INFOnline
Integration Guide

Platform: iOS
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 iOS operating system. **The app platform supported** here is **iOS ab Version 8.0**.

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 (iOS)” explains the limiting and framework conditions for the measuring instrument.
3. The “Integration of the SZM library iOS” 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 (iOS)

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

2.1 Provision

The IOLib iOS 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
- a directory **INFOnlineLibrary**: The IOLib is provided as a "framework" and can therefore easily be integrated into existing iOS projects. For this purpose the enclosed copy framework script can be used.
- a directory **ObjCSample**: a sample project with integrated IOLib on Objective-C basis
- a directory **Swift Sample**: a sample project with integrated IOLib on Swift 3.3 basis

2.2 Requirements

The SZM library for iOS only supports integration via the Xcode development environment under macOS.

2.2.1 Development environment

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

- macOS 10.12.4 (Sierra) and higher
- iOS SDK 11 and higher
- Xcode 9.0 and higher
- Objective-C or Swift

iOS apps that use the IOLib iOS must be compiled with iOS SDK.

Deployment Target must be set to at least 8.0.

2.2.2 Operating system/platform

The IOLib iOS supports operation under 32-bit and 64-bit architectures.

The IOLib iOS requires iOS 8.0 or higher to operate.

2.2.3 Compability

A new version of the Xcode as well as the iOS operating system may require an update to the SZM 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 (***Fehler! Verweisquelle konnte nicht gefunden werden.***).

2.3.2 Offline use

The IOLib iOS 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 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/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 3.3):

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 team.

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 **Fehler! Verweisquelle konnte nicht gefunden werden.** (***Fehler! Verweisquelle konnte nicht gefunden werden.***).

3 Integration of the SZM library iOS

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

3.1 Thread-safe

The IOLib for iOS is completely thread-safe.

3.2 IOLib files

The INFOnline SZM library for iOS comprises the following files

- **RELEASE_NOTES.txt**

  This file includes information about the releases of the IOLib.

- **CHANGE_LOG.txt**

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

- **INFOnlineLibrary**

  Contains the IOLib (INFOnlineLibrary.framework) for measuring the usage data of an app, as well as the copy-framework script for automatic integration of the framework into a project.

- **ObjCSample**

  A sample project demonstrating the use of the IOLibrary for iOS.
  The interaction with the IOLibrary is implemented with Objective-C.
  The project includes a TodayExtension.

- **SwiftSample**

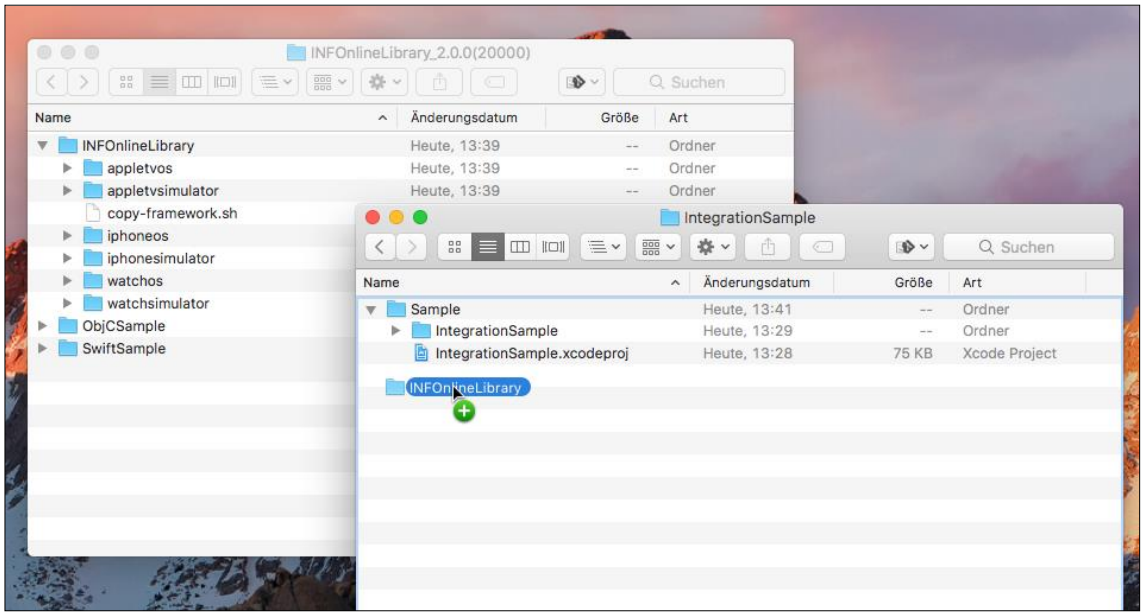
  A sample project demonstrating the use of the IOLibrary for iOS.
  The interaction with the IOLibrary is implemented with Swift 3.3.
  The project includes a TodayExtension.

