Mobile-SDK für iPhone/iPad

Inhalt

Einführung 4
SDK-Komponenten 4
Unterstützte iOS-Versionen 5
Das Starter-Package 5
Anforderungen 6
First Steps – Projekt-Setup und IDE-Konfiguration 6
Allgemeine Einstellungen 6
Xcode-Setup 6
Linked Libraries 6
Weitere Einstellungen 8
Linker Flags hinzufügen 8
Compiler 8
Hinweise zum Targeting und Debugging 9
Geräte- und Simulator-Targets 9
Geräte-Target wählen 10
Simulator-Target wählen 10
Integration und Implementierung des SDK 10
Hinzufügen des GuJ-SDK für den GuJAdView 10
GuJAdView-Methoden (Version 2.0) 10
GuJAdView mit GCD-Blöcken (Version 2.0) 11
GuJAdView mit mOceanBackfill 12
GuJAdView ohne Keywords 12
GuJAdView mit Keywords 13
Positionierung des GuJAdViews 13
Positionierung eines GuJAdViews mit CGPoint-Informationen 13
Positionierung mit Hilfe eines View-Replacements 13

Positionierung eines GuJAdViews mit Keywords 14
Deaktivieren des GEO-Location-Services für GuJAdView 15
Compilieren und starten 15
Notifications mit dem GuJAdViewControllerDelegate 15
Interstitialintegration 17
Notifications mit dem InterstitialViewDelegate 17
InterstitialView-Instanziierungen 18
Besonderheiten beim Startstitial 19
Zusätzliche Request-Parameter und -Header für GuJAdView definieren 20
Fehlermeldungen 21
Bekannte ORMMA-Issues 23
(ISSUE 9000.001) 23
(ISSUE 9000.002) 23
(ISSUE 9000.003) 23
Technische Ansprechpartner 24

Einführung

Die folgende Dokumentation enthält eine detaillierte Beschreibung zur Integration und Implementierung des iOS-Software-Development-Kits (kurz: SDK) von G+J EMS. Folgende Werbeformen und Features werden vom SDK unterstützt.

- Standardwerbeformen
- Interstitials
- Rich Media
- Keyword-Targeting
- Geo-argeting
- Custom-Targeting
- IP-Targeting
- Unique User/Frequency-Capping
- In-App-Browser

SDK-Komponenten

Das GuJBaseSDK: dieses enthält alle grundlegenden Funktionen für das Laden und Anzeigen von Werbung. Für das Ausspielen der Werbung kann darüber hinaus auf die nativen Funktionen des mobilen Endgerätes zurückgegriffen werden. Hierdurch können Location Based Services, Nachrichtenversand, Video und Audio, Kartenintegration uvm. für die Werbung verwendet werden.

Das ORMMA-Modul: erweitert das GuJBaseSDK und stellt das Ausspielen der Werbung nach dem ORMMA-Standard sicher.

Das GuJmOcean-Modul: erweitert das GuJ-ORMMA-SDK und ermöglicht das Ausspielen von Performance Kampagnen des optimobile Systems in der Werbeplatzierung. Weitere Informationen über mOcean siehe http://code.google.com/p/mocean-sdk-ios/

Das GuJ-Smartstream-SDK-Modul: erweitert das GuJmOcean-SDK und ermöglicht das Ausspielen von Video Interstitials als Backfill in der Werbeplatzierung.

Weitere Informationen über ORMMA siehe http://www.ormma.org

Das AdViewContext-SDK unterstützt ORMMA Level 1 und 2, das mOcean-SDK der Version 2.12, das Smartstream-Video-Ad-SDK. Das SDK weist ein spezielles Design auf, welches es ermöglicht mit Hilfe nur einer Zeile im Code verschiedene Werbemittel abzurufen.

Der **GuJAdViewContext**: fasst alle SDKs (Base SDK, ORMMA-SDK, mOcean-SDK und Smartstream-SDK) in einem Modul zusammen und erleichtert dem Entwickler die Integration der SDKs in einem Projekt.

Unterstützte iOS-Versionen

Das SDK unterstützt iOS Version 5.0 aufwärts. Es wurde erfolgreich auf folgenden Geräten und mit folgenden iOS Versionen getestet.

Hardware

- iPhone 3GS
- iPhone 4
- iPhone 4S
- iPhone 5
- iPhone 5S
- iPad 1
- iPad 2
- iPad 3
- iPad Air

iOS-Version

- iOS 5.x
- iOS 6.x
- iOS 7.x

Das Starter-Package

Das SDK-Starter-Paket enthält die folgenden Komponenten:

- libGuJAdViewContext.a für iOS
- libGuJAdViewContextSimulator.a für den Simulator
- ORMMAResourceBundle.bundle
- VideoAdLib.bundle für das Smartstream-Video-SDK
- Notwendiger Header-file: GuJAdViewContext.h
- Optionaler Header-file: GuJAdViewControllerDelegate.h

Die Libraries **libGuJAdViewContext.a** und **libGuJAdViewContextSimulator.a** sind sogenannte „fat“ Libraries. Dies bedeutet, dass die Inhalte verschiedener SDKs und Architekturen (armv7, armv7s, i386) zu einer „großen“ Library zusammengefasst wurden.

Im Falle der libGuJXAdViewContext* Library sind folgende SDKs zusammengefasst:

- libGuJBaseSDK
- libGuJORMMASDK
- libGuJmOceanSDK
- libGuJXAXSISSDK
- libAdMobileSDK (externes mOcean-SDK)
- libAdSDKLib (externes Smartstream-SDK)

Anforderungen

Das SDK wurde in Xcode 4.6.2 kompiliert. Wenn Sie mit einer anderen Xcode Version arbeiten, müssen Sie sicherstellen, dass der Apple LLVM 4.2 Compiler verfügbar ist.

Das SDK ist mit ARC (Automatic Reference Counting) kompiliert worden. Dies bedeutet, dass es notwendig ist das **-fobj-arc Flag** in den Compiler Einstellungen zu setzen wenn eine Kompilierung ohne ARC geplant ist.

