INFOnline
Integration Guide

Plattform: Android
SZM-Library Version: 1.1.5

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 Über dieses Dokument

Das vorliegende Dokument dient dazu, alle notwendigen Informationen für die technische Integration und Nutzung der SZM-Library in Apps mit dem Betriebssystem Android zur Verfügung zu stellen. **Die unterstützte App-Plattfom** ist hier **Android ab Version 2.3.3.**

Es ist in 5 Kapitel gegliedert:

1. Kapitel „Über dieses Dokument“ ermöglicht einen Überblick zum Aufbau und zur Zielsetzung dieses Integration Guides.
2. Kapitel „INFOnline SZM-Library (Android)“ erläutert die Rand- und Rahmenbedingungen des Messinstruments.
3. Kapitel „Integration der SZM-Library Android“ liefert die technischen Informationen für den Einbau des Messinstruments.
4. Kapitel „Vorgaben zum Aufruf der Library“ geht auf Vorgaben zur Verwendung der Messlibrary im Kontext der Messung SZM Mobile Applications ein.
5. Falls Sie Fragen oder Anregungen haben, kontaktieren Sie uns einfach. Alle Kontaktdaten finden Sie im Schlusskapitel „Kontakt“.

2 INFOnline SZM-Library (Android)

Die INFOnline SZM-Library für Android (im Folgenden auch „IOLib“ genannt) ist eine Software-Bibliothek, welche die Nutzung von Android Apps erfasst, speichert und zum Zwecke der Validierung und Kontrolle an ein geeignetes Backend versendet. INFOnline stellt dieses Backend bereit.

2.1 Bereitstellung

Die IOLib Android wird von INFOnline zum Download zur Verfügung gestellt. Die Zugangsdaten erhalten Sie per E-Mail.

Im Download Bereich befinden sich

- die Release Notes als Textdatei
- das Change Log als Textdatei
-
- die Android Library „**infonlinelib_1.1.5.aar**“ als binäre Distribution
-
- ein Android Studio Projekt „**INFOnlineLibSample_AndroidStudio**: eine Beispiel-App mit integrierter IOLib Android

2.2 Anforderungen

Die SZM-Library für Android unterstützt die Integration über die Entwicklungsumgebung „Android Studio“ unter Windows/MacOSX bzw Linux.

2.2.1 Entwicklungsumgebung

Um die Integration der IOLib Android durchzuführen, sind folgende Voraussetzungen notwendig:

- Android SDK Release 23 und höher
-
- Android Studio 2.0 und höher
- Google Play Services 8.1 und höher

Android Apps, welche die IOLib Android benutzen, müssen min. mit Android 2.3.3 (API Level 10) kompiliert werden. Minimum SDK Version im Manifest ist somit 10.

Hinweis:

Die IOLib kompiliert mit Android SDK r24. Android Apps, welche mit älteren Versionen des Android SDK kompiliert werden bei Verwendung von ProGuard womöglich mit folgender Warnung konfrontiert:

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

Diese Warnung kann mit der ProGuard direktive '-dontwarn android.webkit.WebViewClient' ignoriert werden, da die fehlende Referenz erst mit Android SDK r24 hinzugefügt wurde.

2.2.2 Betriebssystem

Die IOLib Android setzt für den Betrieb Android 2.3.3 oder höher voraus.

2.2.3 Kompatibilität

Eine neue Version des Android Betriebssystems macht evtl. ein Update der Library notwendig.

Eine vollständige Aufwärtskompatibilität kann u.U. nicht gewährleistet werden.

2.3 Funktionsweise

2.3.1 Wann muss die Library aufgerufen werden ?

Der Aufruf der Funktionen der IOLib innerhalb der App ist an bestimmte Events gebunden. Hierzu werden für die App-Anbieter seitens der AGOF und IVW Vorgaben bzw. Empfehlungen formuliert, an welchen Stellen der App (bzw. bei welchen Nutzer-Aktionen) die SZM-Library aufgerufen werden soll und welche Informationen zu übermitteln sind.

2.3.2 Offline-Nutzung

Die IOLib Android unterstützt die Nutzung mobiler Apps ohne aktive Internet-Verbindung. Die Events der Offline-Nutzung werden gespeichert und bei nächster Gelegenheit (sobald eine Internet-Verbindung besteht) an das Mess-Backend übertragen. Pro Event werden ein Timestamp, der jeweilige InternetConnection-Status sowie weitere Daten erfasst und dokumentieren damit den Zeitpunkt und die Rahmenbedingungen der Offline-Nutzung.

2.3.3 Übertragung der Messdaten

Um die Datenübertragung möglichst effizient zu gestalten und die Offline-Nutzung zu ermöglichen, werden die Messdaten nicht direkt synchron zum Zeitpunkt der Messung an das Backend übertragen, sondern in einer „Messdaten-Queue“ gesammelt.

Der zu übertragende Datensatz wird kontinuierlich mit den neuen Messdaten konkateniert, sobald ein bestimmtes Größenlimit erreicht ist, wird ein neuer Datensatz gebildet und der vorherige Datensatz zum Versand freigegeben.

Der Versand der Daten selbst findet asynchron statt. Dadurch wird die Interaktion des Benutzers mit der App nicht verzögert oder blockiert.

2.3.4 Opt-out-Funktion und Datenschutzerklärung

Die User einer App müssen informiert werden, dass die App die Nutzeraktionen misst und an das Messsystem der INFOnline übertragt. Für diesen Zweck stellt INFOnline eine Datenschutzerklärung zur Verfügung, die unter https://www.infonline.de/de/extra/downloads/ abgerufen werden kann. Bitte binden Sie diesen Text an entsprechender Stelle in die App ein.