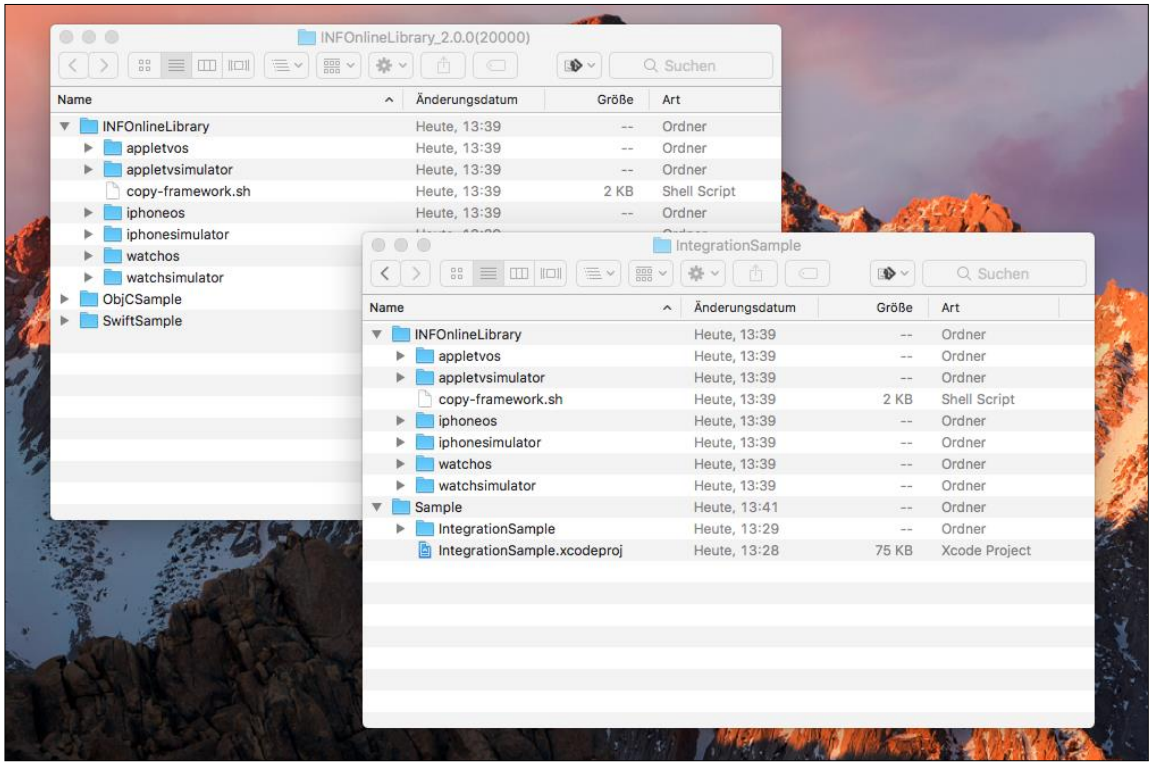
3.3 Integration of the IOLib iOS framework

Integration takes place in a few simple steps.

1. **In the finder:**

Copy the folder “INFOnlineLibrary” into the project folder (or drag it over using drag ’n’ drop)





2. In Xcode: Build Settings

Add the path to the "INFOnlineLibrary" folder to the **Framework Search Paths** in the Build Settings.

In our sample:

```
$(PROJECT_DIR)/../INFOnlineLibrary/$(PLATFORM_NAME)/
```

For the **Other Linker flags** in the Build Settings, the INFOnlineLibrary must be specified as follows:

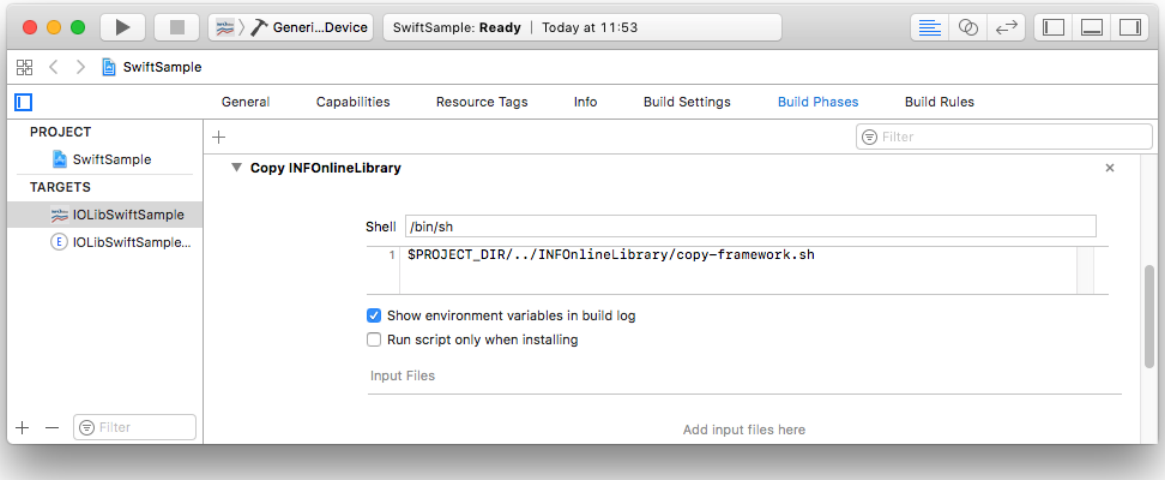
```
-framework INFOnlineLibrary
```

3. In Xcode: Run Script Build Phase

Add a Run Script phase to the Build Phases:

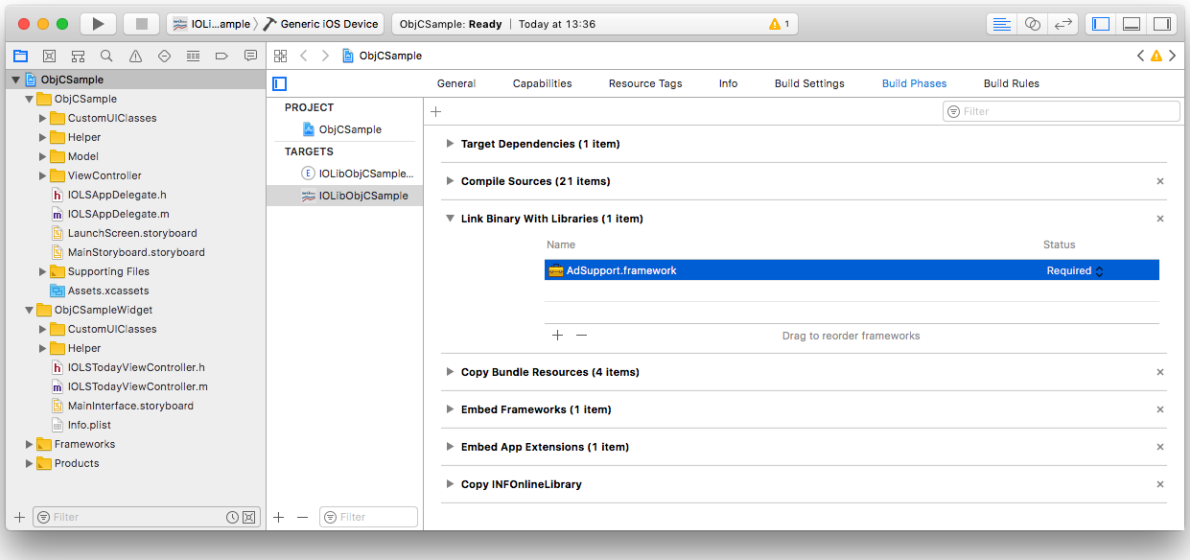
In our sample:

```
$(PROJECT_DIR)/../INFOnlineLibrary/copy-framework.sh
```



4. In Xcode:

Linking of the framework **AdSupport**



NOTE

The frameworks mentioned above must be integrated to ensure correct and complete measurement for participation in the agof study.

If you do not integrate the frameworks, the identifier required for the agof measurement (Advertising Identifier / IDFA) cannot be transferred - participation in the agof measurement is therefore not possible.

If you do not offer advertising space in the application and on the basis of the provisions of the Apple iTunes Connect Developer Guidelines, use of the advertising identifier is not possible, please contact INFOnline's Customer Service team on +49 (0) 228 4102977 or by e-mail to support@INFOnline.de.

You have the technical option to use the Vendor Identifier (IDVA) instead of the Advertising ID. To do so, please remove the linking of the AdSupport framework. As a consequence, the IOLib then records the Vendor Identifier (IDVA) instead of the Advertising Identifier (IDFA).

NOTE We recommend always implementing the Advertising Identifier (IDFA) in accordance with the documentation. Participation in the agof is not possible without using the Advertising Identifier!

5. **In Xcode:**

Objective-C:

Import of the IOLib header in the ApplicationDelegate and in the ViewControllers (alternatively in the prefix header)

```
#import <INFOnlineLibrary/INFOnlineLibrary.h>
```

NOTE The umbrella header should always be used, never individual headers from the framework!

Swift:

Import of the IOLib framework in the ApplicationDelegate and in the ViewControllers

```
import INFOnlineLibrary
```

6. In Xcode: Initialization and start of an IOLib session when the application is started:

Objective-C:

```
@implementation AppDelegate

- (BOOL)application:(UIApplication*)application
didFinishLaunchingWithOptions:(NSDictionary*)launchOptions {
    [self startSession];
    // Other code
    return YES;
}

- (void)startSession {
    // Initialisierung der IOLib; Session-Start
    // privacySetting LIN = Berechtigtes Interesse ist gegeben
    [[IOLSession defaultSessionFor:IOLSessionTypeSZM]
startSessionWithOfferIdentifier:@„<ANGEBOTSKENNUNG>" privacyType:IOLPrivacyTypeLIN];
}
```

```
//
//  AppDelegate.m
//  ObjCSample
//
//  Copyright (c) 2017 RockAByte GmbH. All rights reserved.
//

#import "AppDelegate.h"

#import <INFOnlineLibrary/INFOnlineLibrary.h>

NS_ASSUME_NONNULL_BEGIN

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary * _Nullable)launchOptions {
    [self startSessions];
    return YES;
}

#pragma mark - Helper

- (void)startSessions {
    [IOLLogging setDebugLogLevel:IOLDebugLevelVerbose];

    BOOL isOptOut = [[NSUserDefaults standardUserDefaults] boolForKey:@"IOLSOptOutKey"];
    if (!isOptOut) {
        [[IOLSession defaultSessionFor:IOLSessionTypeSZM] startSessionWithOfferIdentifier:@"szmOfferID" privacyType:IOLPrivacyTypeLIN];
    }
}

@end

NS_ASSUME_NONNULL_END
```

Swift:

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [UIApplicationLaunchOptionsKey : Any]? = nil) -> Bool {
  self.startSession()
  // Other code
  return true
}

func startSession() {
  // Initialisierung der IOLib; Session-Start
  // privacySetting LIN = Berechtigtes Interesse ist gegeben
  IOLSession.defaultSession(for: .SZM).start(withOfferIdentifier:„<ANGEBOTSKENNUNG>“,
  privacyType: .LIN)
}
```

```swift
//
//  AppDelegate.swift
//  SwiftSample
//
//  Copyright © 2017 RockAByte GmbH. All rights reserved.
//

import UIKit
import INFOnlineLibrary

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication,
                     didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey : Any]? = nil) -> Bool {
        startSessions()
        return true
    }