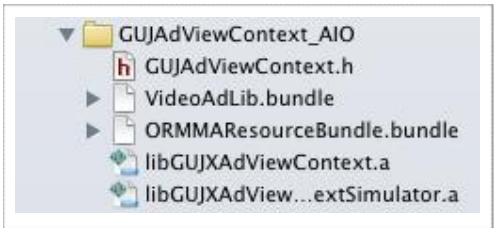
First Steps – Projekt-Setup und IDE-Konfiguration

Allgemeine Einstellungen

Xcode-Setup

Um das SDK integrieren zu können, müssen zunächst alle Dateien des Starterpakets zum Projektordner hinzugefügt werden.

Um einen besseren Überblick zu haben, empfiehlt es sich die Dateien entsprechend Ihrer Zugehörigkeit wie folgt zu gruppieren:

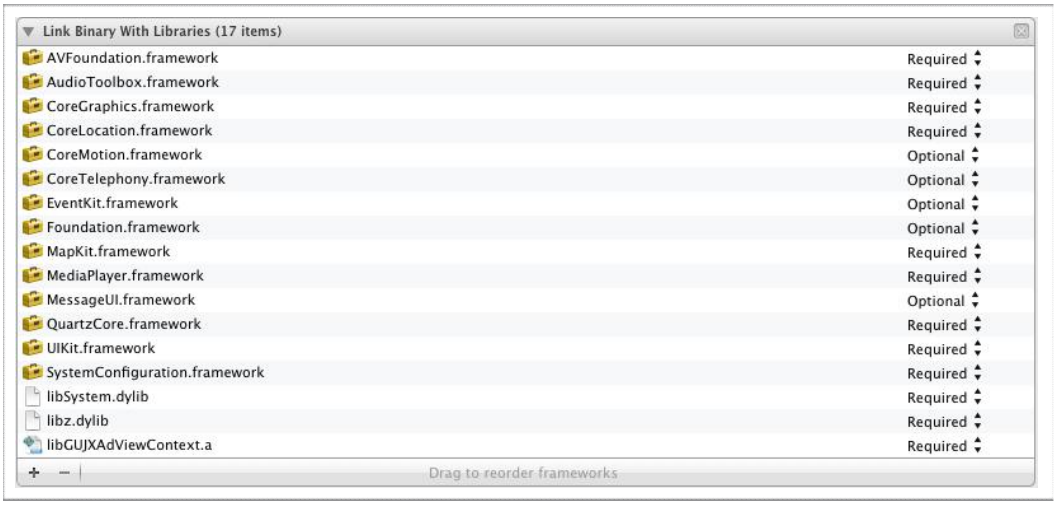


Linked Libraries

ACHTUNG: Dies ist einer der wichtigsten Schritte während der Integration. Bitte achten Sie exakt darauf, dass wirklich alle Libraries und Frameworks dem Projekt hinzugefügt werden!

Stellen Sie sicher, dass alle Libraries unter „Link Binary With Libraries“ unter dem „Build Phase“ Tab des Projektes aufgeführt sind.

- Öffnen Sie den Projektnavigator
- Öffnen Sie das Projekt
- Wählen Sie „Build Phase“.
- Expandieren Sie „Link Binary With Libraries“



Fügen Sie folgende iOS-Frameworks hinzu, um eine Multimediaunterstützung zu gewährleisten. Diese Frameworks sind unabdingbar.

Framework	verwendet von
AVFoundation.framework	ORMMA-SDK, mOcean-SDK, Smartstream-SDK
AudioToolbox.framework	Smartstream-SDK
CoreGraphics.framework	mOcean-SDK
CoreLocation.framework	ORMMA-SDK, mOcean-SDK, Smartstream-SDK
CoreMotion.framework	mOcean-SDK
CoreTelephony.framework	mOcean-SDK, Smartstream-SDK
EventKit.framework	ORMMA-SDK, mOcean-SDK
MapKit.framework	ORMMA-SDK, mOcean-SDK
MediaPlayer.framework	ORMMA-SDK, mOcean-SDK, Smartstream-SDK
MessageUI.framework	ORMMA-SDK, mOcean-SDK
QuarzCore.framework	mOcean-SDK
SystemConfiguration.framework	mOcean-SDK, Smartstream-SDK
libSystem.dylib	mOcean-SDK, Smartstream-SDK (<iOS 4.0)
libz.dylib	Smartstream-SDK

Weitere Einstellungen

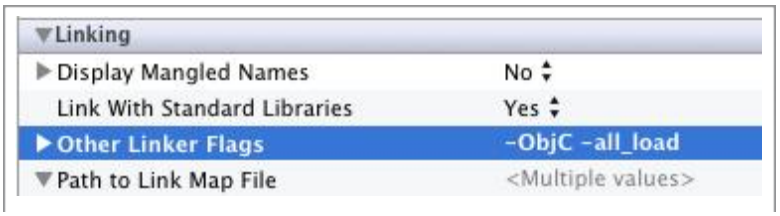
Linker Flags hinzufügen

Wenn Ihr Projekt kein ARC verwendet, vergessen Sie nicht, das -fobj-arc Flag den GuJ Libraries hinzuzufügen.

- Öffnen Sie den Projektnavigator
- Öffnen Sie das Projekt
- Wählen Sie „Build Settings“.
- Scrollen Sie bis „Linking“ Optionen herunter.

Unabhängig von den Projekt-Targets und dem Projekt-Setup sollten Sie die Compiler Flags -ObjCand -all_load zu „Other Linker Flags“ hinzufügen.

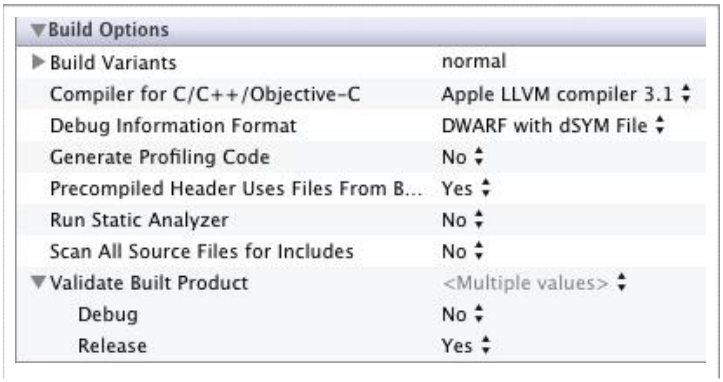
- Öffnen Sie den Projektnavigator
- Öffnen Sie das Projekt
- Wählen Sie „Build Settings“.
- Scrollen Sie bis „Linking“ Optionen herunter.