Darüber hinaus muss dem Nutzer eine Opt-out-Funktion gegeben werden. Die Implementierung dieser obliegt dem Entwickler der jeweiligen App. Nach Einbindung der Funktion können die Nutzer der App das Opt-out aktivieren und deaktivieren. Sofern das Opt-out aktiviert wird, wird kein Zählimpuls ausgelöst.

2.3.5 Thread-safe

Die IOLib für Android ist vollständig Thread-safe implementiert.

3 Integration der SZM-Library Android

Dieses Kapitel beschreibt die technische Integration der SZM-Library Android in eine Android-Projektstruktur.

3.1 IOLib Dateien

Die INFOnline SZM-Library für Android umfasst folgende Dateien/Verzeichnisse

- **CHANGELOG.txt**

Diese Datei enthält eine Historie der Änderungen der einzelnen Releases der IOLib.

- **RELEASE_NOTES.txt**

Diese Datei enthält Informationen zum Release der IOLib.

- **Datei „infonlinelib_1.1.5.aar“**

Die IOLibrary zur Erfassung der Nutzungsdaten einer Android-App als binäre Distribution (s. http://tools.android.com/tech-docs/new-build-system/aar-format)

- **Verzeichnis „INFOnlineLibrarySample_AndroidStudio“**

Ein Beispiel-Projekt für Android Studio, welches den Einsatz der IOLibrary für Android demonstriert

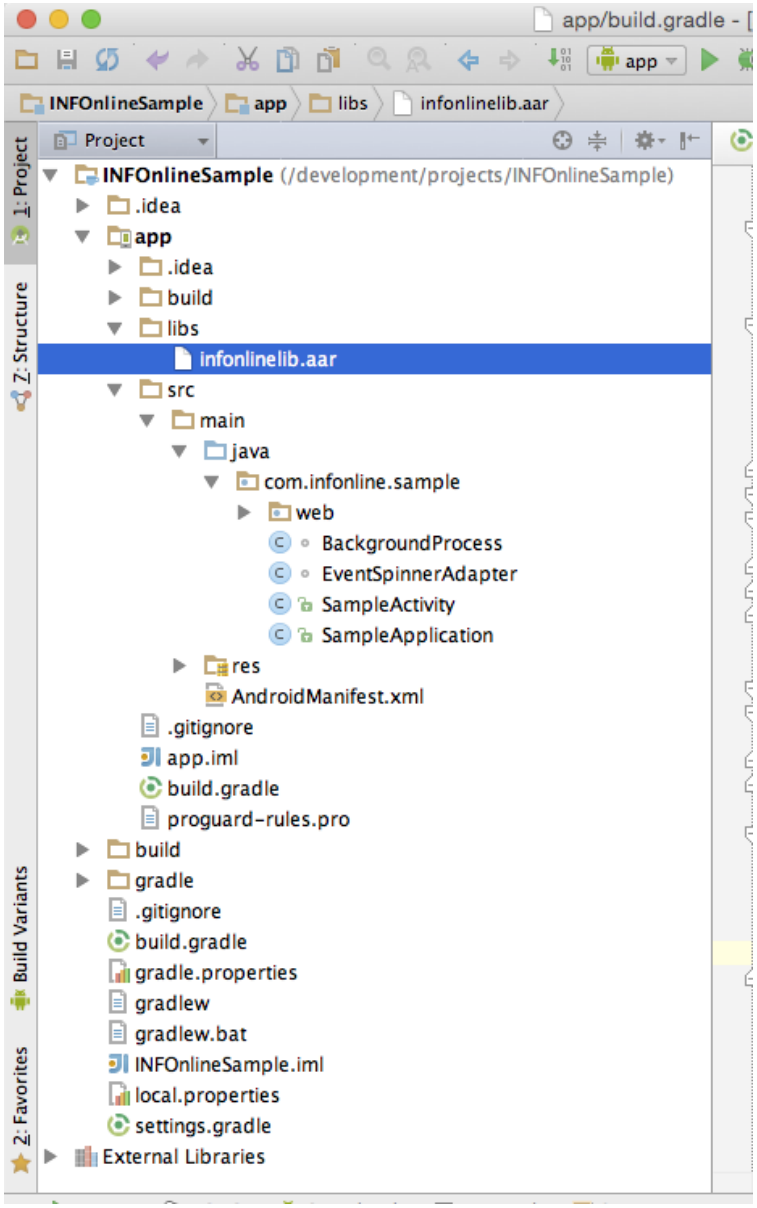
3.2 Integration der IOLib MessLibrary

Im Folgenden wird die Integration der IOLib in ein Android App Projekt beschrieben. Dazu gibt es je nach verwendeter IDE spezifische und allgemeine Tätigkeiten. Daher muss 3.2.1 *(Integration in Android Studio)* in Kombination mit 3.2.2 *(Integration allgemein)* durchgeführt werden.

