Yandex Mobile Ads

Интеграция

10.07.2024

Яндекс

Yandex Mobile Ads. Интеграция. Версия 2.0

Дата подготовки документа: 10.07.2024

Этот документ является составной частью технической документации Яндекса.

Предупреждение об исключительных правах и конфиденциальной информации

Исключительные права на все результаты интеллектуальной деятельности и приравненные к ним средства индивидуализации юридических лиц, товаров, работ, услуг и предприятий, которым предоставляется правовая охрана (интеллектуальную собственность), используемые при разработке, поддержке и эксплуатации службы Yandex Mobile Ads, включая, но не ограничиваясь, программы для ЭВМ, базы данных, изображения, тексты, другие произведения, а также изобретения, полезные модели, товарные знаки, знаки обслуживания, коммерческие обозначения и фирменные наименования, принадлежат ООО «ЯНДЕКС» либо его лицензиарам.

Использование результатов интеллектуальной деятельности и приравненных к ним средств индивидуализации в целях, не связанных с разработкой, поддержкой и эксплуатацией службы Yandex Mobile Ads, не допускается без получения предварительного согласия правообладателя. Настоящий документ содержит конфиденциальную информацию ООО «ЯНДЕКС». Использование конфиденциальной информации в целях, не связанных с разработкой, поддержкой и эксплуатацией службы Yandex Mobile Ads, а равно как и разглашение таковой, не допускается. При этом под разглашением понимается любое действие или бездействие, в результате которых конфиденциальная информация в любой возможной форме (устной, письменной, иной форме, в том числе с использованием технических средств) становится известной третьим лицам без согласия обладателя такой информации либо вопреки трудовому или гражданско-правовому договору.

Отношения ООО «ЯНДЕКС» с лицами, привлекаемыми для разработки, поддержки и эксплуатации службы Yandex Mobile Ads, регулируются законодательством Российской Федерации и заключаемыми в соответствии с ним трудовыми и/или гражданско-правовыми договорами (соглашениями). Нарушение требований об охране результатов интеллектуальной деятельности и приравненных к ним средств индивидуализации, а равно как и конфиденциальной информации, влечет за собой дисциплинарную, гражданско-правовую, административную или уголовную ответственность в соответствии с законодательством Российской Федерации.

Контактная информация

ООО «ЯНДЕКС»

https://www.yandex.ru

Тел.: +7 495 739 7000

Email: pr@yandex-team.ru

Главный офис: 119021, Россия, г. Москва, ул. Льва Толстого, д. 16

Содержание

Подключение Mobile Ads SDK 4
Поддержка SKAdNetwork 9
..... 9

Форматы рекламы 9
Баннерная реклама 9
Типы баннера 10
Подключение баннера 11
Полноэкранная реклама 14
Реклама с вознаграждением 16
Нативная реклама 18
Подключение нативной рекламы 18
Загрузка рекламы 18
Слайдер рекламных объявлений 22
Настройка внешнего оформления рекламы 25
Отладка 32

Подключение Mobile Ads SDK

Внимание:

Это архивная версия документации. Актуальная документация по всем платформам находится здесь.

Примечание:

1. Для загрузки любого вида рекламы необходима версия iOS 13.0 и выше.
2. Для корректной работы на iOS 14 и выше обратите внимание на дополнительные шаги.

Внимание:

Для работы SDK требуется подключение ресурсов, расположенных в `YandexMobileAdsBundle.bundle` из `YandexMobileAds.framework`. При подключении SDK через CocoaPods эти ресурсы подключаются автоматически.

При использовании кастомизированного подключения `YandexMobileAds.framework` обязательно убедитесь в том, что `YandexMobileAds.bundle` копируется в ресурсы проекта.

Библиотека может работать со следующими системами управления зависимостями:

Swift Package Manager

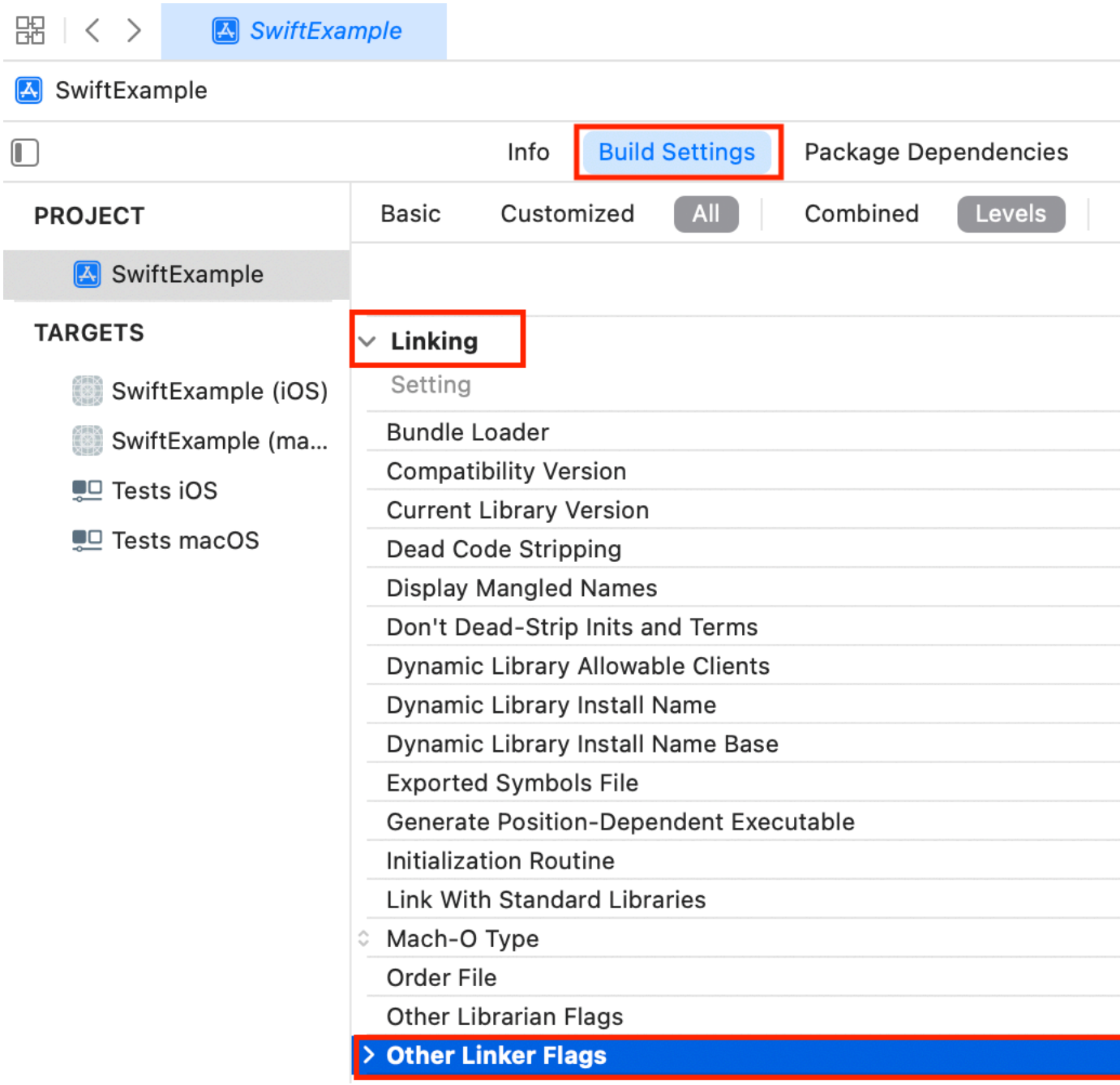
Примечание:

Адаптеры медиации недоступны для подключения через Swift Package Manager. Если вы используете медиацию, рекомендуется интеграция через CocoaPods.

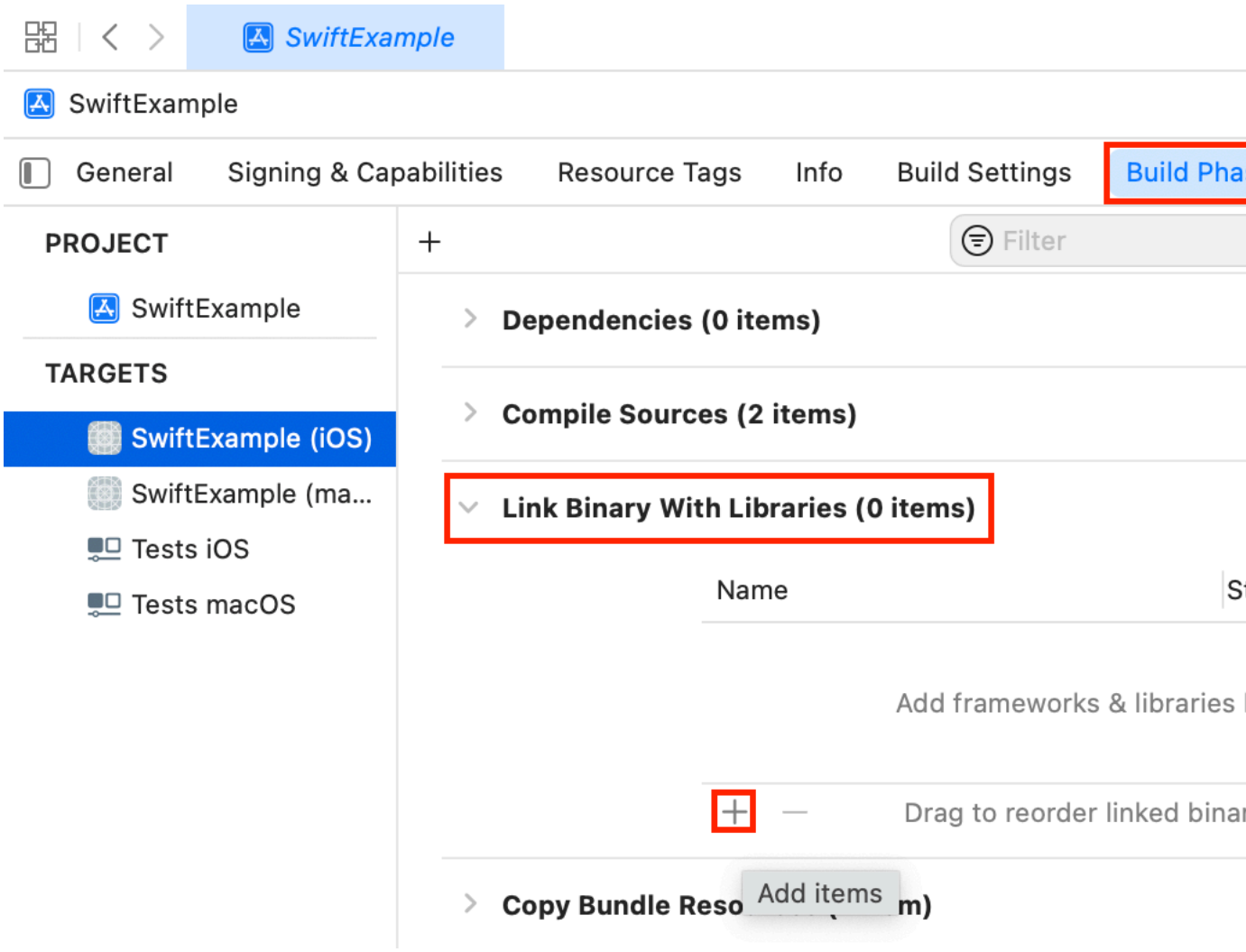
Чтобы подключить библиотеку, выполните следующее:

1. В Xcode, в своем проекте добавьте зависимость через **File** → **Add Packages**.
2. Укажите URL репозитория `https://github.com/yandexmobile/yandex-ads-sdk-swift`, в нем находится Swift-пакет.

3. В настройках сборки **Build Settings**, в секции **Linking**, добавьте значение параметра `Other Linker Flags = -ObjC`.