    // MARK: - Helper

    private func startSessions() {
        IOLLogging.setDebugLogLevel(.verbose)

        let isOptOut = UserDefaults.standard.bool(forKey: "IOLSOptOutKey")
        guard !isOptOut else { return }
        IOLSession.defaultSession(for: .SZM).start(withOfferIdentifier: "szmOfferID", privacyType: .LIN)
    }
}
```

7. In Xcode:

Events can be logged in the View Controllerd of the app, e.g. accessing a view:

Objective-C:

```
// Tracking View Appeared
    IOLViewEvent *event = [[IOLViewEvent alloc] initWithType:IOLViewEventTypeAppeared
category:@"TestCategory" comment:nil];
    [[IOLSession defaultSessionFor:IOLSessionTypeSZM] logEvent:event];
```

```
- (void)viewDidAppear:(BOOL)animated {
    [super viewDidAppear:animated];

    IOLViewEvent *event = [[IOLViewEvent alloc] initWithType:IOLViewEventTypeAppeared category:@"TestCategory" comment:nil];
    [[IOLSession defaultSessionFor:IOLSessionTypeSZM] logEvent:event];
}
```

Swift:

```
// Tracking View Appeared
    let event = IOLViewEvent(type: .appeared, category: "TestCategory", comment: nil)
    IOLSession.defaultSession(for: .SZM).logEvent(event)
```

```
    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)

        let event = IOLViewEvent(type: .appeared, category: "TestCategory", comment: nil)
        IOLSession.defaultSession(for: .SZM).logEvent(event)
    }
```

3.4 Parallel measurement SZM & ÖWA

The IOLib iOS 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.

Prerequisite is an integration of the IOLib iOS according to chapter 3.3 points 1-5

1. **In Xcode: Initialization and start of both sessions at application start:**

Objective-C:

```
@implementation AppDelegate


- (BOOL)application:(UIApplication*)application
didFinishLaunchingWithOptions:(NSDictionary*)launchOptions {
    [self startSessions];
    // Other code
    return YES;
}

- (void)startSessions {
    // Initialisierung der IOLib; Session-Start
    // privacySetting LIN = Berechtigtes Interesse ist gegeben
    [[IOLSession defaultSessionFor:IOLSessionTypeSZM]
      startSessionWithOfferIdentifier:@„<SZM-ANGEBOTSKENNUNG>"
      privacyType:IOLPrivacyTypeLIN];


    [[IOLSession defaultSessionFor:IOLSessionTypeOEWA]
      startSessionWithOfferIdentifier:@„<OEWA-ANGEBOTSKENNUNG>"
      privacyType:IOLPrivacyTypeLIN];
}
```

```
//
//  AppDelegate.m
//  ObjCSample
//
//  Copyright (c) 2017 RockAByte GmbH. All rights reserved.
//

#import "AppDelegate.h"

#import <INFOnlineLibrary/INFOnlineLibrary.h>

NS_ASSUME_NONNULL_BEGIN

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary * _Nullable)launchOptions {
    [self startSessions];
    return YES;
}

#pragma mark - Helper

- (void)startSessions {
    [IOLLogging setDebugLogLevel:IOLDebugLevelVerbose];

    BOOL isOptOut = [[NSUserDefaults standardUserDefaults] boolForKey:@"IOLSOptOutKey"];
    if (!isOptOut) {
        [[IOLSession defaultSessionFor:IOLSessionTypeSZM] startSessionWithOfferIdentifier:@"szmOfferID" privacyType:IOLPrivacyTypeLIN];
        [[IOLSession defaultSessionFor:IOLSessionTypeOEWA] startSessionWithOfferIdentifier:@"oewaOfferID" privacyType:IOLPrivacyTypeLIN];
    }
}

@end

NS_ASSUME_NONNULL_END
```

Swift:

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions
    launchOptions: [UIApplicationLaunchOptionsKey : Any]? = nil) -> Bool {
  self.startSessions()
  // Other code
  return true
}

func startSessions() {
  // Initialisierung der IOLib; Session-Start
  // privacySetting LIN = Berechtigtes Interesse ist gegeben
  IOLSession.defaultSession(for: .SZM).start(withOfferIdentifier:„<SZM-
  ANGEBOTSKENNUNG>“, privacyType: .LIN)
  IOLSession.defaultSession(for: .OEWA).start(withOfferIdentifier:„<OEWA-
  ANGEBOTSKENNUNG>“, privacyType: .LIN)
}
```

```
1  //
2  //  AppDelegate.swift
3  //  SwiftSample
4  //
5  //  Copyright © 2017 RockAByte GmbH. All rights reserved.
6  //
7
8  import UIKit
9  import INFOnlineLibrary
10
11 @UIApplicationMain
12 class AppDelegate: UIResponder, UIApplicationDelegate {
13
14     func application(_ application: UIApplication,
15                      didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey : Any]? = nil) -> Bool {
16         startSessions()
17         return true
18     }
19
20     // MARK: - Helper
21
22     private func startSessions() {
23         IOLLogging.setDebugLogLevel(.verbose)
24
25         let isOptOut = UserDefaults.standard.bool(forKey: "IOLSOptOutKey")
26         guard !isOptOut else { return }
27         IOLSession.defaultSession(for: .SZM).start(withOfferIdentifier: "szmOfferID", privacyType: .LIN)
28         IOLSession.defaultSession(for: .OEWA).start(withOfferIdentifier: "oewaOfferID", privacyType: .LIN)
29     }
30 }
```

2. Measuring an event

Events can be logged in the View Controllerd of the app, e.g. accessing a view:

Objective-C:

```
// Tracking View Appeared
   IOLViewEvent *event = [[IOLViewEvent alloc] initWithType:IOLViewEventTypeAppeared
   category:@"TestCategory" comment:nil];
   [[IOLSession defaultSessionFor:IOLSessionTypeSZM] logEvent:event];
   [[IOLSession defaultSessionFor:IOLSessionTypeOEWA] logEvent:event];
```

```objc
- (void)viewDidAppear:(BOOL)animated {
    [super viewDidAppear:animated];

    IOLViewEvent *event = [[IOLViewEvent alloc] initWithType:IOLViewEventTypeAppeared category:@"TestCategory" comment:nil];
    [[IOLSession defaultSessionFor:IOLSessionTypeSZM] logEvent:event];
    [[IOLSession defaultSessionFor:IOLSessionTypeOEWA] logEvent:event];
}
```

Swift:

```swift
// Tracking View Appeared
    let event = IOLViewEvent(type: .appeared, category: "TestCategory", comment: nil)
    IOLSession.defaultSession(for: .SZM).logEvent(event)
    IOLSession.defaultSession(for: .OEWA).logEvent(event)
```

```swift
    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)

        let event = IOLViewEvent(type: .appeared, category: "TestCategory", comment: nil)
        IOLSession.defaultSession(for: .SZM).logEvent(event)
        IOLSession.defaultSession(for: .OEWA).logEvent(event)
    }
```

3.5 SZM library functions

The IOLib for iOS offers the functions described below:

3.5.1 Starting the default session

All functions of the SZM Library described below must be called on the default session object. The sessiontype must be passed as parameter.

Parameter:

- **IOLSessionType (mandatory)**

The used session type. **IOLSessionTypeSZM** must be used for the SZM measurement!

Example:

Objective-C:

+(IOLSession*)defaultSessionFor:(IOLSessionType)sessionType;

NOTE: For the SZM measurement, **IOLSessionTypeSZM** must be passed as sessionType.

```objc
IOLSession *session = [IOLSession defaultSessionFor:IOLSessionTypeSZM];
```

Swift:

class func defaultSession(for sessionType: IOLSessionType) -> IOLSession

NOTE: For the SZM measurement, **.SZM** must be passed as sessionType.

```
let session = IOLSession.defaultSession(for: .SZM)
```

3.5.2 Start a session

NOTE: **The IOLib must be started before the events are recorded.** The site ID of the app as well as the privacy setting must be passed as parameters.

Parameter:

- **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.

- **Privacy setting (mandatory)**

The reason why measurement is performed. The possible values are fixed.

Example (here is is the site ID “iamtest” and the privacy setting „LIN“):

Objective C:

- (void)startSessionWithOfferIdentifier:(NSString*)offerID privacyType:(IOLPrivacyType)privacyType;

```
[session startSessionWithOfferIdentifier:@„iamtest“ privacyType:IOLPrivacyTypeLIN];
```

```
//
//  AppDelegate.m
//  ObjCSample
//
//  Copyright (c) 2017 RockAByte GmbH. All rights reserved.
//

#import "AppDelegate.h"

#import <INFOnlineLibrary/INFOnlineLibrary.h>

NS_ASSUME_NONNULL_BEGIN

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *
    _Nullable)launchOptions {
    [self startSession];
    return YES;
}

#pragma mark - Helper

- (void)startSession {
    [IOLLogging setDebugLogLevel:IOLDebugLevelVerbose];

    BOOL isOptOut = [[NSUserDefaults standardUserDefaults] boolForKey:@"IOLSOptOutKey"];
    if (!isOptOut) {
        [[IOLSession defaultSessionFor:IOLSessionTypeSZM] startSessionWithOfferIdentifier:@"szmOfferID"
            privacyType:IOLPrivacyTypeLIN];
    }
}

@end

NS_ASSUME_NONNULL_END
```

Swift:

func start(withOfferIdentifier offerIdentifier: String, privacyType: IOLPrivacyType)

```
session.start(withOfferIdentifier: „iamtest", privacyType: .LIN)
```

```swift
//  AppDelegate.swift
//  SwiftSample
//
//  Copyright © 2017 RockAByte GmbH. All rights reserved.
//

import UIKit
import INFOnlineLibrary

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions
                     launchOptions: [UIApplicationLaunchOptionsKey : Any]? = nil) -> Bool {
        startSession()
        return true
    }

    // MARK: - Helper

    private func startSession() {
        IOLLogging.setDebugLogLevel(.verbose)

        let isOptOut = UserDefaults.standard.bool(forKey: "IOLSOptOutKey")
        guard !isOptOut else { return }
        IOLSession.defaultSession(for: .SZM).start(withOfferIdentifier: "szmOfferID", privacyType: .LIN)
    }
}
```

3.5.3 Logging an event

The measurement data is recorded by means of the **logEvent** call. A previously initialized event must be passed.

Objective C:

```
- (void)logEvent:(IOLEvent*)event;
```

Swift:

```
func logEvent(_ event: IOLEvent)
```

To create an event, an initializer of the corresponding IOLEvent subclass must be called. Up to four parameters can be passed, three of which are optional.