Compiler

Stellen Sie sicher, dass Sie mit dem Apple >= LLVM 4.2 Compiler compilieren.

- Öffnen Sie den Projektnavigator
- Öffnen Sie das Projekt
- Wählen Sie „Build Settings“.
- Scrollen Sie bis zu „Build Options“ herunter.



Hinweise zum Targeting und Debugging

Stellen Sie sicher, dass Ihr Kompatibilitätsziel >= iOS 5.0 ist.

Geräte- und Simulator-Targets

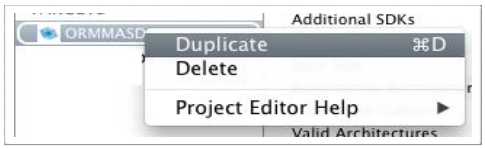
Wenn Sie das Testing mit Hilfe des iOS-Simulators planen, müssen Sie zunächst die Simulator-Libraries Ihrem Projekt hinzufügen.

Ebenso ist es auch möglich zwei verschiedene Targets zu wählen.

- Öffnen Sie den Projektnavigator
- Öffnen Sie das Projekt

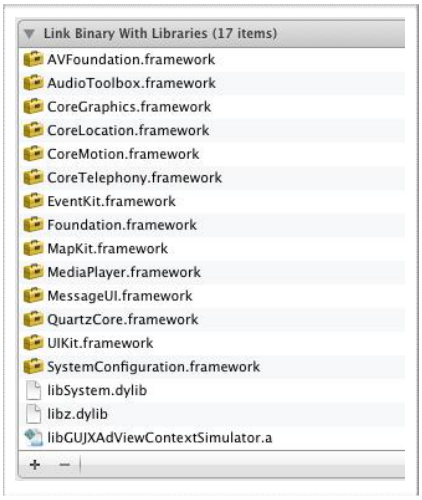
Um eine einfache Geräte- und Simulator-Testumgebung aufzusetzen, gehen Sie wie folgt vor:

- Erstellen Sie ein Geräte-Target
- Kopieren Sie das Device-Target zu den Simulator-Targets.

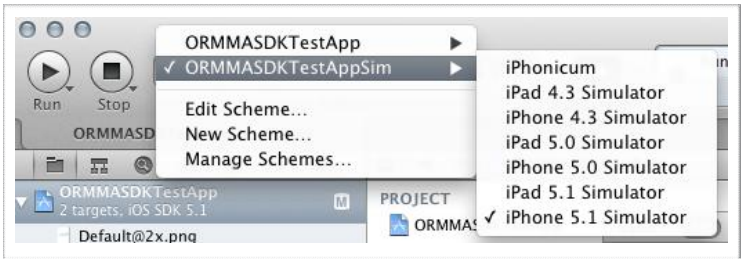


Entfernen Sie die Geräte-Libraries von dem Simulator-Target und fügen Sie die Libraries mit dem „Simulator“ suffix im Namen hinzu.

- Öffnen Sie den Projektnavigator
- Öffnen Sie das Projekt
- Wählen Sie „Build Phase“.
- Expandieren Sie „Link Binary With Libraries“



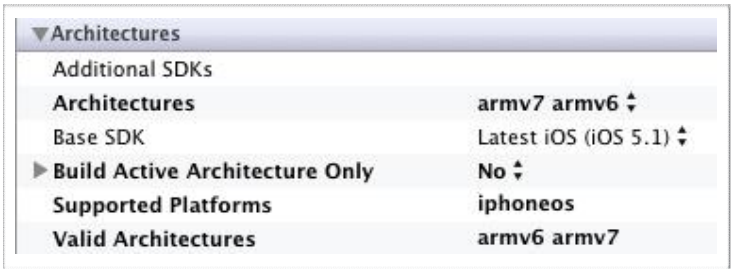
Starten Sie die unterschiedlichen Targets durch das jeweilige Selektieren des „Scheme“-Eintrags im Xcode-Dropdown.



Darüber hinaus können Sie hier auch die „Build Settings“ editieren.

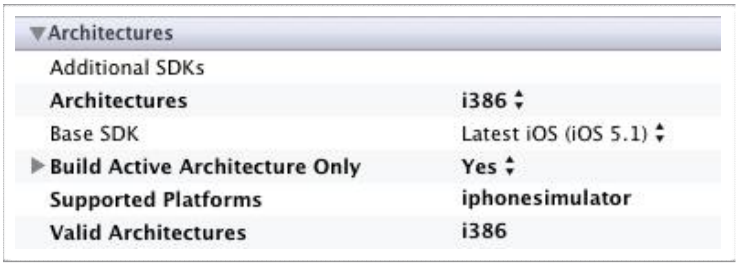
Geräte-Target wählen

- Öffnen Sie den Projektnavigator
- Öffnen Sie das Projekt
- Wählen Sie „Build Settings“.
- Wählen Sie ein Geräte -Target
- Scrollen Sie zum Punkt „Architectures“.



Simulator-Target wählen

Vorgehen siehe “Geräte-Target wählen”, nur dass in diesem Fall ein Simulator-Target gewählt wird.



Integration und Implementierung des SDK

Hinzufügen des GuJ-SDK für den GuJAdView

Importieren Sie **GuJAdViewContext.h** in Ihren **UIViewController** oder in Ihr **UIView Headerfile**.

GuJAdView-Methoden (Version 2.0)

Mit der Version 2.0 des SDK ist es nun möglich das **GuJAdView** über folgende Methoden zu steuern:

```
-(void)show;
```
Zeigt das erfolgreich geladene **GuJAdView** unverzüglich an.

```
-(void)showInterstitialView;
```
Zeigt das erfolgreich geladene **GuJAdView Interstitial** unverzüglich an.

```
-(void)hide;
```
Blendet das **GuJAdView** (auch Interstitial) unverzüglich aus.

Voraussetzung ist, dass das **GuJAdView** (nicht beim Interstitial) zuvor vom Entwickler als SubView eines Parent-View implementiert wurde.

