated, the measurement library logs under the logcat tag “INFOnline”. It is recommended that the value “BuildConfig.DEBUG” should be entered here.

  The default is false if no value is transferred

- **Privacy settings (mandatory)**
  The reason for the measurement (see chapter 2.3.4). The possible values are fixed.

Example:

```
IOLSession.getSessionForType(IOLSessionType.SZM)
          .initIOLSession(this,                              // Application Context
                          "OfferIdentifier",                 // Offer Identifier
                          BuildConfig.DEBUG);                // Debug mode
                          IOLSessionPrivacySetting.LIN);     // Privacy Setting LIN
```

3.6.3 Logging an event

The measurement data is recorded by means of the **logEvent** call. Here, an instance of the measured event class is passed.

logEvent(IOLEvent event)

Parameter:

- **Event (mandatory)**

  The event to be recorded.

Some of the events are automatically recorded by the IOLib. Further details can be found in chapter 4.3.1 (*Events measured automatically by the library*) of the appendix

3.6.4 Instantiation of an event class

IOLEvent(EventType eventType, String category, String comment, Map<String, String> params)

Parameter:

- **EventType (mandatory)**

  The event to be recorded. The individual events can have different states. For example, a download may have been started, cancelled by the user, successfully performed or terminated incorrectly.

  For some events the type parameter is not necessary because only one valid type is defined for these events. Regarding the IOLCustomEvent the freely definable string parameter name is required instead of type.

- **Category (optional): Content code**
  The content code is transferred in the “category” parameter. This code is set by the provider. The code is used to identify the content displayed and is allocated by the provider in the INFOnline Customer Center to the IVW category system 2.0.

  Using the guidelines described in the following section, the provider decides whether an event constitutes a mobile PI as defined by the IVW guidelines. If an event does fall under the definition of a mobile PI, it is essential to provide a content code. If an event does not constitute a mobile PI, nil should be transmitted. This field is limited to 255 characters
- **Comment (optional): Comment**
  Comment field. This field is not limited in length.
  Transfer of this value is optional; if it is not defined, it should not be transferred.
- **params (optional): Parameter**

  A hash map with freely definable additional information about the event. Key and value must be of type string, the maximum length is limited to 255 characters.
  Passing this value is optional.

Available events

The IOLib provides the following event classes derived from "IOLEvent" with the corresponding types:

- **IOLAdvertisementEvent**
  - IOLAdvertisementEventType.Open
  - IOLAdvertisementEventType.Close

- **IOLAudioEvent**
  - IOLAudioEventType.Play
  - IOLAudioEventType.Pause
  - IOLAudioEventType.Stop
  - IOLAudioEventType.Next
  - IOLAudioEventType.Previous
  - IOLAudioEventType.Replay
  - IOLAudioEventType.SeekBack
  - IOLAudioEventType.SeekForward
- **IOLBackgroundTaskEvent**
  - IOLBackgroundTaskEventType.Start
  - IOLBackgroundTaskEventType.End

- **IOLCustomEvent**
  - **no type**
  - **name** instead (freely definable string, limited to 255 characters)

- **IOLDataEvent**
  - IOLDataEventType.Cancelled
  - IOLDataEventType.Refresh
  - IOLDataEventType.Succeeded
  - IOLDataEventType.Failed
- **IOLDeviceOrientationEvent**
  - IOLDeviceOrientationEventType.Changed