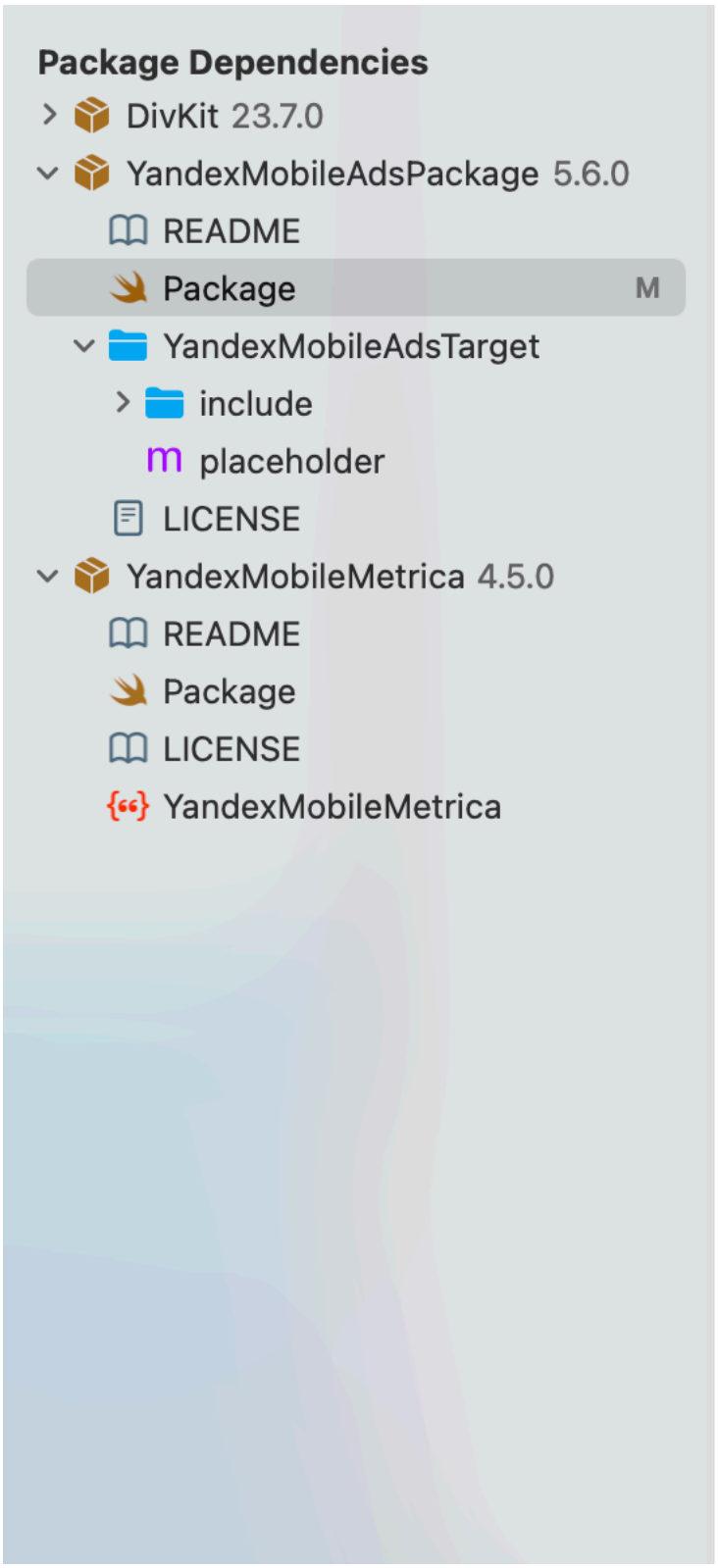


4. Проверьте, что таргет слинкован с YandexMobileAdsPackage. Если таргет не слинкован, добавьте связь с библиотекой в секции **Link Binary With Libraries** через значок **+**.



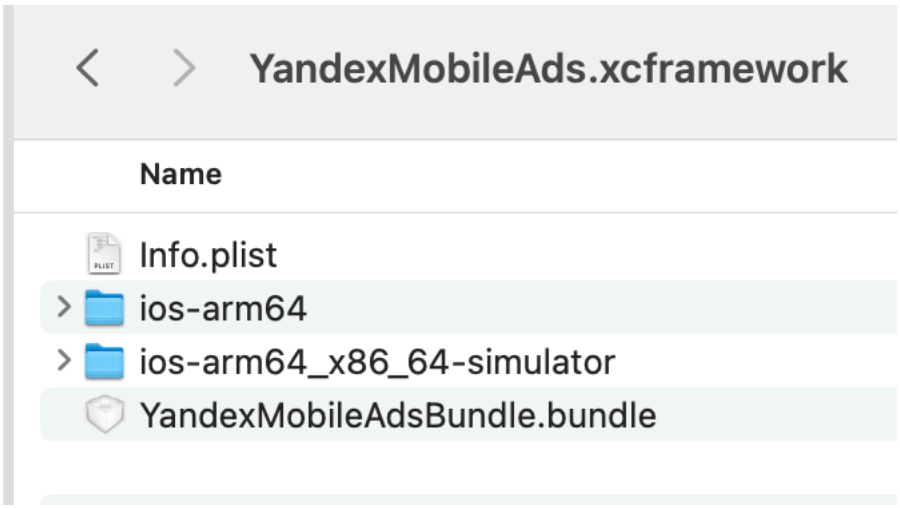
5. Подключите YandexMobileAdsBundle.bundle из YandexMobileAds. Для этого:

   a. перейдите по ссылке для **YandexMobileAds** в секции **.binaryTarget** (**Package Dependencies → YandexMobileAdsPackage → кликните на Package.swift → найдите ссылку для YandexMobileAds в секции .binaryTarget**);