3.2.1 Integration in Android Studio

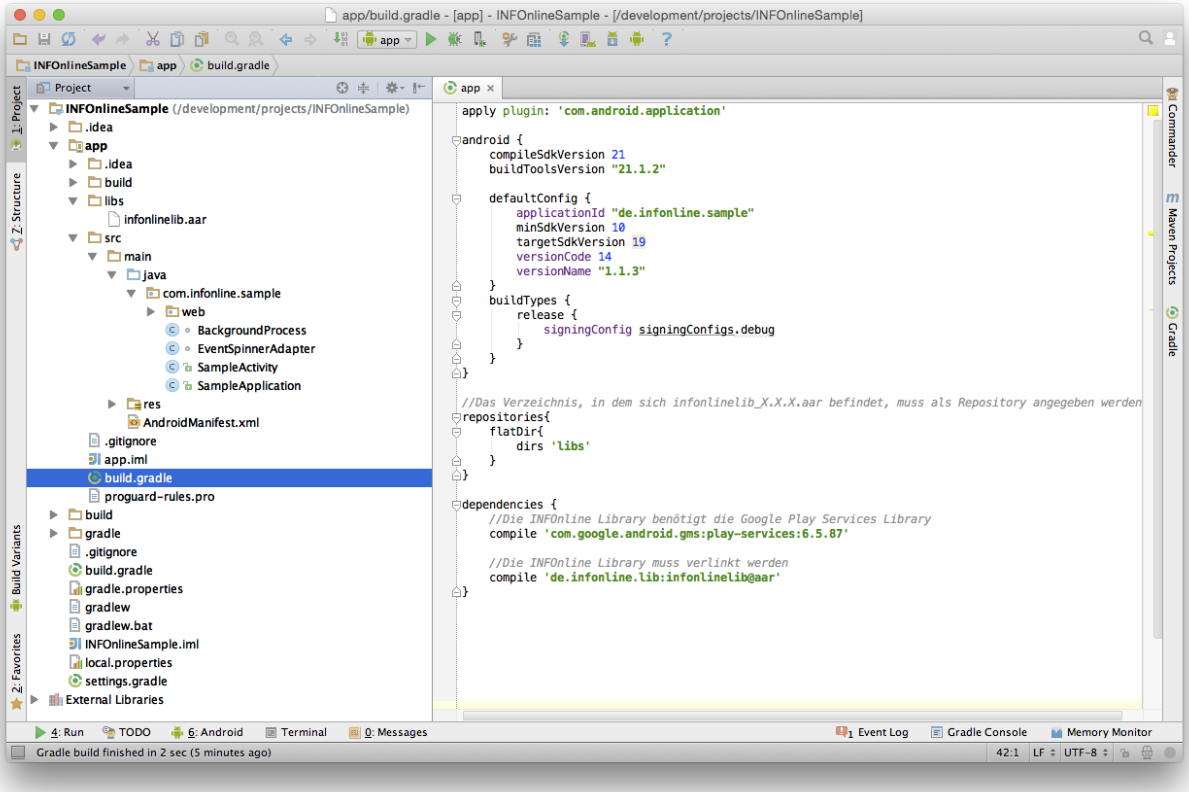
1. Im Android Studio:

- Die Datei „infonlinelib_X.X.X.aar“ in das zu messende Projekt kopieren (z.b. nach <MyApp>/app/aar)


app/build.gradle -
app
INFOnlineSample
app
libs
infonlinelib.aar
Project
1: Project
7: Structure
Build Variants
2: Favorites
INFOnlineSample (/development/projects/INFOnlineSample)
.idea
app
.idea
build
libs
infonlinelib.aar
src
main
java
com.infonline.sample
web
BackgroundProcess
EventSpinnerAdapter
SampleActivity
SampleApplication
res
AndroidManifest.xml
.gitignore
app.iml
build.gradle
proguard-rules.pro
build
gradle
.gitignore
build.gradle
gradle.properties
gradlew
gradlew.bat
INFOnlineSample.iml
local.properties
settings.gradle
External Libraries

2. In Android Studio:

- Die Datei <MyApp>/app/build.gradle anpassen:




```
//Das Verzeichnis, in dem sich infonlinelib_X.X.X.aar befindet, muss als repository
angegeben werden

repositories{
          flatDir{
                 dirs 'libs'
                 }
}

dependencies {
    //INFOnline Library benötigt die Mobile Ads API der Google Play Services Library
    compile 'com.google.android.gms:play-services-ads:8.4.0'

     //Die INFONline Library verlinken
    compile 'de.infonline.lib:infonlinelib_X.X.X@aar'
}
```

3.2.2 Integration allgemein

3. In Android Studio

- Neue Klasse erstellen oder bestehende Klasse verwenden, von Application ableiten

- IOLib per initIOLSession erzeugen, dies startet implizit die IOLib Instanz

HINWEIS Die IOLib sollte in der onCreate() Methode der Application initialisiert werden!

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

public class SampleApplication extends Application {

    @Override
    public void onCreate() {
        super.onCreate();
        IOLSession.initIOLSession(this,                // Application Context
                                  "OfferIdentifier",   // Offer Identifier
                                  BuildConfig.DEBUG); //Debug mode    on/off
    }
```

4. In Android Studio: Code

- IOLib kann explizit gestoppt und gestartet werden

```
        IOLSession.startSession();
        IOLSession.terminateSession();
```

HINWEIS Dies funktioniert nur, wenn die IOLSession in der onCreate() Methode der Application initialisiert wurde! Die Methode ist in Kapitel 3.2.2 / Pkt.3 beschrieben.

5. In Android Studio: AndroidManifest.xml

- Die Application muss als Attribut im AndroidManifest.xml unter Application Name eingetragen werden, z.B. <application android:name="com.my.package.SampleApplication" />

6. **In Android Studio:**
   HINWEIS Um die Messung der automatischen Events zu aktivieren, müssen entweder

   - **alle Activities von „IOLActivity“ abgeleitet werden**
   - **in jeder Activity der App muss folgender markierter Code eingebunden werden:**

```
package de.infonline.lib;

import de.infonline.lib.IOLSession;
import android.app.Activity;

public class IOLActivity extends Activity {

    @Override
    protected void onStart() {
        super.onStart();
        IOLSession.onActivityStart();
    }

    @Override
    protected void onStop() {
        IOLSession.onActivityStop();
     super.onStop();
    }
```

   - Dies ist notwendig, damit die IOLib z.B. messen kann, ob sich die App im Hintergrund befindet oder nicht.

7. **In Android Studio: Jede Activity**

   - Imports aller Klassen des IOLib-Packages

```
import de.infonline.app.IOLActivity;
import de.infonline.lib.IOLEventType;
import de.infonline.lib.IOLEventType.IOLAd;
import de.infonline.lib.IOLEventType.IOLPaid;
```

8. **In Android Studio:**

   - Events werden über die IOLSession geloggt.
   - Events können in den Activities der App geloggt werden, z.B. den Aufruf eines ViewAppeared