- **IOLDocumentEvent**
  - IOLDocumentEventType.Open
  - IOLDocumentEventType.Edit
  - IOLDocumentEventType.Close
- **IOLDownloadEvent**
  - IOLDownloadEventType.Cancelled
  - IOLDownloadEventType.Start
  - IOLDownloadEventType.Succeeded
  - IOLDownloadEventType.Failed
- **IOLGameEvent**
  - IOLGameEventType.Action
  - IOLGameEventType.Started
  - IOLGameEventType.Finished
  - IOLGameEventType.Won
  - IOLGameEventType.Lost
  - IOLGameEventType..NewHighscore
  - IOLGameEventTypeNewAchievement
- **IOLGestureEvent**
  - I*OLGestureEventType.Shake*
- **IOLHardwareButtonEvent**
  - I*OLHardwareButtonEventType.Pushed*
- **IOLIAPEvent**
  - IOLIAPEventType.Started
  - IOLIAPEventType.Finished
  - IOLIAPEventType.Cancelled
- **IOLLoginEvent**
  - IOLLoginEventType.Succeeded
  - IOLLoginEventType.Failed
  - IOLLoginEventType.Logout
- **IOLOpenAppEvent**
  - IOLOpenAppEventType.Maps
  - IOLOpenAppEventType.Other
- **IOLPushEvent**
  - ***I****OLPushEventType.Received*

- **IOLUploadEvent**
  - IOLUploadEventType.Cancelled
  - IOLUploadEventType.Start
  - IOLUploadEventType.Succeeded
  - IOLUploadEventType.Failed
- **IOLVideoEvent**
  - IOLVideoEventType.Play
  - IOLVideoEventType.Pause
  - IOLVideoEventType.Stop
  - IOLVideoEventType.Next
  - IOLVideoEventType.Previous
  - IOLVideoEventType.Replay
  - IOLVideoEventType.SeekBack
  - IOLVideoEventType.SeekForward
- **IOLViewEvent**
  - IOLViewEventType.Appeared
  - IOLViewEventType.Refreshed
  - IOLViewEventType.Disappeared

For further details of measurable events and the associated states, see section 4.3 (*Events*).

Examples:

IOLEventType.ViewAppeared

```
public class SampleActivity extends Activity {

    @Override
    protected void onResume() {
          super.onResume();
          IOLEvent event = new IOLViewEvent(
                IOLViewEvent.IOLViewEventType.Appeared,
                "category",
                "comment");
          IOLSession.getSessionForType(IOLSessionType.SZM).logEvent(event);
          // Other Code ..
   }
}
```

IOLEventType.ViewRefreshed

```
public class SampleActivity extends Activity {

    @Override
    protected void onResume() {
        super.onResume();
          IOLEvent event = new IOLViewEvent(
                IOLViewEvent.IOLViewEventType.Refreshed,
                "category",
                "comment");
          IOLSession.getSessionForType(IOLSessionType.SZM).logEvent(event);
          // Other Code ..
   }
}
```

IOLEventType.AudioPlay

```
public class SampleActivity extends Activity {

    @Override
    protected void onResume() {
          super.onResume();

          IOLEvent event = new IOLAudioEvent(
                IOLAudioEvent.IOLAudioEventType.Play,
                "Audio",
                "Audio Playback");
          IOLSession.getSessionForType(IOLSessionType.SZM).logEvent(event);
          // Other Code ..
    }
}
```

IOLEventType.HardwareButtonPushed

If genuine KeyEvents are to be logged, the **dispatchKeyEvent()** method must be overwritten in the corresponding activities:

```
public class SampleActivity extends Activity {

   @Override
   public boolean dispatchKeyEvent (KeyEvent event){
          if(event.getAction() == KeyEvent.ACTION_UP &&
          event.getKeyCode() == KeyEvent.KEYCODE_BACK){

                Map customParams = new HashMap<String, String>();
                customParams.put("myCustomKey", "myCustomValue");

                IOLEvent event = new IOLHardwareButtonEvent(
                      IOLHardwareButtonEvent.IOLHardwareButtonEventType.Pushed,
                      "Category",
                      "Comment",
                      customParams);
                IOLSession.getSessionForType(IOLSessionType.SZM).logEvent(event);
                // Other Code ..
          }
          return super.dispatchKeyEvent(event);
    }
```

In this example, release of the BACK key is measured with additional parameters.

3.6.5 Sending the measurement data

sendLoggedEvents()

The IOLib controls sending of the measurement data independently and entirely transparently. **sendLoggedEvents may** be accessed to force sending of the data. The IOLib then attempts to send the measured data immediately or to resend it, as soon as a data connection has been established.