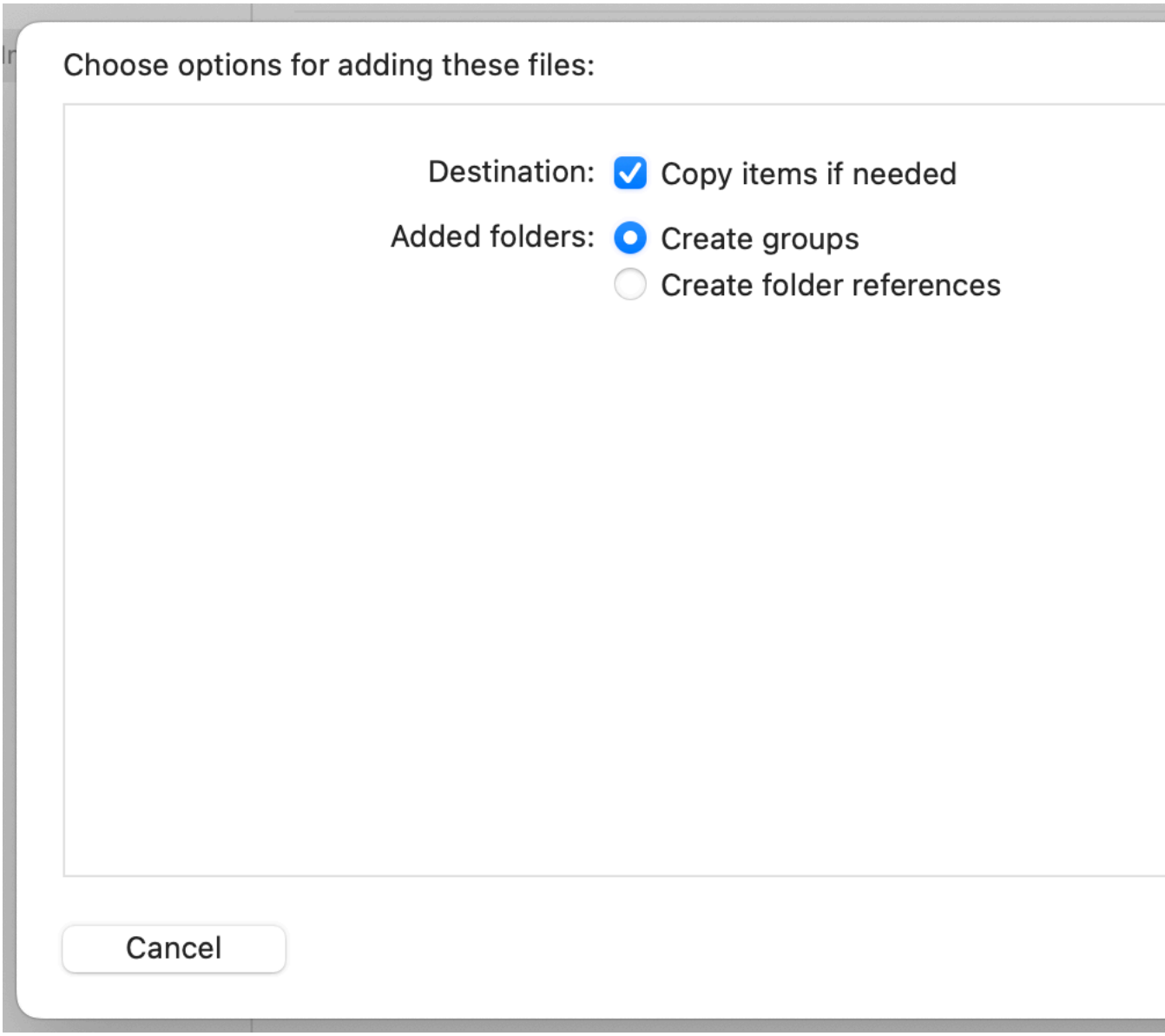


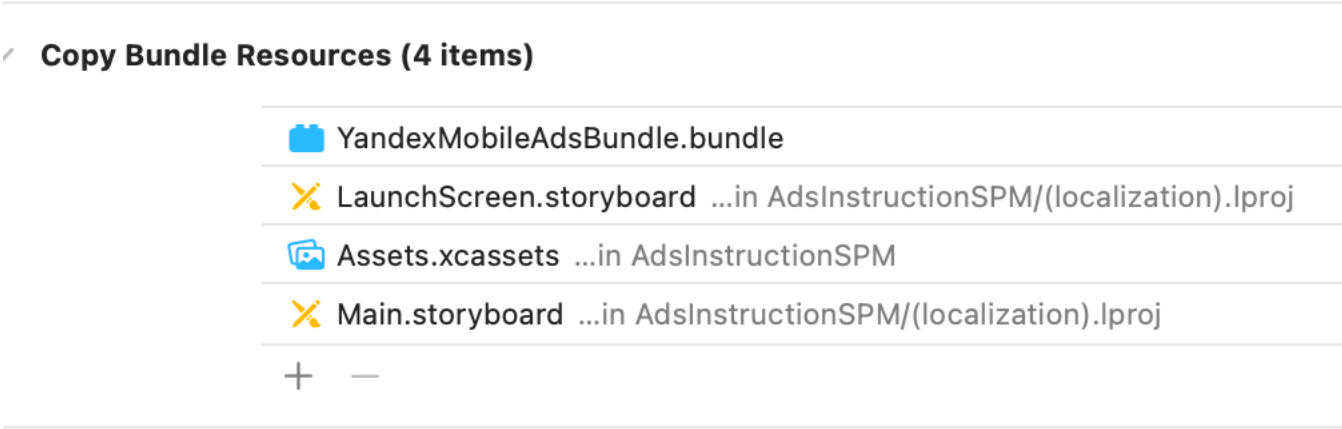

```
24              dependencies
25                  .target(
26                  .target(
27                  .product
28                  .product
29              ],
30              path: "Yande
31              linkerSettin
32                  .linkedF
33                  .linkedF
34                  .linkedF
35                  .linkedF
36                  .linkedF
37                  .linkedF
38                  .linkedF
39                  .linkedF
40                  .linkedF
41                  .linkedF
42                  .linkedF
43                  .linkedF
44                  .linkedF
45                  .linkedF
46                  .linkedF
47                  .linkedF
48              ]
49          ),
50          .binaryTarget(
51              name: "Yande
52              url:
                    "https:/
                    .0/Yande
53              checksum: "d
54          ),
55          .binaryTarget(
56              name: "Yande
57              url:
```

   b. откройте скачанный архив;



c. подключите YandexMobileAdsBundle.bundle в Copy Bundle Resource фазу (**Ваш таргет → Build Phases → Copy Bundle Resource → Кнопка + → Add other → выберите YandexMobileAdsBundle.bundle → выберите пункты Copy items if needed и Create groups**).





CocoaPods

Библиотека Yandex Mobile Ads SDK адаптирована для работы с системой управления зависимостями CocoaPods и поддерживает статический способ подключения. Чтобы подключить библиотеку, добавьте в `Podfile` проекта зависимости (пример статического фреймворка):

```
pod 'YandexMobileAds', '5.9.1'
pod 'YandexMobileAdsInstream', '0.18.0'
```

Поддержка SKAdNetwork

Примечание:

SKAdNetwork поддерживается для версии SDK 4.1.2 и выше.

Mobile Ads SDK поддерживает трекинг установок приложений с помощью фреймворка SKAdNetwork. Трекинг установок работает для всех устройств, даже если доступ к IDFA отсутствует.

Чтобы включить функциональность, добавьте идентификаторы поддерживаемых рекламных сетей в файл `Info.plist` приложения.

```
<key>SKAdNetworkItems</key>
        <array>
        <dict>
        <key>SKAdNetworkIdentifier</key>
        <string>zq492l623r.skadnetwork</string>
        </dict>
        </array>
```

Для дополнительной информации ознакомьтесь с разделом Configuring a Source App документации Apple.