```
    // Tracking View Appeared
    IOLSession.logEvent(IOLEventType.ViewAppeared);
```

3.2.3 Hinweis für Target SDK 23

Durch die Einführung der 'Runtime Permissions' kann die IMEI ohne die erforderliche Permission 'android.permission.READ_PHONE_STATE, auf Android 6 Geräten nicht mehr ermittelt werden.

Sollte die IMEI (oder MEID) relevant sein, stellen Sie sicher, dass Sie frühzeitig die Permission vom Benutzer einfordern.

https://developer.android.com/training/permissions/requesting.html

Eine beispielhafte Implementierung ist in der mitgelieferten Sample-Anwendung zu finden.

3.3 Integration der Google Play Services Library

Ab August 2014 muss gemäß den Bestimmungen der Google Play Developer Program Policies der AdvertisingIdentifier in Bezug auf AudienceMeasurement zum Einsatz kommen:

http://play.google.com/about/developer-content-policy.html

Um den AdvertisingIdentifier zu erfassen, muss die Google Play Services Bibliothek in das Projekt eingebunden werden, jedoch nur die Google Mobile Ads API:

```
dependencies {
    compile 'com.google.android.gms:play-services-ads:8.4.0'
}
```

Der AdvertisingIdentifier kann jedoch nur auf Geräten ausgelesen werden, die Google Play installiert haben.

Google stellt eine ausführliche Anleitung bereit, die Schritt für Schritt beschreibt, wie man die Bibliothek in das jeweilige Projekt integriert:

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

HINWEIS Die Integration der Google Play Services Library muss auch für Apps erfolgen, welche nicht im Google Play Store veröffentlicht werden!

3.4 SZM-Library Funktionen

Die SZM-Library für Android bietet die im Folgenden beschriebenen Funktionen:

3.4.1 Initialisierung

initIOLSession(Context context, String offerIdentifier, boolean debugModeEnabled)

HINWEIS Die IOLib muss vor der Erfassung der Events initalisiert werden. Dabei muss die Angebotskennung der App als Parameter übergeben werden.

Parameter:

- **Application Context (mandatory)**
  Der Application Context der Android App muss hier übergeben werden.

- **Angebotskennung (mandatory)**
  Die eindeutige Kennung des Angebots der jeweiligen App. Die Angebotskennung wird von der INFOnline pro App und pro Betriebssystem eindeutig vergeben.

- **debugModeEnabled (optional)**
  Wenn der Debug Modus aktiviert ist, loggt die MessLibrary unter dem logcat tag "INFOnline". Es wird empfohlen, hier den Wert „BuildConfig.DEBUG“ zu übergeben.

  Default ist false, falls kein Wert übergeben wird.

Beispiel:

```
IOLSession.initIOLSession(this,                 // Application Context
                          "OfferIdentifier",  // Offer Identifier
                          BuildConfig.DEBUG); // Debug mode
```

3.4.2 Logging eines Events

Die Messdaten werden mittels des Aufrufs **logEvent** erfasst. Dabei können bis zu 3 Parameter übergeben werden, 2 davon sind optional.

logEvent(IOLEventType eventType)

logEvent(IOLEventType eventType, String category, String comment)

Der erste Aufruf ist eine Convenience-Funktion, welche intern die Letztgenannte aufruft.

Die fehlenden Werte werden dann um Default-Werte ergänzt.

Einige der Events werden durch die IOLib automatisch erfasst. Weitere Details hierzu finden sich im Anhang unter Kapitel 4.3.1 (*Automatisch durch die Library gemessene Events*).

Parameter:

- **EventType (mandatory)**

Das zu erfassende Event. Die IOLib stellt Konstanten innerhalb des Enums „IOLEventType“ zur Verfügung:

EVENT: *BackgroundTask*

- BackgroundTaskStart
- BackgroundTaskEnd

EVENT: *DeviceOrientation*

- DeviceOrientationChanged

EVENT: *Data*

- DataCancelled
- DataRefresh
- DataSucceeded
- DataFailed

EVENT: *Document*

- DocumentOpen
- DocumentEdit
- DocumentClose

EVENT: *Download*

- DownloadCancelled
- DownloadStart
- DownloadSucceeded
- DownloadFailed

EVENT: *Game*

- GameAction
- GameStarted
- GameFinished
- GameWon
- GameLost
- GameNewHighscore
- GameNewAchievement

EVENT: *Gesture*

- GestureShake

EVENT: *HardwareButton*

- HardwareButtonPushed

EVENT: IAP

- IAPStarted
- IAPFinished
- IAPCancelled

EVENT: Login

- LoginSucceeded
- LoginFailed
- LoginLogout

EVENT: Audio

- AudioPlay
- AudioPause
- AudioStop
- AudioNext

- AudioPrevious
- AudioReplay
- AudioSeekBack
- AudioSeekForward

EVENT: Video

- VideoPlay
- VideoPause
- VideoStop
- VideoNext
- VideoPrevious
- VideoReplay
- VideoSeekBack
- VideoSeekForward

EVENT: Push

- PushReceived

EVENT: Upload

- UploadCancelled
- UploadStart
- UploadSucceeded
- UploadFailed

EVENT: View

- ViewAppeared
- ViewRefreshed
- ViewDisappeared

Für weitere Details zu den messbaren Events und der dazugehörigen States sind in Kapitel 4.3 (*Events*) beschrieben.

- **Category (optional): Inhaltscode**
  Der Inhaltscode wird im Parameter “category“ übermittelt. Dieser Code wird vom Anbieter selbst festgelegt. Der Code dient zur inhaltlichen Kennzeichnung des angezeigten Content und wird vom Anbieter im INFOnline Kundencenter dem IVW Kategoriensystem 2.0 zugeordnet.

  Der Anbieter entscheidet anhand der im folgenden Kapitel beschriebenen Richtlinien, ob ein Event eine mobile PI im Sinne der IVW Richtlinien darstellt. Wenn ein Event unter die Definition einer mobilen PI fällt, ist zwingend ein Inhaltscode mitzugeben. Stellt ein Event keine mobile PI dar, soll nil übergeben werden. Die Länge dieses Feldes ist auf 255 Zeichen beschränkt.
- **„comment“: Kommentar**
  Kommentarfeld. Die Länge dieses Feldes ist nicht beschränkt.
  Übergabe dieses Wertes ist optional, ist er nicht definiert, soll er nicht übergeben werden.

Beispiele:

IOLEventType.ViewAppeared