Parameter:

- -

Example:

```
IOLSession.getSessionForType(IOLSessionType.SZM).sendLoggedEvents();
```

3.6.6 Debug mode

setDebugModeEnabled(boolean enable);

The measurement library can be put into debug mode. Various outputs in the log flow are generated here (errors, warnings, information, events and requests).

The default value is **false** if the measurement library was initialised with **IOLSession.initIOLSession(Context context, String offerIdentifier)**

Parameter:

- **boolean enable**
  Possible values: true|false

Example:

```
IOLSession.setDebugModeEnabled(true);
```

3.6.7 Terminate session

terminateSession()

The active IOLib session can be terminated explicitly. This facilitates an opt-out during the app runtime. The data collected up to that point is discarded and will not be sent.

NOTE: **Only use with opt-out by the user!**

Parameter:

- -

Example:

```
IOLSession.getSessionForType(IOLSessionType.SZM).terminateSession();
```

NOTE Session does not then have to be re-initialised, but can be started at any time with startSession()!

3.6.8 Start session

startSession()

If the active session of the IOLib was terminated explicitly, it can be re-started at any time with **startSession()**. Re-initialisation is not necessary.

Parameter:

- -

Example:

```
IOLSession.getSessionForType(IOLSessionType.SZM).startSession();
```

3.6.9 Integration of the opt-out function

Users of an app must be given an opt-out function. Implementation is the responsibility of the developer of the app concerned and, if activated by the user, it should lead to the SZM library either not being initialised at all or the running session being terminated explicitly. The procedure is described in section 3.6.7 (*Terminate session*).

On integration of the function, users of the app can activate and deactivate the opt-out. When the opt-out is activated, no counter impulse is triggered.

NOTE If the running session is terminated explicitly, all of the measurement data recorded up to this point but not yet sent is discarded.

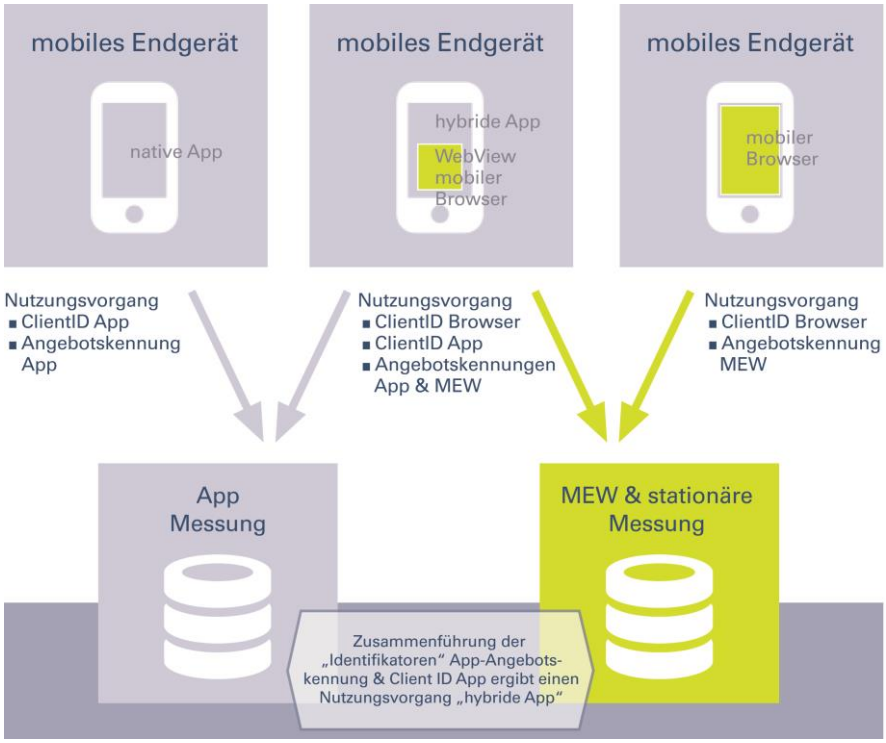
If the opt-out is revised, the measurement library should be restarted. The procedure is described in section 3.6.8 (*Start session*).