Ознакомьтесь с примерами использования SDK.

Форматы рекламы

Баннерная реклама

Внимание:

Это архивная версия документации. Актуальная документация по всем платформам находится здесь.

Баннер — настраиваемое объявление, которое занимает часть экрана и реагирует на нажатие.

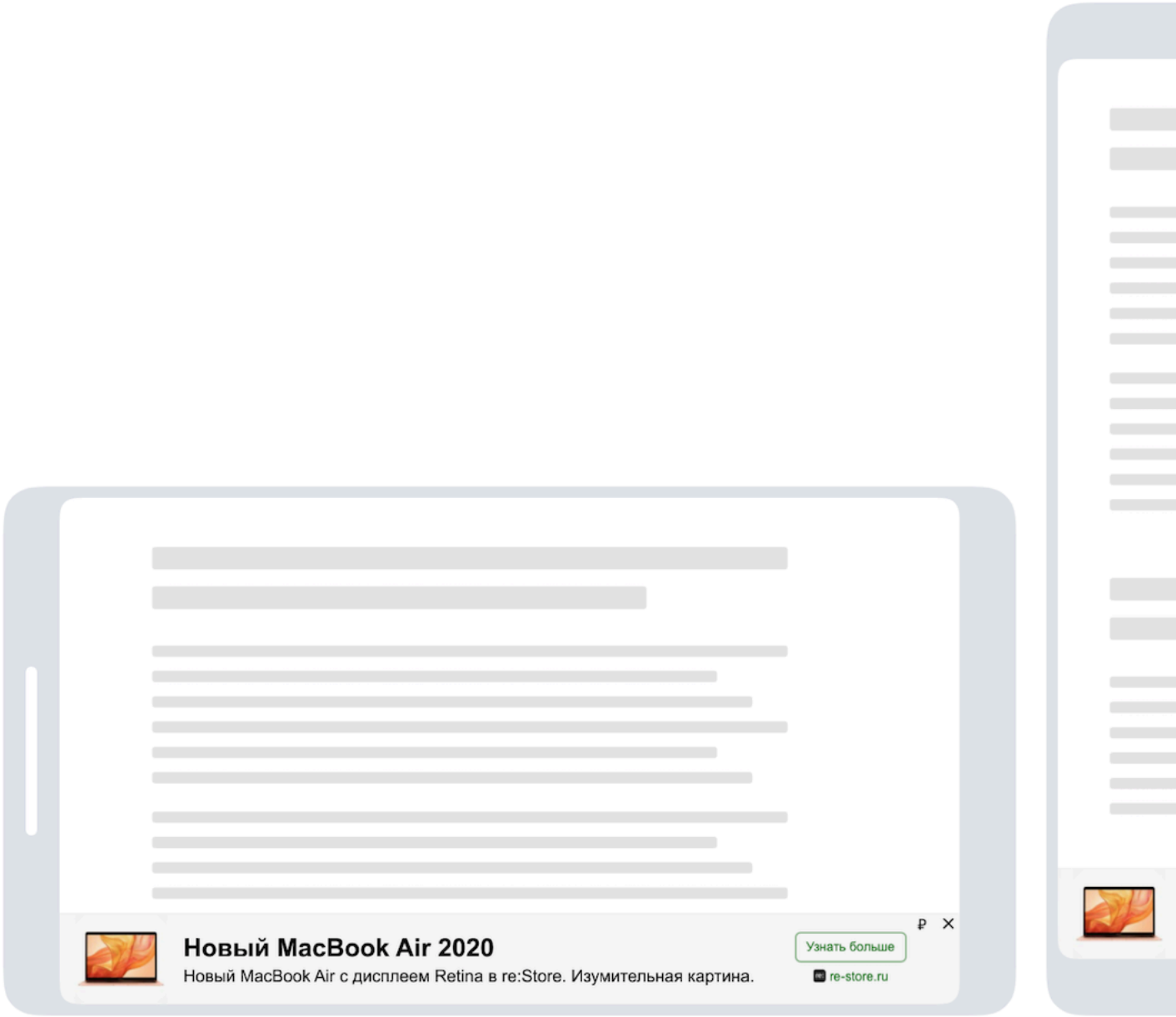
Типы баннера

Sticky баннер

Особенности:

1. Используется заданная ширина баннера. Высота подбирается автоматически.
2. Ширина баннера задается с помощью метода +stickySizeWithContainerWidth:.
3. Высота баннера не должна превышать 15% высоты устройства и не должна быть меньше 50 dp.

Примеры отображения баннера:

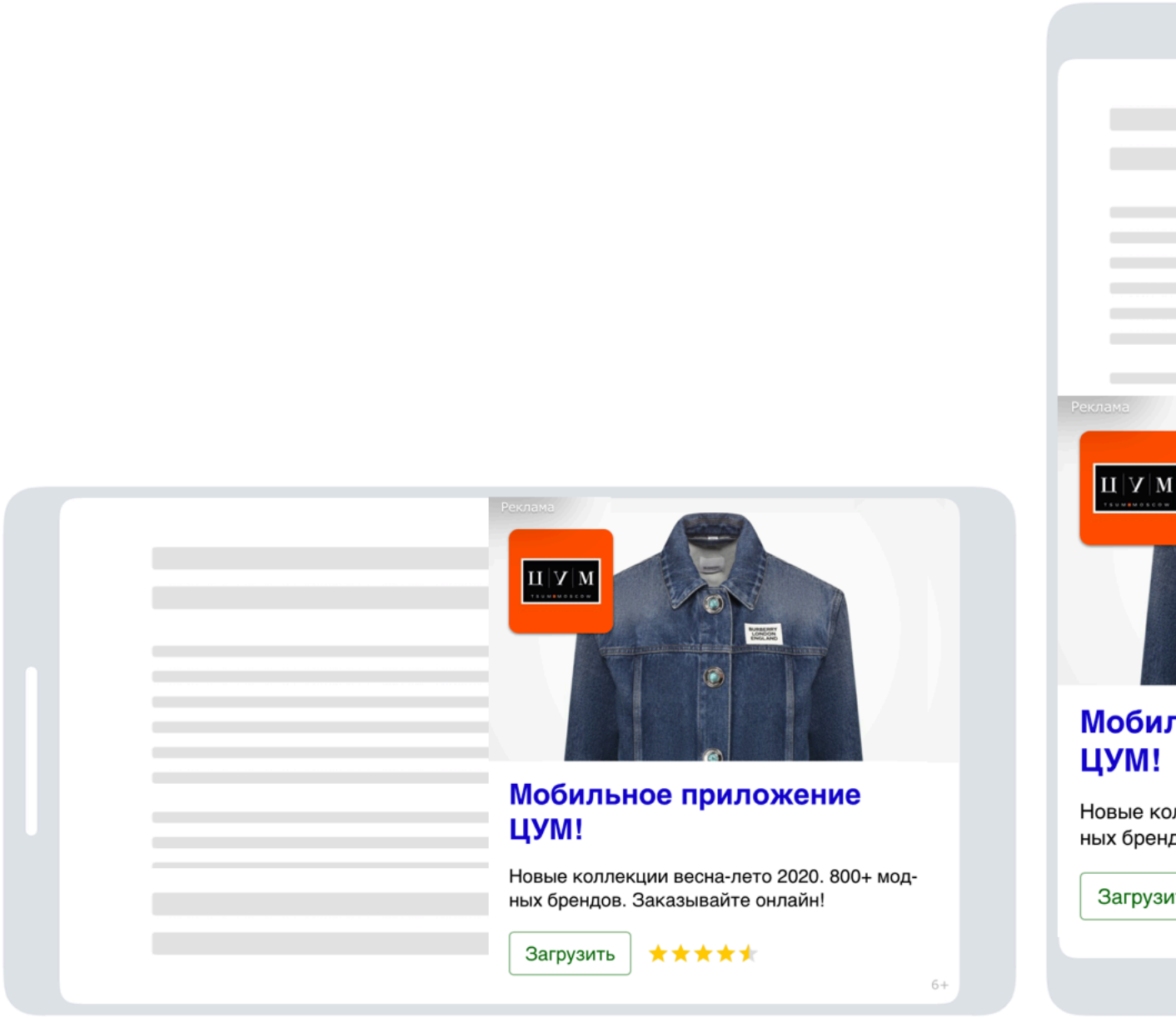


Flex баннер

Особенности:

1. Баннер заполняет весь блок, используя заданные максимальные размеры.
2. Ширина и высота баннера задается с помощью метода +flexibleSizeWithCGSize:.
3. SDK накладывает `NSLayoutConstraint` на высоту баннера.

Примеры отображения баннера:



Подключение баннера

Чтобы подключить баннер:

1. Добавьте импорт:

Swift

```
import YandexMobileAds
```

Objective-C

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

2. Создайте @property, где будет храниться ссылка на баннер:

   Swift

   ```
   var adView: YMAAdView!
   ```

   Objective-C

   ```
   @property (nonatomic, strong) YMAAdView *adView;
   ```

3. Создайте баннер:

   Sticky баннер

   Чтобы задать ширину баннера, вызовите метод +stickySizeWithContainerWidth:.

   Swift

   ```
   let adSize = YMAAdSize.stickySize(withContainerWidth: width)
   let adView = YMAAdView(adUnitID: "", adSize: adSize)
   adView.delegate = self
   ```

   Objective-C

   ```
   YMAAdSize adSize = [YMAAdSize stickySizeWithContainerWidth:width];
   YMAAdView *adView = [[YMAAdView alloc] initWithAdUnitID:<AdUnitID> adSize:adSize];
   adView.delegate = self;
   ```

   Flex баннер

   Чтобы задать ширину и высоту баннера, вызовите метод +flexibleSizeWithCGSize:.

   Swift

   ```
   let adSize = YMAAdSize.flexibleSize(with: size)
   let adView = YMAAdView(adUnitID: "", adSize: adSize)
   adView.delegate = self
   ```

   Objective-C

   ```
   YMAAdSize adSize = [YMAAdSize flexibleSizeWithCGSize:size];
   YMAAdView *adView = [[YMAAdView alloc] initWithAdUnitID:<AdUnitID> adSize:adSize];
   adView.delegate = self;
   ```

   Ограничение: Требования к размерам баннера при отображении видеорекламы

   Минимальный размер баннера, в котором поддерживается воспроизведение видео: 300x160 или 160x300.

   `AdUnitId` — уникальный идентификатор рекламного места, который выдается в Партнерском интерфейсе и имеет вид: R-M-XXXXXX-Y.

   При этом `self` должен удовлетворять протоколу YMAAdViewDelegate. В случае если делегат реализует метод -viewControllerForPresentingModalView, ссылки будут открываться во встроенном браузере внутри приложения. Иначе ссылки будут открываться в браузере, установленном на устройстве.

   Чтобы определить причины, по которым реклама работает некорректно, используйте метод -adViewDidFailLoading:error:.

   Описание ошибок доступно в разделе YMAAdErrorCode.

4. Разместите баннер. Вы можете разместить баннер двумя способами:

   - С помощью `autolayout constraints`.

     Добавьте баннер в `UIView`. Затем добавьте `autolayout constraints`, чтобы баннер отображался в нужном месте.

     Swift

     ```
     view.addSubview(adView)
     adView.translatesAutoresizingMaskIntoConstraints = false
     var adViewConstraints = [
         adView.leadingAnchor.constraint(equalTo: adView.superview!.leadingAnchor),
         adView.trailingAnchor.constraint(equalTo: adView.superview!.trailingAnchor)
     ]
     let bottomDistance: CGFloat = 8
     if #available(iOS 11.0, *) {
         adViewConstraints.append(
     ```

```
        adView.bottomAnchor.constraint(equalTo: view.safeAreaLayoutGuide.bottomAnchor, constant:
 bottomDistance)
    )
} else {
    adViewConstraints.append(
        adView.bottomAnchor.constraint(equalTo: view.bottomAnchor, constant: bottomDistance)
    )
}
NSLayoutConstraint.activate(adViewConstraints)
```

Objective-C

```
UIView *adView = self.adView;
[self.view addSubview:adView];
adView.translatesAutoresizingMaskIntoConstraints = NO;

NSMutableArray *adViewConstraints = [NSMutableArray arrayWithArray:@[
    [adView.leadingAnchor constraintEqualToAnchor:adView.superview.leadingAnchor],
    [adView.trailingAnchor constraintEqualToAnchor:adView.superview.trailingAnchor]
]];
int bottomDistance = 8;
if (@available(iOS 11.0, *)) {
    UILayoutGuide *guide = self.view.safeAreaLayoutGuide;
    [adViewConstraints addObject:[adView.bottomAnchor constraintEqualToAnchor:guide.bottomAnchor
                                                                     constant:bottomDistance]];
} else {
    [adViewConstraints addObject:[adView.bottomAnchor
 constraintEqualToAnchor:adView.superview.bottomAnchor
                                                                     constant:bottomDistance]];
}
[NSLayoutConstraint activateConstraints:adViewConstraints];
```

- С помощью следующих методов:

  Swift

  ```
  displayAtTop(in:)
  displayAtBottom(in:)
  ```

  Objective-C

  ```
  -displayAtTopInView:;
  -displayAtBottomInView:;
  ```

  В обоих случаях баннеры центрируются по горизонтали.