```
public class SampleActivity extends IOLActivity {

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

      IOLSession.logEvent(IOLEventType.ViewAppeared, "category", "comment");
      // Other Code ..
   }
}
```

IOLEventType.ViewRefreshed

```
public class SampleActivity extends IOLActivity {

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

      IOLSession.logEvent(IOLEventType.ViewRefreshed, "category", "comment");
      // Other Code ..
   }
}
```

IOLEventType.AudioPlay

```
public class SampleActivity extends IOLActivity {

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

        IOLSession.logEvent(IOLEventType.AudioPlay, "Audio", "Audio Playback");

    // Other Code ..
    }
}
```

IOLEventType.HardwareButtonPushed

Falls echte KeyEvents geloggt werden sollen, muss die Methode **dispatchKeyEvent()** in den entsprechenden Activities überschrieben werden:

```
@Override
   public boolean dispatchKeyEvent (KeyEvent event){
          if(event.getAction() == KeyEvent.ACTION_UP &&
             event.getKeyCode() == KeyEvent.KEYCODE_BACK){
                IOLSession.logEvent(IOLEventType.HardwareButtonPushed, "", "");
          }
          return super.dispatchKeyEvent(event);
   }
```

In diesem Beispiel wird das Loslassen der BACK-Taste geloggt.

3.4.3 Versand der Messdaten

sendLoggedEvents()

Die IOLib steuert den Versand der Messdaten selbständig und völlig transparent für den Enduser. Um den Versand der Daten zu forcieren, **kann sendLoggedEvents** aufgerufen werden. Die IOLib versucht dann, die Messdaten sofort bzw. nochmals zu versenden, sobald eine Datenverbindung aufgebaut wurde.

Parameter:

- -

Beispiel:

```
IOLSession.sendLoggedEvents();
```

3.4.4 Debug Modus

setDebugModeEnabled(boolean enable);

Die Messlib kann in einen Debug-Modus versetzt werden. Hier werden diverse Ausgaben im Logstrom erzeugt (Fehler, Warnungen, Infos, Events und Requests).

Default-Wert ist **false**, wenn die MessLibrary mit **IOLSession.initIOLSession(Context context, String offerIdentifier)** initialisiert wurde.

Parameter:

- **boolean enable**
  Mögliche Werte: true|false

Beispiel:

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

3.4.5 Session beenden

terminateSession()

Die aktive Session der IOLib kann explizit beendet werden. Dies ermöglicht ein Opt-out während der App-Laufzeit. Die bis dahin erfassten Daten werden verworfen und auch nicht mehr versendet **HINWEIS**: **Nur bei Opt-Out durch den Nutzer verwenden!**

Parameter:

- -

Beispiel:

```
IOLSession.terminateSession();
```

HINWEIS Session muss anschließend nicht neu initialisiert werden, sondern kann per startSession() jederzeit gestartet werden!

3.4.6 Session starten

startSession()

Wurde die aktive Session der IOLib explizit beendet, kann diese jederzeit per **startSession()** erneut gestartet werden. Eine Neuinitialisierung ist nicht notwendig.

Parameter:

- -

Beispiel:

```
IOLSession.terminateSession();
```

3.4.7 Einbindung Opt-out Funktion

Den Nutzer einer App muss eine Opt-out Funktion gebeben werden. Die Implementierung obliegt dem Entwickler der jeweiligen App und sollte bei Aktivierung durch den Benutzer dazu führen, dass die SZM-Library entweder gar nicht initialisiert wird oder die laufende Session explizit beendet wird. Das Vorgehen ist in Kapitel 3.4.5 (*Session beenden*) beschrieben.

Nach Einbindung der Funktion können die Nutzer der App das Opt-out aktivieren und deaktivieren. Sofern Opt-out aktiviert wird, wird kein Zählimpuls ausgelöst.

HINWEIS Wird die laufende Session explizit beendet, dann werden alle bis dahin erfassten, aber noch nicht versandten Messdaten verworfen.

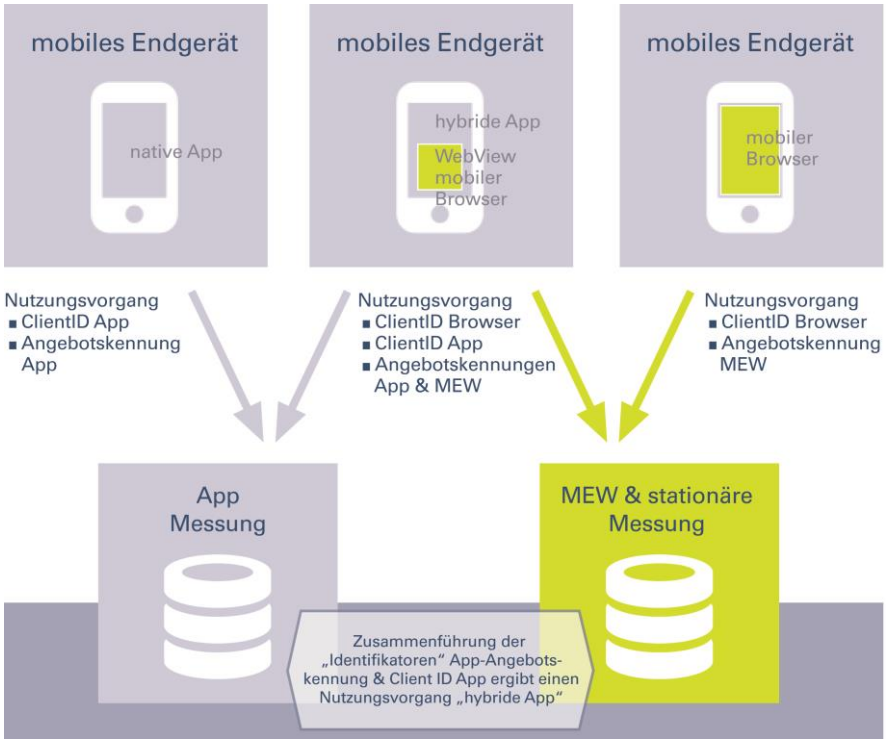
Wird das Opt-out revidiert, dann sollte die MessLib wieder gestartet werden. Das Vorgehen ist in Kapitel 3.4.6 (*Session starten*) beschrieben.

3.5 Hybrid-Messung

Die SZM-Library ist in der Lage die Nutzung von Hybrid Apps zu messen, d.h. auch Nutzeraktionen innerhalb mobiler Inhalte, welche in sog. WebViews dargestellt werden, können erfasst und mit den Messdaten des nativen App-Rahmens zusammengezuführt werden.

Voraussetzung hierfür ist, dass die über den WebView aufgerufenen Webseiten ebenfalls mit dem SZM Tag für mobile enabled Websites vertaggt sind. Außerdem muss ein von der SZM-Library bereitgestellter WebView verwendet werden, um die beiden Messdatensätze aus der App-Messung und der MEW Messung zusammenzuführen. Die App und die aus der App aufgerufenen MEWs müssen unterschiedliche, von INFOnline vergebene Angebotskennungen verwenden.

Nachfolgendes Schaubild zeigt eine Übersicht mobiler Nutzung im SZMnG:



Um die Messung von Hybrid-Apps zu ermöglichen, müssen Webseiten in der App durch einen speziellen WebView, den **de.infonline.lib.IOLWebView**, aufgerufen werden.

Anschließend kann der View wie ein regulärer **android.webkit.WebView** behandelt werden. Der **de.infonline.lib.IOLWebView** leitet direkt vom **android.webkit.WebView** ab, bietet nach außen keinerlei neue APIs und verhält sich somit vollkommen transparent.

Hier ein Beispiel, um einen **IOLWebView** in einem XML-Layout zu erzeugen:

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

HINWEIS Der IOLWebView aktiviert automatisch JavaScript und DOM-Storage. Der IOLWebView funktioniert nur korrekt, wenn diese Einstellungen nicht geändert werden.

3.6 Debug-Informationen

Zum Zweck der allgemeinen Fehleranalyse und insbesondere des Versands der Messdaten kann die Library in einen Debug-Modus versetzt werden. In diesem Debug-Modus erzeugt die Library diverse Ausgaben im Logstrom (Konsole): Fehler, Warnungen, Infos, Events und Requests.

Diese Ausgaben können dann in der Konsole komfortabel gefiltert werden.

Um die IOLib in den DebugModus zu versetzen, wir die Methode **setDebugModeEnabled** aufgerufen:

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

Per default ist der Debug-Modus **deaktiviert**, es sei denn, der Debug-Modus wurde bei der Initialisierung explizit aktiviert. Das Vorgehen ist in Kapitel 3.4.1 (*Initialisierung*) beschrieben.

HINWEIS Bitte deaktivieren Sie den Debug-Modus vor der Veröffentlichung der App.

3.7 Android TV / Nexus Player

Die IOLib Android ist vollständig geeignet für die Messung einer Android App auf der Plattform “Android TV / Nexus Player”, da diese technisch auf dem gleichem Betriebssystem basiert wie andere Android Geräte.

Soll die IOLib Android unter Android TV / Nexus Player zum Einsatz kommen, so sind alle Integrationsschritte gemäß Kapitel 3.2 durchzuführen. Die Methoden zur Steuerung der IOLib sind ebenfalls, gemäß Kapitel 3.4, analog zu verwenden.

Auch die Hybrid-Messung gemäß Kapitel 3.5 kann wie beschrieben eingesetzt werden.

3.8 Fire OS

Die IOLib Android ist grundsätzlich geeignet zur Messung einer Android App auf dem Betriebssystem „Fire OS 5“. Diese Kompatibilität bezieht sich auf die Veröffentlichung einer nativen Android App unter Fire OS 5, nicht auf HTML5/WebApps.

Soll die IOLib Android unter Fire OS 5 zum Einsatz kommen, so sind alle Integrationsschritte gemäß Kapitel 3.2 durchzuführen. Die Methoden zur Steuerung der IOLib sind ebenfalls, gemäß Kapitel 3.4, analog zu verwenden.

Auch die Hybrid-Messung gemäß Kapitel 3.5 kann wie beschrieben eingesetzt werden.

HINWEIS Die Messung von Second Screens in Verbindung mit einer Fire TV App und dem Amazon Fling SDK ist aktuell nicht gewährleistet

4 Vorgaben zum Aufruf der Library

4.1 Allgemeines

Die Erfassung der App-Nutzung durch den Benutzer erfolgt, indem die App die SZM-Library bei definierten Ereignissen, welche eine Nutzer-Interaktion kennzeichnen, aufruft.
Die Nutzer-Interaktion wird als Event bezeichnet.

HINWEIS Die SZM-Library muss von der App bei Eintreten des Events explizit aufgerufen werden.