Zusätzlich wurde das **GuJAdView** um eine Referenz auf die übergebene AdSpaceID erweitert. Diese ist nützlich für Debugzwecken oder Identifizierung mehrerer AdViews.

```
-(NSString*)adSpaceId;
```

GuJAdView mit GCD-Blöcken (Version 2.0)

Mit der Version 2.0 des GuJ-SDKs ist es möglich den **GuJAdView** mit einem Grand-Central-Dispatch-Block (GCD-Block) aufzurufen. Hierzu wurde folgender Typ in dem Header-File **GuJAdViewContext.h** definiert:

```
typedef BOOL (^adViewCompletion)(GuJAdView* _adView, NSError* _error);
```

Der Entwickler hat innerhalb des Blocks die Möglichkeit auf das vom **GuJAdViewContext** geladene GuJAdView (_adView) und einer möglichen Fehlermeldung (_error) zuzugreifen.

Es ist zwingend notwendig einen boolesche Wert innerhalb des Blocks als Rückgabewert zu definieren. Dieser boolesche Wert trifft die Aussage darüber, ob das geladene **GuJAdView** unverzüglich angezeigt wird oder nicht:

- Gibt der Entwickler YES(true/1) zurück wird das **GuJAdView** angezeigt.
- Im Falle von NO(false/0) wird das **GuJAdView** nicht angezeigt und der Entwickler muss in einen späteren Programmierungsschritt dafür Sorge tragen, dass das **GuJAdView** angezeigt wird.

Beispiel:

```
[[GuJAdViewContext instanceForAdspaceId:@"12345"] adView:^BOOL(GuJAdView
*_adView, NSError *_error) {
        if( _error != nil ) {
            NSLog(@"AdViewError: %@",_error);
            return NO;
        } else {
            [self.view addSubview:_adView];
            return YES;
        }
}];
```

Folgenden AdView-Aufrufe wurden zusätzlich mit diesem Block-Ausdruck versehen.

- (void)adView:(adViewCompletion)completion;
- (void)adViewWithOrigin:(CGPoint)origin completion:(adViewCompletion)completion;
- (void)adViewForKeywords:(NSArray*)keywords completion:(adViewCompletion)completion;
- (void)adViewForKeywords:(NSArray*)keywords origin:(CGPoint)origin completion:(adViewCompletion)completion;
- (void)interstitialAdViewWithCompletionHandler:(adViewCompletion)completion;

(!)Achtung:

Die Delegate-Methode **adViewController:canDisplayAdView** wird bei der Verwendung von GCD-Blöcken NICHT ausgeführt.

GuJAdView mit mOceanBackfill

Um das mOceanBackfill zu aktivieren, übergeben Sie bitte bei der Initialisierung des **GuJAdViewContext** eine gültige mOcean-Site- und Zone-ID.

```
[GuJAdViewContext instanceForAdspaceId:@"AdSpaceId" site:1234 zone:1234]
Oder mit delegate:
[GuJAdViewContextinstanceForAdspaceId:@"AdSpaceId" site:1234 zone:1234
delegate:self]
```

Das **mOceanBackfill** wird automatisch aktiviert, wenn eine Site- und Zone-ID übergeben wurde. Über folgenden Aufruf können Sie das mOceanBackfill manuell deaktivieren/aktivieren. Falls keine mOcean Site- und Zone-ID übergeben wurde, hat der Aufruf dieser Methode keinen Effekt.

```
Aktivieren:
[myAdViewContext setMOceanBackFill:YES]
Deaktivieren:
[myAdViewContext setMOceanBackFill:NO]
```

GuJAdView ohne Keywords

Instanziieren Sie den **GuJAdViewContext**, und fügen Sie den AdContainer der viewDidLoad Methode Ihres ViewControllers hinzu.

```
[self.viewaddSubview:
[[GuJAdViewContex tinstanceForAdspaceId:@"AdSpaceId"] adView]
];
> OR GCD BLOCK:
[self.viewaddSubview:[[GuJAdViewContext instanceForAdspaceId:@"AdSpaceId"]
adView:^BOOL(GuJAdView *_adView, NSError *_error) {
        return YES;
    }];
];
```

GuJAdView mit Keywords

Der folgende Beispielcode zeigt wie Sie einen AdView inklusive Keywords aufsetzen können. Diese Keywords werden beim Adrequest mit übergeben und als Targeting Option verwendet. Der AdServer prüft hierbei die eingestellten Keywords einer Kampagne und gleicht diese mit dem übermittelten Keyword des AdViews ab.

```
[[GuJAdViewContextinstanceForAdspaceId:@"AdSpaceId"] adViewForKeywords:[NSArray
arrayWithObjects:@"key1",@"key2", nil]];
> OR GCD BLOCK:
[[GuJAdViewContext instanceForAdspaceId:@"AdSpaceId"] adViewForKeywords:[NSArray
arrayWithObjects:@"key1",@"key2", nil] completion:^BOOL(GuJAdView *_adView,
NSError *_error) {
        return YES;
}];
```

Positionierung des GuJAdViews

Positionierung eines GuJAdViews mit CGPoint-Informationen

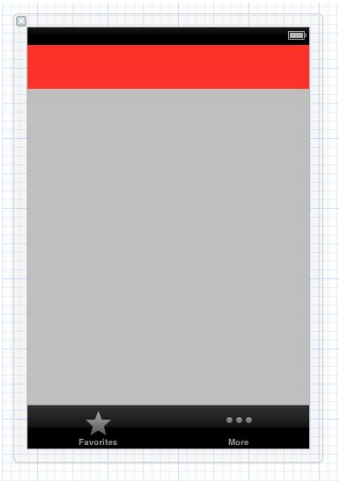
Sie können einen CGPoint dem GuJAdViewContext übergeben und hierdurch Zugriff auf die Position des Ads bekommen.

Der X-Wert des CGPoints wird ignoriert, da der Container automatisch vom SDK zentriert wird. Der Y-Wert des CGPoints wird dem AdView übergeben.

```
[[GuJAdViewContextinstanceForAdspaceId:@"AdSpaceId"]
adViewWithOrigin:CGPointMake(0.0f, 20.0f)];
> OR GCD BLOCK:
[[GuJAdViewContext instanceForAdspaceId:@"AdSpaceId"]
adViewWithOrigin:CGPointMake(0.0f, 20.0f) completion:^BOOL(GuJAdView *_adView,
NSError *_error) {
        return YES;
}];
```

Positionierung mit Hilfe eines View-Replacements

Beispiel eines AdViews mit View-Template im XCodeInterfaceBuilder Mit der Verwendung des XcodeInterfaceBuilders kann der Entwickler in einem Interface-File einen UIView als Template für den GuJAdView verwenden.