Objective C:

```
- (IOL_xy_Event*)initWithType:(IOL_xy_EventType)type;
- (IOL_xy_Event*)initWithType:(IOL_xy_EventType)type category:(nullable
  NSString*)category comment:(nullable NSString*)comment;
- (IOL_xy_Event*)initWithType:(IOL_xy_EventType)type category:(nullable
  NSString*)category comment:(nullable NSString*)comment parameter:(nullable
  NSDictionary*)parameter;
```

Swift:

```
public init(type: IOL_xy_EventType) -> IOL_xy_Event
public init(type: IOL_xy_EventType, category: String?, comment: String?) ->
IOL_xy_Event
public init(type: IOL_xy_EventType, category: String?, comment: String?, parameter:
[String: String]?) -> IOL_xy_Event
```

The first two calls are convenience functions that call the latter internally. The missing values are then supplemented by **nil** or default values.

Some of the events are measured automatically by the IOLib. Further details can be found in chapter **Fehler! Verweisquelle konnte nicht gefunden werden.** (***Fehler! Verweisquelle konnte nicht gefunden werden.***).

Parameter:

- **EventType (mandatory)**

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 syntactic specifications are given in chapter **Fehler! Verweisquelle konnte nicht gefunden werden.** (***Fehler! Verweisquelle konnte nicht gefunden werden.***). 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 chapter **Fehler! Verweisquelle konnte nicht gefunden werden.** (***Fehler! Verweisquelle konnte nicht gefunden werden.***), 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 field. This field is not limited in length.
Transfer of this value is optional; if it is not defined, **nil** should be transferred.

- **Parameter (optional)**

A dictionary with freely definable additional information about the event. Key and value must be of type string, the maximum length is limited to 255 characters.
Transfer of this value is optional; if it is not defined, **nil** should be transferred.

Available events

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

- **IOLAdvertisementEvent**
  - IOLAdvertisementEventTypeOpen
  - IOLAdvertisementEventTypeClose
- **IOLAudioEvent**
  - IOLAudioEventTypePlay
  - IOLAudioEventTypePause
  - IOLAudioEventTypeStop
  - IOLAudioEventTypeNext
  - IOLAudioEventTypePrevious
  - IOLAudioEventTypeReplay
  - IOLAudioEventTypeSeekBack
  - IOLAudioEventTypeSeekForward
- **IOLBackgroundTaskEvent**
  - IOLBackgroundTaskEventTypeStart
  - IOLBackgroundTaskEventTypeEnd
- **IOLCustomEvent**
  - **no type**
  - **name** instead (freely definable string, limited to 255 characters)
- **IOLDataEvent**
  - IOLDataEventTypeCancelled
  - IOLDataEventTypeRefresh
  - IOLDataEventTypeSucceeded
  - IOLDataEventTypeFailed
- **IOLDeviceOrientationEvent**
  - **no type** *(IOLDeviceOrientationEventTypeOrientationChanged)*
- **IOLDocumentEvent**
  - IOLDocumentEventTypeOpen
  - IOLDocumentEventTypeEdit

- IOLDocumentEventTypeClose

- **IOLDownloadEvent**
  - IOLDownloadEventTypeCancelled
  - IOLDownloadEventTypeStart
  - IOLDownloadEventTypeSucceeded
  - IOLDownloadEventTypeFailed
- **IOLGameEvent**
  - IOLGameEventTypeAction
  - IOLGameEventTypeStarted
  - IOLGameEventTypeFinished
  - IOLGameEventTypeWon
  - IOLGameEventTypeLost
  - IOLGameEventTypeNewHighscore
  - IOLGameEventTypeNewAchievement
- **IOLGestureEvent**
  - **no type** *(IOLGestureEventTypeShake)*
- **IOLHardwareButtonEvent**
  - **no type** *(IOLHardwareButtonEventTypePushed)*
- **IOLIAPEvent**
  - IOLIAPEventTypeStarted
  - IOLIAPEventTypeFinished
  - IOLIAPEventTypeCancelled
- **IOLLoginEvent**
  - IOLLoginEventTypeSucceeded
  - IOLLoginEventTypeFailed
  - IOLLoginEventTypeLogout
- **IOLOpenAppEvent**
  - IOLOpenAppEventTypeMaps
  - IOLOpenAppEventTypeOther
- **IOLPushEvent**
  - **no type** *(IOLPushEventTypeReceived)*
- **IOLUploadEvent**
  - IOLUploadEventTypeCancelled
  - IOLUploadEventTypeStart

- IOLUploadEventTypeSucceeded
- IOLUploadEventTypeFailed

- IOLVideoEvent

- IOLVideoEventTypePlay
- IOLVideoEventTypePause
- IOLVideoEventTypeStop
- IOLVideoEventTypeNext
- IOLVideoEventTypePrevious
- IOLVideoEventTypeReplay
- IOLVideoEventTypeSeekBack
- IOLVideoEventTypeSeekForward

- IOLViewEvent

- IOLViewEventTypeAppeared
- IOLViewEventTypeRefreshed
- IOLViewEventTypeDisappeared

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

Examples:

- **IOLViewEvent / IOLViewEventTypeAppeared**

Objective-C:

```
@implementation ViewController

- (void)viewDidAppear:(BOOL)animated {
    [super viewDidAppear:animated];

    IOLEvent *event = [IOLViewEvent initWithState:IOLViewEventTypeAppeared
                                         category:@"Home"
                                          comment:nil];
    [[IOLSession defaultSessionFor:IOLSessionTypeSZM] logEvent:event];

    // Other Code ..
}
```

Swift:

```
class ViewController: UIViewController {

   override func viewDidAppear(_ animated: Bool) {
      super.viewDidAppear(animated)
      let event = IOLViewEvent(type: .appeared,
                               category: "Home"
                                comment: nil)
      IOLSession.defaultSession(for: .SZM).logEvent(event)

      // Other Code ..
   }
}
```