Weiterhin misst die IOLib bestimmte System- oder App-spezifische Werte automatisch. Zu diesem Zweck muss die Integration der IOLib Android exakt wie in Kapitel 3.2.2 |Pkt 8 beschrieben erfolgen.

4.1.1 Interpretation von Events als mobile PI

Die nachfolgend aufgeführten Vorgaben definieren, wie die SZM-Library im Kontext der SZM Mobile Applications Messung zu verwenden ist.

Aus technischer Sicht wird zwischen 2 Typen von Event unterschieden:

1. PI-Events

Bei den PI-Events wird der Event dazu benutzt, analog zum stationären Web, eine PageImpression zu erzeugen. Diesem Event muss ein Code zugeordnet werden. Dieser Code kann anschließend den unterschiedlichen Kategorien zugeordent werden und dient als Grundlage für die Bildung von Belegungseinheiten. Bei den PI-Events sind die Vorgaben zur Mobile Impression der IVW zu beachten:

Eine Mobile Impression ist eine Nutzeraktion innerhalb eines mobilen Angebots, die zu einem Aufruf eines Werbemittels führt oder führen könnte. Jede Nutzeraktion darf nur einmal gezählt werden. Nutzeraktionen, die zu keiner potentiellen Werbeauslieferung führen, dürfen nicht gezählt werden.

„Voraussetzungen für die Zuweisung einer MI zu einem Angebot:
Der ausgelieferte Inhalt muss (bei mobile enabled Websites) den FQDN bzw. (bei Apps) den App-Namen des Angebots (oder Alias/Redirect) oder den zugewiesenen MEW- oder App-Namen des Angebots tragen.

Nutzeraktion:
Eine MI wird ausgelöst durch eine vom Nutzer durchgeführte Aktion.
Darunter fallen ebenfalls: Reload, Öffnen einer App, Öffnen eines Browsers

Keine Nutzeraktion:
Aufruf eines Inhalts durch eine automatische Weiterleitung (außer Redirects und Alias)., automatischer Reload, das Aufrufen eines Inhaltes beim Schließen (auch: Background) eines Browserfensters oder einer App, das Aufrufen von Inhalten über Robots/Spider und Ähnliches.

Keine Mobile Impression:
Das Scrollen innerhalb eines bereits geladenen Inhalts.

2. Non-PI-Events

Non-PI-Events sind Nutzeraktionen, die als Event im SZM-System erfasst werden, jedoch nicht zur Zählung einer Mobile Impression führen. Diesem Event **darf kein Code** zugeordnet werden. Beispiele für Non-PI-Events sind

- automatisch von der IOLib erfasste Events
- vom Anbieter festgelegte Events, welche keine mobile PI darstellen und dennoch als Events gemessen werden sollen, um z.B. die Nutzung der APP durch die Nutzer besser nachvollziehen zu können.

4.2 Richtlinien zur Vergabe der Codes

Bei den **PI-Events** ist der Code als eindeutige Kennzeichnung des angezeigten Inhalts mitzugeben. Der Code wird vom App-Anbieter spezifiziert.

Bei der Spezifikation der Inhalts-Codes sind die Code-Richtlinien der INFOnline zu beachten

- Länge der Codes: Ein Code darf maximal 255 Zeichen enthalten
- Anzahl der Codes: es dürfen maximal 2000 Codes verwendet werden
- Erlaubte Zeichen: a-z, A-Z, 0-9; Komma „,“; Bindestrich „-“; Unterstrich „_“; Slash „/“

Eine ausführliche Dokumentation der INFOnline Code-Richtlinien finden Sie im „INFOnline Configuration Guide“ unter:

https://www.infonline.de/downloads

4.3 Events

Im den nachfolgenden Tabellen sind Events aufgeführt, welche in der Messung erhoben werden bzw. erhoben werden können. Unter welchen Umständen ein Event zu einer PageImpression führen kann, wird im Folgenden ebenfalls erläutert.

4.3.1 Automatisch durch die Library gemessene Events

Die nachfolgende Tabelle beschreibt die Events, bei denen die SZM-Library automatisch aufgerufen wird. Bei den Events handelt es sich um Nutzeraktionen, welche aus technischen Gründen erhoben werden, jedoch nicht zur Zählung einer Mobile Impression führen. Diesem Event muss kein Code zugeordnet werden.

Event	Beschreibung	Bemerkung
ApplicationStart, ApplicationEnterBackground, ApplicationEnterForeground, ApplicationCrashed	App-spezifische Event, z.B. Start der App, Beenden der App, Crash, etc.	EnterFore/Background: Eingehender Anruf, Push-Notification Alert, Timer Alarm, App geht in den Hintergrund, etc. Crash: Wird nur im Vordergrund geloggt!
InternetConnectionEstablished InternetConnectionLost InternetConnectionSwitchedInterface	Art der Konnektivität ändert sich	Verbindung aufgebaut bzw. verloren Wechsel von Mobile auf Wifi bzw. umgekehrt
WebViewInit	Hybrid-Messung wird aktiviert	

Die „automatisch durch die Lib gemessenen Events“ werden erfasst, sobald die Klasse „IOLActivitiy“ der MessLib als Basis der Activities der App verwendet wird oder alternativ die unter Kap. 3.2.6 b) beschriebene Methode verwendet wird.

4.3.2 PI-Events

Im Folgenden sind Events aufgeführt, welche typischerweise zur Auslösung einer PI führen. Die Übernahme des Schemas wird empfohlen. Die Events müssen manuell ausgelöst werden, eine automatische Erhebung erfolgt nicht. PI-Events muss ein Code zugeordnet werden. Dieser Code kann anschließend den unterschiedlichen Kategorien zugeordnet werden und dient als Grundlage für die Bildung von Belegungseinheiten.
HINWEIS Bei den PI-Events sind die Vorgaben zur Mobile Impression der IVW zu beachten.

Event	Beschreibung	Bemerkung