Der rote Bereich soll die spätere Position des AdView markieren

Der folgende Code zeigt, wie dieser UIView in einen GuJAdView gewandelt wird ohne die Position im Interface zu verlieren. Die Variable „containerView“ referenziert das UIView Template (roter Kasten) im InterfaceBuilder.

```
- (void)_loadAdview
{
adViewCTX_ = [GuJAdViewContextinstanceForAdspaceId:@"12345" delegate:self];
CGRect frame = containerView.frame;
[containerView removeFromSuperview];
containerView = [adViewCTX_ adViewWithOrigin:CGPointMake(0.0, frame.origin.y)];
[self.view addSubView:containerView];
> OR GCD BLOCK:
[[GuJAdViewContext instanceForAdspaceId:@"AdSpaceId"]
adViewWithOrigin:CGPointMake(0.0f, frame.origin.y) completion:^BOOL(GuJAdView
*_adView, NSError *_error) {
    [self.view addSubview:_adView];
    return YES;
}];
}
```

Positionierung eines GuJAdViews mit Keywords

Der folgende Beispielcode zeigt die Implementierung eines AdViews mit definierten Keywords und einer gewünschten Positionierung. Weitere Details zur Positionierung siehe “Positionierung des AdViews”.

```
[[GuJAdViewContextinstanceForAdspaceId:@"AdSpaceId"] adViewForKeywords:[NSArray
arrayWithObjects:@"key1",@"key2", nil] origin:CGPointMake(0.0f, 20.0f)];
> OR GCD BLOCK:
[[GuJAdViewContext instanceForAdspaceId:@"AdSpaceId"] adViewForKeywords:[NSArray
arrayWithObjects:@"key1",@"key2", nil] origin:CGPointMake(0.0f, 20.0f)
completion:^BOOL(GuJAdView *_adView, NSError *_error) {
        [self.view addSubview:_adView];
        return YES;
}];
```

Deaktivieren des GEO-Location-Services für GuJAdView

Dem Entwickler ist die Möglichkeit gegeben, die Standortermittlung bei Bedarf ein oder aus zu schalten. Zuvor muss ein **GuJAdViewContext** als Property initialisiert sein.

(!) Achtung: Mit Version 2.0 sind die Methoden enable- und disableLocationService nicht mehr statisch.

```
Aktivieren:
[myAdViewContext enableLocationService]
Deaktivieren:
[myAdViewContext disableLocationService]
```

Compilieren und starten

Wenn Sie den Code kompiliert und gestartet haben, werden Sie die Werbung in Top-Position in Ihrem UIView angezeigt bekommen.

Merke: Wenn Sie sich für diesen Weg der Implementierung entscheiden, ist es nicht möglich den aktuellen Ad-Status zu analysieren. Dies ist darauf zurückzuführen, dass es nicht möglich ist festzustellen, ob ein Ad geladen ist oder nicht. Desweiteren sind Sie nicht in der Lage jegliche Art von Fehlern während des Ladens zu identifizieren.

Notifications mit dem GuJAdViewControllerDelegate

Das GuJAdViewControllerDelegate kommuniziert mit dem AdView und übergibt verschiedene Controller-Messages an die App. Darüber hinaus ermöglicht es die Kontrolle des Ads auf viele verschiedene Weisen.

In der Regel wird durch die Verwendung der GCD-Blöcke eine Vielzahl der Delegate-Methoden optional, da der Entwickler innerhalb der Blöcke äquivalente Informationen zu den Delegate-Methoden zur Verfügung gestellt bekommt. Jedoch ist es ratsam - gerade beim Debugging - die Delegates zu verwenden.

Um die Delegate-Methoden verwenden zu können, fügen Sie Ihrem Projekt den optionalen Header-File „**GuJAdViewControllerDelegate.h**“ hinzu und implementieren Sie im Header-File der entsprechenden Klasse, welche die Delegate-Nachrichten erhalten soll, das Protokoll wie folgt:

```
@interface ViewController : UIViewController<GuJAdViewControllerDelegate>
```

Wenn Sie das **GuJAdViewControllerDelegate** Protokoll implementieren, müssen Sie den **GuJAdViewController** mit dem ‚delegate' instanziieren, um DelegateBenachtrichtigungen zu erhalten.

```
[GuJAdViewContextinstanceForAdspaceId:@"AdSpaceId" delegate:self]
```

Darüber hinaus ist es notwendig folgende Delegate-Methoden Ihrer UIViewController oder implementierenden Klasse hinzuzufügen.

```
- (void)adViewController:(GuJAdViewController*)adViewController
didConfigurationFailure:(NSError*)error;
- (void)bannerView:(GuJAdView*)bannerView didFailLoadingAdWithError:(NSError*)error;
- (void)interstitialView:(GuJAdView*)interstitialView
didFailLoadingAdWithError:(NSError*)error;
```

Zusätzlich können Sie auch noch diese Methoden bei Bedarf implementieren.

```
- (BOOL)adViewController:(GuJAdViewController*)adViewController
canDisplayAdView:(GuJAdView*)adView;
- (void)bannerViewInitialized:(GuJAdView*)bannerView;
- (void)bannerViewDidShow:(GuJAdView*)bannerView;
- (void)bannerViewDidHide:(GuJAdView*)bannerView;
- (void)bannerViewWillLoadAdData:(GuJAdView*)bannerView;

- (void)bannerViewDidLoadAdData:(GuJAdView*)bannerView;

- (void)bannerView:(GuJAdView*)bannerView receivedEvent:(GuJAdViewEvent*)event;
```

Wenn der adViewController nicht hinreichend konfiguriert ist, wird die Delegate-Methode **adViewController:didConfigurationFailure:** angesprochen. Sie erhalten detaillierte Informationen zum Fehler/Problem durch die Analyse des NSError. NSError ist ein Parameter dieser Methode. Zumeist wird dieser Fehler durch eine falsche oder fehlende AdSpaceID ausgelöst.