5. Загрузите баннер методом -loadAdWithRequest:.

   С помощью класса YMAAdRequest передайте код, полученный в интерфейсе Adfox (подробнее смотрите в помощи по Adfox).

   Swift

   ```
   // Код из интерфейса Adfox для работы с прямыми кампаниями.
   var parameters = [String: String]()
   parameters["adf_ownerid"] = "<example>"
   parameters["adf_p1"] = "<example>"
   parameters["adf_p2"] = "<example>"
   parameters["adf_pfc"] = "<example>"
   parameters["adf_pfb"] = "<example>"
   parameters["adf_pt"] = "<example>"
   parameters["adf_pd"] = "<example>"
   parameters["adf_pw"] = "<example>"
   parameters["adf_pv"] = "<example>"
   parameters["adf_prr"] = "<example>"
   parameters["adf_pdw"] = "<example>"
   parameters["adf_pdh"] = "<example>"
   let request = YMAMutableAdRequest()
   request.age = age
   request.contextQuery = contextQuery
   request.contextTags = contextTags
   request.gender = gender
   request.location = location
   request.parameters = parameters
   adView.loadAd(with: adRequest)
   ```

   Objective-C

   ```
   // Код из интерфейса Adfox для работы с прямыми кампаниями.
   NSMutableDictionary *parameters = [[NSMutableDictionary alloc] init];
   parameters[@"adf_ownerid"] = @"<example>";
   parameters[@"adf_p1"] = @"<example>";
   parameters[@"adf_p2"] = @"<example>";
   parameters[@"adf_pfc"] = @"<example>";
   parameters[@"adf_pfb"] = @"<example>";
   parameters[@"adf_pt"] = @"<example>";
   parameters[@"adf_pd"] = @"<example>";
   ```

```
parameters[@"adf_pw"] = @"<example>";
parameters[@"adf_pv"] = @"<example>";
parameters[@"adf_prr"] = @"<example>";
parameters[@"adf_pdw"] = @"<example>";
parameters[@"adf_pdh"] = @"<example>";
YMAMutableAdRequest *request = [[YMAMutableAdRequest alloc] init];
request.age = age;
request.contextQuery = contextQuery;
request.contextTags = contextTags;
request.gender = gender;
request.location = location;
request.parameters = parameters;
[self.adView loadAdWithRequest:adRequest];
```

6. Опционально можно включить логирование с помощью метода +enableLogging. Если показ рекламы не был засчитан, в консоли появится сообщение.
7. Опционально можно настроить получение уведомлений об окончании проигрывания видео в баннерной рекламе.

Пример реализации

Swift

```
// Получение экземпляра YMAVideoController с помощью videoController.
let videoController = adView.videoController

// Установка делегата, который реализует протокол YMAVideoDelegate.
videoController.delegate = self

// Реализация метода протокола YMAVideoDelegate.
// MARK: - YMAVideoDelegate
func videoControllerDidFinishPlayingVideo(_ videoController: YMAVideoController) {
    print("Video complete");
}
```

Objective-C

```
// Получение экземпляра YMAVideoController с помощью videoController.
YMAVideoController *videoController = self.adView.videoController;

// Установка делегата, который реализует протокол YMAVideoDelegate.
videoController.delegate = self;

// Реализация метода протокола YMAVideoDelegate.
#pragma mark - YMAVideoDelegate
- (void)videoControllerDidFinishPlayingVideo:(YMAVideoController *)videoController
{
    NSLog(@"%@", @"Video complete");
}
```

Понятия, связанные с данным

Классы и протоколы для работы с баннерной рекламой

Отслеживание работы рекламы

Узнайте больше

Пример рекламы

Полноэкранная реклама

Внимание:

Это архивная версия документации. Актуальная документация по всем платформам находится здесь.

Полноэкранная реклама (Interstitial) — это настраиваемое объявление, отображаемое на весь экран и реагирующее на нажатие.

Чтобы подключить рекламу:

1. Добавьте импорт:

   Swift

   ```
   import YandexMobileAds
   ```

   Objective-C

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

2. Создайте `@property`, где будет храниться ссылка на баннер:

   Swift

   ```
   var interstitialAd: YMAInterstitialAd!
   ```

   Objective-C

   ```
   @property (nonatomic, strong) YMAInterstitialAd *interstitialAd;
   ```

3. Произведите инициализацию и предзагрузку рекламы методом -loadWithRequest:.

   С помощью класса YMAAdRequest передайте код, полученный в интерфейсе Adfox (подробнее смотрите в помощи по Adfox).

   Swift

   ```
   // Код из интерфейса Adfox для работы с прямыми кампаниями.
   var parameters = [String: String]()
   parameters["adf_ownerid"] = "<example>"
   parameters["adf_p1"] = "<example>"
   parameters["adf_p2"] = "<example>"
   parameters["adf_pfc"] = "<example>"
   parameters["adf_pfb"] = "<example>"
   parameters["adf_pt"] = "<example>"
   parameters["adf_pd"] = "<example>"
   parameters["adf_pw"] = "<example>"
   parameters["adf_pv"] = "<example>"
   parameters["adf_prr"] = "<example>"
   parameters["adf_pdw"] = "<example>"
   parameters["adf_pdh"] = "<example>"
   let request = YMAMutableAdRequest()
   request.age = age
   request.contextQuery = contextQuery
   request.contextTags = contextTags
   request.gender = gender
   request.location = location
   request.parameters = parameters
   interstitialAd = YMAInterstitialAd(adUnitID: "<AdUnitID>")
   interstitialAd.delegate = self
   interstitial.loadAd(with: request)
   ```