- **IOLViewEvent / IOLViewEventTypeRefreshed**

Objective-C:

```
@implementation ViewController

- (IBAction)refresh:(id)sender {
   IOLEvent *event = [IOLViewEvent initWithState:IOLViewEventTypeRefreshed
                                        category:@"Home"
                                         comment:@"AdBanner shown"];
    [[IOLSession defaultSessionFor:IOLSessionTypeSZM] logEvent:event];

   // Other Code ..
}
```

Swift:

```
class ViewController: UIViewController {

   @IBAction func refresh(sender: AnyObject) {
      let event = IOLViewEvent(type: .refreshed,
                               category: "Home"
                                comment: „AdBanner shown")
      IOLSession.defaultSession(for: .SZM).logEvent(event)

      // Other Code ..
   }
}
```

- **IOLAudioEvent / IOLAudioEventTypePlay**

Objective-C:

```
@implementation ViewController

- (IBAction)playMusic:(id)sender {
   IOLEvent *event = [IOLAudioEvent initWithState:IOLAudioEventTypePlay
                                         category:@"Audio"
                                          comment:@„Audio Playback"];
   [[IOLSession defaultSessionFor:IOLSessionTypeSZM] logEvent:event];

   // Other Code ..
}
```

Swift:

```
class ViewController: UIViewController {

   @IBAction func playMusic(sender: AnyObject) {
      let event = IOLAudioEvent(type: .play,
                                category: "Audio"
                                 comment: "Audio playback")
      IOLSession.defaultSession(for: .SZM).logEvent(event)

      // Other Code ..
   }
}
```

3.5.4 Sending the measurement data

- (void)sendLoggedEvents;

The IOLib controls sending of the measurement data independently and entirely transparently for the end user. **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.

Example:

Objective-C:

```
@implementation ViewController

- (IBAction)send:(id)sender {
   [[IOLSession defaultSessionFor:IOLSessionTypeSZM] sendLoggedEvents];
}
```

Swift:

```
class ViewController: UIViewController {

    @IBAction func send(sender: AnyObject) {
        IOLSession.defaultSession(for: .SZM).sendLoggedEvents()
    }
}
```

3.5.5 Terminate session

- (void)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!**

Example:

Objective-C:

```
@implementation ViewController

- (void)disableIOLSession {
    [[IOLSession defaultSessionFor:IOLSessionTypeSZM] terminateSession];
}
```

Swift:

```
class ViewController: UIViewController {

    func disableIOLSession() {
        IOLSession.defaultSession(for: .SZM).terminateSession()
    }
}
```

NOTE: The IOLib session must be restarted afterwards! The procedure is described in Chapter 3.5.2 (*Start a session*).

3.5.6 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 chapter 3.5.5 (*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.**

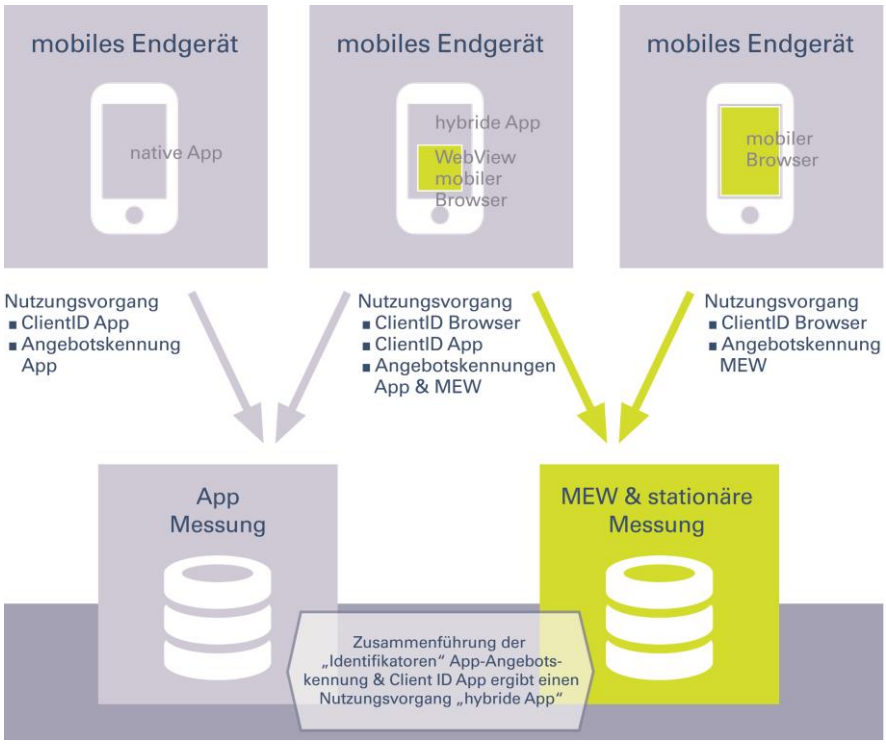
If the opt-out is revised, the measurement library should be restarted. The procedure is described in section 3.5.2 (*Start a session*).

3.6 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 in 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. For this purpose, an **IOLWKWebView** must be created instead of a standard **WKWebView** each time a WebView is initialized in the source code.

The view can then be treated as a regular **WKWebView**. The **IOLWKWebView** derives directly from the **WKWebView**, offers no new APIs to the outside and therefore behaves entirely transparently.

3.6.1 Support for UIWebView

The INFOnlineLibrary also supports the use of a UIWebViews. For this purpose, an **IOLUIWebView** must be created instead of a standard **UIWebView** each time a WebView is initialized in the source code.