Wenn ein Fehler während des Ladens des Ads auftritt wird die Delegate-Methode **bannerView:didFailLoadingAdWithError:** angesprochen.

Optional haben Sie noch folgende Möglichkeiten um Probleme zu identifizieren:

- Das Ad erneut laden
- Den AdViewController löschen und erneut integrieren
- Nichts tun

Wenn das GuJAdView vollständig geladen wurde, keine GCD-Blocks verwendet werden und das GuJAdView kurz vor der Sichtbarkeit steht, wird die Delegate-Methode **(BOOL)adViewController:canDisplayAdView** ausgeführt. In dieser Methode ist es dem Entwickler möglich, die Darstellung zu steuern. Sollte die Delegate-Methode einen positiven, booleschen Wert (YES/true/1) zurückgeben, wird das GuJAdView unverzüglich angezeigt. Bei

einem negativen Wert (NO/false/0) wird das AdView nicht angezeigt und kann zu einem späteren Zeitpunkt durch den Aufruf [adView show] durch den Entwickler angezeigt werden.

Wenn der AdView geladen und als SubView Ihrer App hinzugefügt wurde, wird die Delegate-Methode **bannerViewInitialized:** angesprochen. Diese ermöglicht es Ihnen, mehr Informationen über das AdView zu sammeln. Diese Informationen sind mit Hilfe des Parameters bannerView der Methode abzurufen.

Sobald der Adview damit begonnen hat das Ads zu laden, wird die Delegate-Methode **bannerViewWillLoadAdData:** angesprochen. Dies bedeutet, dass eine Verbindung zum AdServer aufgebaut wird.

Wenn das Ad in Ihrem AdView vollständig und korrekt geladen wurde, wird die Delegate-Methode **bannerViewDidLoadAdData:** angesprochen. Hierdurch können Sie überprüfen, ob es Probleme beim Laden des Ads gab oder nicht. Wenn ein Fehler aufgetreten ist wird diese Methode nicht angesprochen.

Wird das Ad im ViewController angezeigt, wird die Delegate-Methode **bannerViewDidShow:** angesprochen. Hier können Sie auf die Sichtbarkeit des Ads reagieren.

Wird das Ad im ViewController ausgeblendet, wird die Delegate-Methode **bannerViewDidHide:** angesprochen. In diesem Fall können Sie auf die Unsichtbarkeit des Ads reagieren (z.B. den neuentstandenen Platz mit anderen UI-Elementen belegen).

Wenn das Ad ein Event oder eine Systemnachricht empfängt, wird diese an die Delegate-Methode **bannerView:receivedEvent:** weitergeleitet. So werden beispielsweise Informationen von Drittanbieter-SDKs (in diesem Fall mOcean und XAXIS) an die Delegate-Klasse weitergeleitet und dem Entwickler zugänglich gemacht.

Interstitialintegration

Notifications mit dem InterstitialViewDelegate

Ein Interstitial wird beim Wechsel zwischen zwei Views angezeigt. Um den Einstiegs- und Ausstiegspunkt für die Anzeige des Interstitials kontrollieren zu können, müssen folgende Delegate-Methoden implementiert werden. Andernfalls werden Sie nicht in der Lage sein, von einem View zum Nächsten zu gelangen, wenn ein Interstitial angezeigt wird. Fügen Sie deshalb folgende GuJAdViewControllerDelegate-Methoden Ihrem UIViewController hinzu:

```
- (void)interstitialViewInitialized:(GuJAdView*)interstitialView;
- (void)interstitialViewWillLoadAdData:(GuJAdView*)interstitialView;
- (void)interstitialViewDidLoadAdData:(GuJAdView*)interstitialView;
- (void)interstitialViewWillAppear;
- (void)interstitialViewDidAppear;
- (void)interstitialViewWillDisappear;
- (void)interstitialViewDidDisappear;
-  (void)interstitialViewReceivedEvent:(GuJAdViewEvent*)event;
```

interstitialViewDidFailLoadingWithError ist deprecated in der iOS Version 2.0 und wird in zukünftigen Versionen entfernt. Verwenden Sie stattdessen: interstitialView:didFailLoadingAdWithError:

Wenn das Laden eines Interstitials fehlschlagen sollte, wird die Delegate-Methode **interstitialViewDidFailLoadingWithError**: angesprochen. Auch hier können Sie wieder über den NSError Details über den Fehler in Erfahrung bringen. Im Fehlerfall wird der Interstitialview nicht angezeigt.

Sobald der Adview damit begonnen hat das Ad zu laden, wird die Delegate-Methode **interstitialViewWillLoadAdData**: angesprochen. Dies bedeutet, dass eine Verbindung zum AdServer aufgebaut wird.

Wenn das Ad in Ihrem AdView vollständig und korrekt geladen wurde, wird die Delegate-Methode **interstitialViewDidLoadAdData**: angesprochen. Hierdurch können Sie überprüfen, ob es Probleme beim Laden des Ads gab oder nicht. Wenn ein Fehler aufgetreten ist wird diese Methode nicht angesprochen.

Wenn das InterstitialView geladen wurde, wird die Delegate-Methode **interstitialViewWillAppear** angesprochen. Hier ist der richtige Ansatzpunkt um den aktuellen View für Benutzerinteraktionen zu disablen.

Sobald das Interstitialview angezeigt wird, wird die Delegate-Methode **interstitialViewDidAppear** angesprochen.

Sobald das Interstitialview plant sich wieder zu schließen, wird die Delegate-Methode **interstitialViewWillDisappear** angesprochen. Zu diesem Zeitpunkt sollte Ihre Applikation sich zur Anzeige des Ziel-Views bereit machen und alle notwendigen Inhalte laden.

Wird nun das Interstitialview geschlossen wird die Delegate-Methode **interstitialViewDidDisappear** angesprochen. Dieser Call signalisiert Ihnen nun den Ziel-View anzuzeigen.

Wenn der InterstitialView einen Event oder eine Systemnachricht empfängt, wird diese an die Delegate-Methode **interstitialView:receivedEvent**: weitergeleitet. So werden beispielsweise Informationen von Drittanbieter-SDKs (in diesem Fall mOcean und XAXSIS) an die Delegate-Klasse weitergeleitet und dem Entwickler zugänglich gemacht.