3.7 Hybrid measurement

The SZM library is able to measure the use of hybrid apps, i.e. user actions within mobile content that are represented in so-called WebViews can also be recorded and combined with the measurement data of the native app framework.

The prerequisite for this is that the websites accessed via the WebView must also be tagged with the SZM tag for mobile-enabled websites. A WebView provided by the SZM library must also be used to combine the two measurement datasets from the app measurement and the MEW measurement. The app and the MEWs accessed from the app must use different site IDs allocated by INFOnline.

The following diagram provides an overview of mobile use un SZMnG:



NOTE Hybrid apps usually obtain the external web content from websites that are optimized for the use of mobile devices. These websites are referred to in this document as MEW (=mobile-enabled website). If your hybrid app obtains content from a stationary website (optimized for the use of PCs and notebooks), all conditions described in the document INFOnline Integration Guide SZM-Tag apply.

In order to facilitate the measurement of hybrid apps, websites must be accessed in the app by means of a special WebView, the **de.infonline.lib.IOLWebView**.

The view can then be treated as a regular **android.webkit.WebView**. The **de.infonline.lib.IOLWebView** derives directly from the **android.webkit.WebView**, offers no new APIs to the outside and therefore behaves entirely transparently.

Here is an example of how to generate an **IOLWebView** in an XML layout:

```
<de.infonline.lib.IOLWebView
        android:id="@+id/webview"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
```

NOTE The IOLWebView automatically activates JavaScript and DOM storage. The IOLWebView will not work correctly if these settings are changed.

3.8 Debug information

The library can be put into debug mode for the purposes of general error analysis and, in particular, to send the measurement data. In this debug mode, the library generates various outputs in the log flow (console): errors, warnings, information, events and requests.

These outputs can then be filtered easily on the console.

In order to put the IOLib into debug mode, the **setDebugModeEnabled** method is accessed:

```
IOLSession.setDebugModeEnabled(true);
```

Debug mode is **deactivated** by default unless the debug mode was explicitly activated on initialization. The procedure is described in section 3.6.2 (*Initialization*).

IOLSession **getSessionForType(IOLSessionType iolSessionType)**

Functions of the IOLib must be called in a concrete IOLSession. To do this, pass the corresponding IOLSessionType.

Parameter:

- **IOLSessionType (mandatory)**
  The desired IOLSessionType.

Example:

```
IOLSession iolSession = IOLSession.getSessionForType(IOLSessionType.SZM);
```

NOTE **Please deactivate debug mode before publishing the app.**

3.9 Android TV / Nexus Player

The IOLib Android is completely suitable for the measurement of an Android app on the ‘Android TV / Nexus Player’ platform, as it technically concerns the same operating system as other Android devices.

If you want to use the IOLib Android among Android TV / Nexus Player, please make sure to follow the integration steps according to chapter 3.2. Regarding the methods for managing the IOLib, please use the handling given in chapter 3.4.

In addition, you can implement the hybrid measurement following the description of chapter 3.7.

3.10 Fire OS

In principle, the IOLib is suitable for the measurement of an Android app on the ‘Fire OS 5’ operation system. But, this compatibility only applies to the publishing of a native Android app on Fire OS 5, not on HTML5/WebApps.

If you want to use the IOLib Android among Fire OS 5, please make sure to follow the integration steps according to chapter 3.3. Regarding the methods for managing the IOLib, please use the handling given in chapter 3.4.

In addition, you can implement the hybrid measurement following the description of chapter 3.7.

NOTE **Currently, there is no provision of Second Screens measurement in conjunction with the Amazon Fling SDK and a Fire TV app.**

4 Requirements for accessing the library

4.1 General information

Recording of the app usage by the user is carried out by the app accessing the SZM library when defined events take place that characterise a user interaction.

The user interaction is referred to as an event.

NOTE The SZM library must be accessed explicitly by the app when the event occurs.

The IOLib also measures certain system or app-specific values automatically. Integration of the IOLib Android must therefore be carried out exactly as described in section 3.3 |point 6.