If the **UIWebView** is created in the InterfaceBuilder, only the class name of the object must be changed from **UIWebView** to **IOLUIWebView**.

NOTE: From iOS 8 and higher Apple recommends using the WKWebView (WebKit) instead of the UIWebView. Likewise, we recommend using the IOLWKWebView instead of the IOLUIWebView.

3.7 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 SZM library generates various outputs in the log flow (console).

Objective-C:

```
- (void)applicationDidFinishLaunching:(UIApplication*)application {
    [IOLLogging setDebugLogLevel:IOLDebugLevelInfo];
}
```

Swift:

```
func applicationDidFinishLaunching(application: UIApplication) {
    IOLLogging.setDebugLogLevel(.info)
}
```

The type and extent of the log outputs can be determined via a DebugLevel.

The following debug levels are defined:

- **IOLDebugLevelOff**
  - Log output: **Deaktiviert (Default)**
- **IOLDebugLevelError**
  - Log output: error only
- **IOLDebugLevelWarning**
  - Log output: error and warnings
- **IOLDebugLevelInfo**
  - Log output: error, warnings and information

- **IOLDebugLevelTrace**
  - Log output: errors, warnings, information, events, requests and responses

NOTE Requests and responses are saved in the documents folder. For debugging purposes, file sharing can be switched on in iTunes and the corresponding files simply copied via iTunes.

4 Requirements for accessing the SZM library

4.1 Generel 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 characterize 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 iOS must therefore be carried out exactly as described in section 3.3 (*Integration of the IOLib iOS frameworkk*).

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 content code must be allocated to this event (hereafter simply referred to as a “code”). This code can subsequently 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 code: 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 SZM 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
IOLAccessoryEvent	Connected, Disconnected		
IOLApplicationEvent	Start, EnterBackground, EnterForeground, ResignActive, BecomeActive, Terminate, Crashed	App-specific event, e.g. start of the app, termination of the app, crash, etc.	ResignActive: Incoming call, push notification alert, timer alarm, app goes into the background, etc. EnterFore/Background: App goes into the background Terminate: App is terminated
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	

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. The procedure is described in section 3.5.3 (*Logging an event*). Automatic recording is not carried out. 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 With PI events, the IVW's specifications for mobile impressions must be observed.

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	Examples: 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	Audio playback	Play, pause, stop, next/previous title, rewind/fast forward, replay

	Stop Next Previous Replay SeekBack SeekForward		
IOLVideoEvent	Play Pause Stop Next Previous Replay SeekBack SeekForward	Video playback	Play, pause, stop, next/previous title, rewind/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”) was left	Beispiele: Disappeared: Screen left
IOLDocumentEvent	Open Edit Close	Document / list editing	Liste editiert Document saved
IOLDataEvent	Cancelled Refresh Succeeded Failed	Data connection/ processing	Data connection lost Data has been updated Data was transferred successfully Data was not transferred
IOLDownloadEvent	Cancelled Start Succeeded Failed	Download of data	Download was initiated Download was cancelled Download completed successfully Download failed
IOLUploadEvent	Cancelled Start	Upload of data	Upload was initiated Upload was cancelled

	Succeeded Failed		Upload completed successfully Upload 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 on device changed by rocker switch Device locked (power button 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	A push notification is received
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**. An instance of the corresponding IOLEvent subclass must be passed as parameter (see also chapter 3.5 (*SZM library*)).

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.

4.3.4 Special case event „ViewRefreshed“

If a view is refreshed (Event IOLEventTypeView, State IOLViewRefreshed), 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.

4.3.5 Special case of WatchKit Events

The pendant to the **UIViewController** is the **WKInterfaceController** class in the WatchKit. This class includes the two methods **willActivate** and **didDeactivate**, which represent the pendants to the **viewWillAppear** and **viewDidDisappear** methods from **UIViewController**.

However, in order to make recording of WatchKit events as easy as possible on the server side, the event type **IOLViewEvent** and its types **IOLViewEventTypeAppeared** and **IOLViewEventTypeDisappeared** should be used in the callback methods of the **WKInterfaceController** class.

Example (code in a sub-class of **WKInterfaceController**):

Objective-C:

```
26  - (void)willActivate {
27      // This method is called when watch view controller is about to be visible to user
28      [super willActivate];
29
30      IOLViewEvent *event = [[IOLViewEvent alloc] initWithType:IOLViewEventTypeAppeared
            category:NSStringFromClass([self class]) comment:nil];
31      [[IOLSession defaultSessionFor:IOLSessionTypeSZM] logEvent:event];
32  }
33
34  - (void)didDeactivate {
35      // This method is called when watch view controller is no longer visible
36      [super didDeactivate];
37
38      IOLViewEvent *event = [[IOLViewEvent alloc] initWithType:IOLViewEventTypeDisappeared
            category:NSStringFromClass([self class]) comment:nil];
39      [[IOLSession defaultSessionFor:IOLSessionTypeSZM] logEvent:event];
40  }
```

Swift:

```swift
    override func willActivate() {
        // This method is called when watch view controller is about to be visible to user
        super.willActivate()

        let event = IOLViewEvent(type: .appeared, category: self.classForCoder, comment: nil)
        IOLSession.defaultSession(for: .SZM).logEvent(event)
    }

    override func didDeactivate() {
        // This method is called when watch view controller is no longer visible
        super.didDeactivate()

        let event = IOLViewEvent(type: .disappeared, category: self.classForCoder, comment: nil)
        IOLSession.defaultSession(for: .SZM).logEvent(event)
    }
```

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