InterstitialView-Instanziierungen

Der einfachste Weg einen Interstitialview zu implementieren ist, den GuJAdViewContext wie folgt zu verwenden:

```
[[GuJAdViewContextinstanceForAdspaceId:@"AdSpaceId"] interstitialAdView]
> OR GCD BLOCK:
[[GuJAdViewContext instanceForAdspaceId:@"AdSpaceId"]
interstitialAdViewWithCompletionHandler:^BOOL(GuJAdView *_adView, NSError *_error) {
    return YES;
}];
```

Optional können Sie dem Interstitialview Keywords mitgeben. Wenn keine Kampagne, die über den AdServer ausgespielt wird, mit dem Keyword übereinstimmt, wird der Interstitialview nicht angezeigt. Hierbei wird Ihnen eine Fehlermeldung über die Delegate-Methode **interstitialViewDidFailLoadingWithError**: zur Verfügung gestellt.

```
[[GuJAdViewContextinstanceForAdspaceId:@"AdSpaceId"] interstitialAdViewForKeywords:
[NSArray arrayWithObjects:@"key1",@"key2", nil]
];
> OR GCD BLOCK:
[[GuJAdViewContext instanceForAdspaceId:@"AdSpaceId"]
interstitialAdViewForKeywords:[NSArray arrayWithObjects:@"key1",@"key2", nil]
completion:^BOOL(GuJAdView *_adView, NSError *_error) {
        return YES;
}];
```

Besonderheiten beim Startstitial

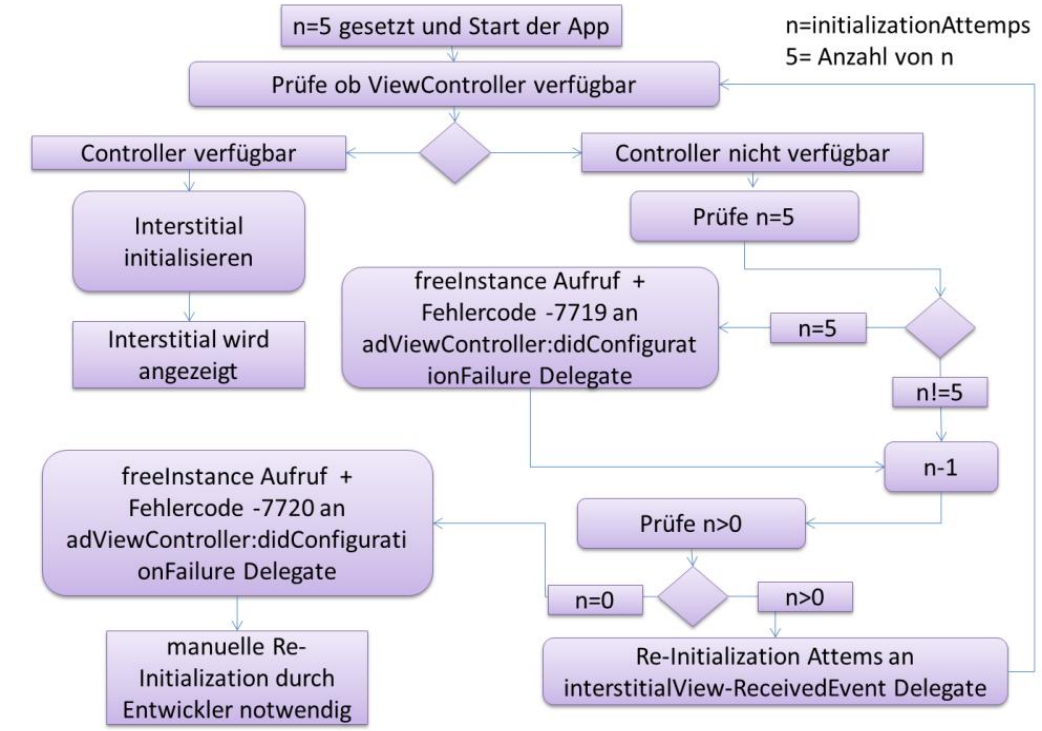
Erst wenn die App einen ViewController (RootViewController) zur Verfügung stellt, kann sich das InterstitialView erfolgreich einblenden. In einigen Fällen, gerade dann wenn das InterstitialView kurz nach dem Laden der App eingeblendet werden soll, ist ein solch benötigter ViewController noch nicht verfügbar oder noch im Begriff geladen zu werden. Dies betrifft in der Regel Apps welche das InterstitialView in der AppDelegate Klasse aufrufen und **NICHT** die GCD-Blöcke verwenden.

Für diesen Fall bietet das SDK eine Initialisierungs-Warteschleife. In dieser Schleife wird im SDK in einem Intervall von einer (1) Sekunde überprüft ob der/ein erforderlicher ViewController vorhanden ist. Der Entwickler kann die Häufigkeit dieser Überprüfungen wie folgt steuern:

```
[GuJAdViewContext_ initalizationAttempts:5];
```

Die Zahl fünf (5) würde in diesem Beispiel dazu führen, dass das SDK fünf (5) Sekunden lang versuchen würde, einen ViewController für die erfolgreiche Darstellung des InterstitialView zu suchen. Sollte nach fünf Sekunden kein ViewController vorhanden sein, beendet sich das SDK selbst. Hier ist der Prozess gleich dem Aufruf: **freeInstance**. Hier müsste der Entwickler die Re-Initialisierung manuell erneut vornehmen.

In dieser Initialisierungs-Warteschleife werden verschiedene Fehlercodes/Events zurückgespielt. Folgende Übersicht verdeutlicht, wann welche Info vom SDK zurückgespielt wird und wohin dies erfolgt.