DeviceOrientation	Ausrichtung des Gerätes	LandscapeLeft / LandscapeRight oder Portrait / Portrait UpsideDown
GestureShake	Gerät wird geschüttelt	
ViewAppeared ViewRefreshed	Ein View (aka „Page“) wurde angezeigt oder mit neuen Daten aktualisiert	Beispiele: Appeared: initialer Aufruf einer Seite Refreshed: Suchfilter o. Aktualisierung von Daten
GameAction GameStarted	Gaming-Events	Aktion innerhalb eines Spiels Spiel gestartet
AudioPlay AudioPause AudioStop AudioNext AudioPrevious AudioReplay AudioSeekBack AudioSeekForward	Audio Playback	Wiedergabe, Pause, Stop, Nächster/vorheriger Titel, Vor/Zurückspulen, Wiederholung
VideoPlay VideoPause VideoStop VideoNext VideoPrevious VideoReplay VideoSeekBack VideoSeekForward	Video Playback	Wiedergabe, Pause, Stop, Nächster/vorheriger Titel, Vor/Zurückspulen, Wiederholung

4.3.3 Non-PI-Events

Im folgenden sind Events aufgeführt, welche typischerweise nicht zur Zählung einer PI führen. Sollten jedoch im Einzelfall Umstände vorliegen, unter denen auch hier eine Mobile PI erzeugt werden soll, können diese Events benuzt werden, um dies zu ermöglichen. Bevor Sie diese Events zur Erzeugung von PIs benutzen, klären Sie den jeweiligen Sachverhalt bitte unmittelbar mit der IVW-Geschäftsstelle ab. Sollen diese Events zur Zählung von PIs führen, muss das auch hier manuell ausgelöst werden. Diesem Event muss dann ein Code zugeordnet werden. Dieser Code kann anschließend den unterschiedlichen Kategorien zugeordnet werden und dient als Grundlage für die Bildung von Belegungseinheiten.

Event	Beschreibung	Bemerkung
ViewDisappeared	Ein View (aka „Page“) wurde verlassen	Beispiele: Disappeared: Screen verlassen
DocumentOpen DocumentEdit DocumentClose	Dokument / Liste Bearbeitung	Liste editiert Dokument gespeichert
DataCancelled DataRefresh DataSucceeded DataFailed	Datenverbindung/ -verarbeitung	Datenverbindung abgebrochen Daten wurden aktualisiert Daten wurden erfolgreich übertragen Daten wurden nicht übertragen
DownloadCancelled DownloadStart DownloadSucceeded DownloadFailed	Download von Daten	Download wurde initiiert Download wurde abgebrochen Download erfolgreich beendet Download fehlgeschlagen
UploadCancelled UploadStart UploadSucceeded UploadFailed	Upload von Daten	Upload wurde initiiert Upload wurde abgebrochen Upload erfolgreich beendet Upload fehlgeschlagen
LoginSucceeded LoginFailed LoginLogout	Login	Login erfolgreich durchgeführt Login fehlgeschlagen Logout durchgeführt/Session beendet
GameFinished GameWon GameLost GameNewHighscore GameNewAchievement	Gaming-Events	Spiel beendet Spiel(runde) gewonnen Spiel(runde) verloren Neuer Highscore erreicht Neues Achievement erreicht
HardwareButtonPushed	Schalter oder Knopf am Gerät gedrückt	z.B.Lautstärke über Schalterwippe am Gerät geändert, Mute-Schalter, Device gelocked, Home-Button, Back, Options-Menü, Suchen und Tasks
OpenAppMaps OpenAppOther	Eine andere APP wird gestartet bzw. APP wird über eine URL verlassen	Maps: Google Maps bzw Maps wird aufgerufen Other: andere APPs bzw. URIs werden aufgerufen (Email, Phone, Websites, etc.)

Event	Beschreibung	Bemerkung
PushReceived	Push Notifications	Push innerhalb der App empfangen (der Empfang von Push Nachrichten außerhalb der App wird nicht gemessen
IAPStarted IAPFinished IAPCancelled	In-App Billing / Kauf von digitalen Inhalten innerhalb einer App	Kaufvorgang initiiert Kaufvorgang (erfolgreich) abgeschlossen Kaufvorgang abgebrochen

Sobald in der App ein unter „manuell auszulösende Events“ beschriebenes Ereignis ausgelöst wird, ist die MessLibrary per logEvent aufzurufen. Hierbei ist die in der Spalte „Event“ gelistete Enums als Parameter zu übergeben.

Die Übermittlung der Non-PI-Events an das SZM System ist optional.
HINWEIS Die Non-PI-Events haben keinen Einfluss auf die Reichweitenermittlung Ihrer App.

Diese Events **können** von Ihnen zur quantitativen Ermittlung der Häufigkeit des Auftretens dieser Events (in den INFOnline Auswertesystemen) verwendet werden. Hierbei ist zu beachten, dass die Erfassung und Übermittlung der Non-PI-Events technische Ressourcen (CPU-Zeit, Netzwerkverkehr, Batterie) auf dem Endgerät verwendet.

Mit Hinblick auf die Inanspruchnahme der technischen Ressourcen auf dem Endgerät bitten wir Sie, in der Planung der Umsetzung der Integration der Mess-Libs zu entscheiden, ob ihre App Non-PI-Events an das SZM-System übermitteln soll.

4.4 Sonderfall Event „ViewRefreshed“

Für den Fall, dass ein Screen aktualisiert wird (Event IOLEventType.ViewRefreshed), ist Folgendes zu beachten:

HINWEIS Das Event darf nur geloggt werden (bzw. die SZM-Library aufgerufen werden) wenn der Refresh der Daten manuell durch den Nutzer ausgelöst wurde. Bei einem automatischen Refresh darf das Event nicht geloggt werden.

5 Kontakt

Das Service & Support-Team ist werktags von 9 bis 18 Uhr erreichbar via

Telefon:	0228 / 410 29 – 77
E-Mail für organisatorische Anfragen:	service@INFOnline.de
E-Mail für technische Anfragen:	support@INFOnline.de