4.1.1 Interpretation of events as mobile PI

The specifications listed below define how the SZM library is to be used in the context of the SZM mobile applications measurement.

From a technical point of view, a distinction is made between 2 types of events:

1. PI events

With PI events, the event is used to generate a page impression in the same way as for the stationary web. A code must be allocated to this event. This code can then be allocated to various categories and serves as the basis for creating booking units. With PI events, the IVW's specifications for mobile impressions must be observed:

a mobile impression is a user action within a mobile service that leads or could lead to accessing advertising. Each user action may only be counted once. User actions that do not lead to a potential delivery of advertising may not be counted.

"Prerequisites for the allocation of an MI to a service:
the content delivered must have the FQDN (for mobile enabled websites) or the app name of the service (for apps – or an alias/redirect) or the allocated MEW or app name of the service.

User action:
An MI is triggered by an action carried out by a user.
This also includes: reloading, opening an app, opening a browser

No user action:

Opening of content by automatic forwarding (except for redirects and aliases), automatic reload, opening content when closing (including: background) a browser window or an app, opening content via robots/spiders, etc

No mobile impression:
Scrolling within content that has already been loaded.

2. Non-PI-Events

Non-PI events are user actions that are recorded as events in the SZM system but do not lead to counting of a mobile impression. A **code may not be** allocated to this event. Examples of non-PI events are

- events recorded automatically by the IOLib
- events defined by the provider as not representing a mobile PI that are nevertheless to be measured as events to improve tracking of use of the app by users, for example

4.2 Guidelines for allocation of the codes

For **PI events**, the code must be given as a unique identifier of the content displayed. This code is specified by the app provider.

When specifying the content code, the INFOnline code guidelines must be observed

- Length of the codes: a code may contain a maximum of 255 characters
- Number of codes: a maximum of 2,000 codes may be used
- Permitted characters: a-z, A-Z, 0-9; comma „,“; hyphen „-“; underscore „_“; slash „/“

Detailed information of the INFOnline code guidelines can be found in the “INFOnline Configuration Guide” at:

https://www.infonline.de/downloads

4.3 Events

The following tables list events that are or can be recorded in the measurement. The circumstances under which an event can lead to a page impression are also explained below

4.3.1 Events measured automatically by the library

The following table describes the events for which the SZM library is accessed automatically. The events are user actions that are recorded for technical reasons but do not count as a mobile impression. A code may not be allocated to this event.

Event class	Event type	Event	Note
IOLApplicationEvent	Start, EnterBackground, EnterForeground, Crashed	App-specific event, e.g. start the app, exit the app, crash, etc.	ResignActive: incoming call, Push-Notification Alert, Timer Alarm, etc. EnterFore/Background: app goes into the background Terminate: app is closed
IOLInternetConnectionEvent	Established Lost SwitchedInterface	Type of connectivity changes	Connection established or lost Change from mobile to WiFi or vice versa
IOLWebViewEvent	Init	Hybrid measurement is activated	

The “events measured automatically by the library” are recorded as soon as methods described under section 3.3 point 3 are used.

4.3.2 PI events

Events that typically lead to triggering of a PI are listed below. Application of the scheme is recommended. The events must be triggered manually, automatic recording is not carried out. The procedure is described in Chapter 3.6.3 (*Logging an Event*). A code must be allocated to PI events. This code can subsequently be allocated to various categories and serves as the basis for creating booking units.

NOTE The IVW’s specifications for mobile impressions should be observed for PI events.