- Fehlercode -7719 (Missing (Root)ViewController) wird an das Delegate **„adViewController:didConfigurationFailure:“** gesendet, wenn die Abfrage erstmalig durchlaufen wurde. Wenn der Entwickler die Initialisierungs-Warteschleife nicht nutzen möchte, wäre auf diesen Fehlercode zu reagieren, um die Re-Initialisierung manuell vornehmen zu können.
- Während der Initialisierungs-Warteschleife feuert das SDK die Information „Re-Initalization Attempt“ an das Delegate „interstitialViewReceivedEvent“ für jeden Initialisierungsversuch.
- Im Falle des Endes der Initialisierungsschleife und dem Ausbleiben des erfolgreichen Initialisierens wird der Fehlercode -7720 an das Delegate **„adViewController:didConfigurationFailure:“** gesendet und informiert den Entwickler, dass kein ViewController gefunden wurde und das SDK sich selbst beendet hat.

Zusätzliche Request-Parameter und -Header für GuJAdView definieren

Mit folgendem Aufruf können Sie zusätzliche Parameter für die Ad-Server-Anfrage setzten. Hier wird zwischen RequestHeaderField und RequestParameter unterschieden.

- Ein RequestHeaderField ist ein Parameter, welcher im HTTP-Header übergeben wird.
- Ein RequestParameter ist ein Parameter welcher im QueryString, also in der URL übergeben wird.

```
[[myGuJAdViewContext] addAdServerRequestHeaderField:@"name" value:@"key1"]
[[myGuJAdViewContext] addAdServerRequestParameter:@"name" value:@"key1"]
```

Die Parameter sollten vor dem Absenden des Ad-Server-Requests gesetzt werden. Also vor dem Aufruf **adView:adViewWithOrigin:**, **adViewForKeywords:adViewForKeywords:origin**, **interstitialAdView** und **interstitialAdViewForKeywords**.

Fehlermeldungen

Fehler-meldung	SDK-interner Name	Beschreibung
-1000	GUJ_ERROR_CODE_GENERAL_ UNDEFINED	Ein unbekannter oder undefinierter Fehler, welcher auf ein iOS-Problem oder einen Fehler in der Ad-Server-Reponse hinweist.
-1004	GUJ_ERROR_CODE_UNABLE_ TO_COMPLETE	Eine Funktion oder Anfrage konnte vom SDK nicht beendet werden. Dies kann im Entstehungsprozess des AdViews oder bei der Initialisierung des SDK auftreten, wenn z.B. einer dieser Prozesse manuell oder automatisch abgebrochen wird. Auch bei einer Server-Verbindung, welche in ein Time-Out läuft, kann dieser Fehler ausgegeben werden.
-1005	GUJ_ERROR_CODE_COMMAND _ FAILED_OR_UNKNOWN	Wenn ein ORMMA-Kommando nicht ausgeführt werden kann oder unbekannt ist.
-1006	GUJ_ERROR_CODE_ UNAVAILABLE	Wenn ein ORMMA-Kommando oder -Funktion und/oder eine Resource nicht verfügbar ist.
-1007	GUJ_ERROR_CODE_FAILED_TO _ ASSIGN_OBJ	Wenn das SDK ein AdView-Objekt nicht an eine Instanz eines Drittanbieter-SDKs weiterleiten kann.
-1109	GUJ_ERROR_CODE_ORMMA_ CALL_UNHANDLED	Wenn ein ORMMA-Call /eine ORMMA-Funktion verfügbar ist, aber nicht vom SDK ausgeführt werden kann.
1	GUJ_ERROR_CODE_ADSPACE_ ID	Wenn die Ad-Space-ID nicht richtig konfiguriert wurde.
1005	GUJ_ERROR_CODE_SERVER_ ERROR	Wenn die Ad-Server-Response in einem unbekannten Format ist.

Fehler-meldung	SDK-interner Name	Beschreibung
1006	GUJ_ERROR_CODE_ INCORRECT_ AD_FORMAT	Wenn die Ad-Server-Response ein unbekanntes Ad-Format hat.
1008	GUJ_ERROR_CODE_INVALID_ AD_ FORMAT_HEADER	Wenn die Ad-Server-Response einen leeren Body, fehlerhafte Flight-Header-Codes oder einen Serverfehler ausweist.
1009	GUJ_ERROR_CODE_MISSING_ ADCONFIGURATION	Wenn die Ad-Konfiguration fehlerhaft ist.
1010	GUJ_ERROR_CODE_INVALID_ AD_ SERVER_RESPONSE	Wenn die Ad-Server-Response zwar korrekte Flights-Header ausweist, aber ein ungültiges Body-Format hat.
22	GUJ_ERROR_CODE_MOCEAN_ AD_FAILD_LOADING	Wenn das SDK das mOcean-SDK nicht laden kann.
400	GUJ_ERROR_CODE_CORE_ LOCATION	Wenn das Core-Location-Framework fehlerhaft ist und/oder nicht geladen und/oder nicht initialisiert werden kann.
2003	GUJ_ERROR_CALENDAR_ UNAVAILABLE	Wenn das Calendar-Framework fehlerhaft ist und/oder nicht geladen und/oder initialisiert werden kann.

Bekannte ORMMA-Issues

(ISSUE 9000.001)

Das ORMMA-Javascript (V1.1) liefert keine Antwort auf einige native Funktionscallbacks wie beispielsweise Video- oder Audio-Wiedergaben, Maps oder Events.
In diesem Falle muss der Ad-Designer zusätzliche Javascript-Handler implementieren um auf relevante Notifications, die vom SDK gesendet werden, zu reagieren.

(ISSUE 9000.002)

Bei der Funktion Click toCalendar (ormma.createEvent()) muss der Ad-Designer eine Notification an das User-Interface senden, um zu signalisieren, dass ein neuer Kalendereintrag erzeugt wurde.

(ISSUE 9000.003)

Der Inhalt des ausgelieferten Ads sollte Informationen zu den Abmaßen des Ads enthalten. Andernfalls ist das SDK nicht in der Lage die Größe des benötigten Ormma-Views zu ermitteln. Folge ist, dass kein Ad angezeigt wird, da per Default das Ormma-View auf 1x1px gesetzt ist. In diesem Falle wird eine Fehlermeldung mit dem Error-Code: 9002 zurückgeliefert.

Technische Ansprechpartner

Arne Steinmetz

Head Product & Technology
+49 40 3703-7384
steinmetz.arne@ems.guj.de

Daniel Gerold

Product Manager Mobile
+49 221 533-4981
gerold.daniel@ems.guj.de

Jens Jensen

Product Manager Mobile
+49 40 3703-7475
jensen.jens@ems.guj.de