Event class	Event type	Event	Note
IOLDeviceOrientationEvent	Changed	Orientation of the device	LandscapeLeft / LandscapeRight or Portrait / Portrait UpsideDown
IOLGestureEvent	Shake	Device is shaken	
IOLViewEvent	Appeared Refreshed	A view (aka “page”) was displayed or updated with new data	Ecamples: Appeared: initial opening of a page

			Refreshed: Search filter or update of data
IOLGameEvent	Action Started	Gaming events	Action within a game Game started
IOLAudioEvent	Play Pause Stop Next Previous Replay SeekBack SeekForward	Audio Playback	Play, pause, stop, next/previous title, re-wind/fast forward, replay
IOLVideoEvent	Play Pause Stop Next Previous Replay SeekBack SeekForward	Video Playback	Play, pause, stop, next/previous title, re-wind/fast forward, replay

4.3.3 Non-PI events

Events that typically do not lead to counting of a PI are listed below. If, however circumstances obtain in individual cases under which a mobile PI should also be generated here, these events can be used to enable this. Before using these events to generate PIs, please clarify the issue directly with the IVW. If these events should lead to counting of PIs, this must be triggered manually here. A code must then be allocated to this event. This code can subsequently be allocated to various categories and serves as the basis for creating booking units.

Event class	Event type	Event	Note
IOLViewEvent	Disappeared	A view (aka “page”) is left	Examples: Disappeared: screen left
IOLDocumentEvent	Open Edit Close	Document / list editing	Edit list Document saved
IOLDataEvent	Cancelled Refresh	Data connection/ processing	Data connection lost Data has been updated Data was transferred successfully

	Succeeded Failed		Data was not transferred
IOLDownloadEvent	Cancelled Start Succeeded Failed	Download of data	Download wurde initiiert Download wurde abgebrochen Download erfolgreich beendet Download fehlgeschlagen
IOLUploadEvent	Cancelled Start Succeeded Failed	Upload von Daten	Download was initiated Download was cancelled Download completed successfully Download failed
IOLLoginEvent	Succeeded Failed Logout	Login	Login completed successfully Login failed Logout completed/session terminated
IOLGameEvent	Finished Won Lost NewHighscore NewAchievement	Gaming events	Game finished (Round of) game won (Round of) game lost New high score achieved New achievement attained
IOLHardwareButtonEvent	Pushed	Switch or button on device pushed	volume over rocker switch on device changed, device locked (Power key pressed), home button pressed
IOLBackgroundTaskEvent	Start End	A background process is started or ended	Download or upload of larger files, which should possibly continue in the background
IOLOpenAppEvent	Maps Other	Another app is started or the app is left via a URL	Maps: Apple Maps is opened Other: other apps or URLs are opened (e-mail, phone, websites, etc.)
IOLAdvertisementEvent	Open, Close	Advertisement is displayed or hidden	Advertising banner open or closed
IOLIAPEvent	Started, Finished, Cancelled	In-App purchases are carried out	IAP process is initiated (start) IAP process is terminated (finished) IAP process is terminated prematurely (cancelled

IOLPushEvent	Received	Push Notifications	Receive push within the app (receipt of push notification outside the app is not measured)
IOLCustomEvent	*	Definable by the user	A CustomEvent can measure a freely definable status or action (intended for future use).

As soon as an event described under “events to be triggered manually” is triggered in an app, the measurement library must be opened via **logEvent**. The enums listed in the “Event” column must be transferred as parameters in this process. An instance of the corresponding IOLEvent subclass must be passed as parameter (see also chapter 3.6.4 (***Instantiation of an event class***).

Transfer of the non-PI events to the SZM system is optional.
NOTE The non-PI events have no effect on determining the range of your app.

These events **can** be used by you for qualitative determination of the frequency of occurrence of these events (in the INFOnline analysis systems). It should be noted here that recording and transfer of non-PI events uses technical resources on the end device (CPU time, network traffic, battery).

In view of the usage of technical resources on the end device, we request that you decide when planning implementation of the measurement library’s implementation whether your app should transfer non-PI events to the SZM system.

4.4 Special case event „ViewRefreshed“

If a screen is updated (IOLViewEvent.IOLViewEventType.Refreshed), the following should be noted

NOTE The event may only be logged (or the SZM library accessed) if the refresh of the data was triggered manually by the user. With an automatic refresh, the event may not be logged.

5 Contact

You can contact the Customer Service team any working day between 9 a.m. and 6 p.m. by

telephone:	+49 (0)228 / 410 29 – 77
e-mail for organisational queries:	service@INFOnline.de
e-mail for technical queries:	support@INFOnline.de