   Objective-C

   ```
   // Код из интерфейса Adfox для работы с прямыми кампаниями.
   NSMutableDictionary *parameters = [[NSMutableDictionary alloc] init];
   parameters[@"adf_ownerid"] = @"<example>";
   parameters[@"adf_p1"] = @"<example>";
   parameters[@"adf_p2"] = @"<example>";
   parameters[@"adf_pfc"] = @"<example>";
   parameters[@"adf_pfb"] = @"<example>";
   parameters[@"adf_pt"] = @"<example>";
   parameters[@"adf_pd"] = @"<example>";
   parameters[@"adf_pw"] = @"<example>";
   parameters[@"adf_pv"] = @"<example>";
   parameters[@"adf_prr"] = @"<example>";
   parameters[@"adf_pdw"] = @"<example>";
   parameters[@"adf_pdh"] = @"<example>";
   YMAMutableAdRequest *request = [[YMAMutableAdRequest alloc] init];
   request.age = age;
   request.contextQuery = contextQuery;
   request.contextTags = contextTags;
   request.gender = gender;
   request.location = location;
   request.parameters = parameters;
   self.interstitialAd = [[YMAInterstitialAd alloc] initWithAdUnitID:<ваш уникальный AdUnitID>];
   self.interstitialAd.delegate = self;
   [self.interstitialAd loadAdWithRequest:request];
   ```

4. Начните отображение объявления, используя данный метод:

   Swift

   ```
   func interstitialAdDidLoad(_ interstitialAd: YMAInterstitialAd) {
   ```

```
    interstitialAd.present(from: self)
}
```

Objective-C

```
- (void)interstitialAdDidLoad:(YMAInterstitialAd *)interstitialAd
{
    [interstitialAd presentFromViewController:self];
}
```

5. Опционально можно включить логирование с помощью метода +enableLogging. Если показ рекламы не был засчитан, в консоли появится сообщение.

Чтобы определить причины, по которым реклама работает некорректно, используйте методы

Swift

```
func interstitialAdDidFail(toLoad interstitialAd: YMAInterstitialAd, error: Error)
func interstitialAdDidFail(toPresent interstitialAd: YMAInterstitialAd, error: Error)
```

Objective-C

```
- (void)interstitialAdDidFailToLoad:(YMAInterstitialAd *)interstitialAd error:(NSError *)error;
- (void)interstitialAdDidFailToPresent:(YMAInterstitialAd *)interstitialAd error:(NSError *)error;
```

Описание ошибок доступно в разделе YMAAdErrorCode.

Чтобы посмотреть, как реклама будет отображаться в приложении, используйте демонстрационный `AdUnitID`:

- `demo-interstitial-yandex`

Понятия, связанные с данным

Классы и протоколы для работы с полноэкранной рекламой

Отслеживание работы рекламы

Узнайте больше

Пример рекламы

Подключение рекламы с вознаграждением

Внимание:

Это архивная версия документации. Актуальная документация по всем платформам находится здесь.

Реклама с вознаграждением (Rewarded Ad) — настраиваемое объявление, отображаемое на весь экран. За просмотр такой рекламы пользователь получает вознаграждение.

Чтобы подключить рекламу:

1. Добавьте импорт:

   Swift

   ```
   import YandexMobileAds
   ```

   Objective-C

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

2. Создайте `@property`, где будет храниться ссылка на рекламное объявление:

   Swift

   ```
   var rewardedAd: YMARewardedAd!
   ```

   Objective-C

   ```
   @property (nonatomic, strong) YMARewardedAd *rewardedAd;
   ```

3. Произведите инициализацию рекламы:

   Swift

   ```
   rewardedAd = YMARewardedAd(adUnitID: "<AdUnitID>")
   rewardedAd.delegate = self
   ```

   Objective-C

   ```
   self.rewardedAd = [[YMARewardedAd alloc] initWithAdUnitID:<ваш уникальный AdUnitID>];
   self.rewardedAd.delegate = self;
   ```

   `AdUnitId` — уникальный идентификатор рекламного места, который выдается в Партнерском интерфейсе и имеет вид: R-M-XXXXXX-Y.

   При этом `self` должен удовлетворять протоколу YMARewardedAdDelegate.
4. Загрузите рекламу.

   Swift

   ```
   rewardedAd.load()
   ```

   Objective-C

   ```
   [self.rewardedAd load];
   ```

   Опционально, с помощью класса YMAAdRequest, можно передать данные для таргетирования. Пример использования класса см. в разделе Полноэкранная реклама.

   Производить предзагрузку рекламы необходимо в той же ориентации, в которой она будет показана (иначе реклама не будет показана из-за несоответствия размеров экрана и баннера).
5. Начните отображение объявления, используя данный метод:

   Swift

   ```
   func rewardedAdDidLoad(_ rewardedAd: YMARewardedAd) {
       rewardedAd.present(from: self)
   }
   ```

   Objective-C

   ```
   - (void)rewardedAdDidLoad:(YMARewardedAd *)rewardedAd
   {
       [rewardedAd presentFromViewController:self];
   }
   ```

6. Если вы используете механизм награждения на стороне клиента («client-side reward»), реализуйте метод делегата -rewardedAd:didReward:. Он вызывается, когда показ засчитан и пользователь может быть награжден за просмотр рекламы. Используйте этот момент, чтобы выдать награду пользователю приложения.
7. Опционально можно включить логирование с помощью метода +enableLogging. Если показ рекламы не был засчитан, в консоли появится сообщение.

   Чтобы определить причины, по которым реклама работает некорректно, используйте методы

   Swift

   ```
   func rewardedAdDidFail(toLoad rewardedAd: YMARewardedAd, error: Error)
   func rewardedAdDidFail(toPresent rewardedAd: YMARewardedAd, error: Error)
   ```

   Objective-C

   ```
   - (void)rewardedAdDidFailToLoad:(YMARewardedAd *)rewardedAd error:(NSError *)error;
   - (void)rewardedAdDidFailToPresent:(YMARewardedAd *)rewardedAd error:(NSError *)error;
   ```

   Описание ошибок доступно в разделе YMAAdErrorCode.

Чтобы посмотреть, как реклама будет отображаться в приложении, используйте демонстрационный `AdUnitID`:

- `demo-rewarded-yandex`

Понятия, связанные с данным

Классы и протоколы для работы с видеорекламой с вознаграждением

Нативная реклама

Подключение нативной рекламы

Внимание:

Это архивная версия документации. Актуальная документация по всем платформам находится здесь.

Yandex Mobile Ads SDK предоставляет возможность показывать рекламу с использованием собственных визуальных компонентов.

Нативная реклама может изменяться в зависимости от функций и дизайна приложения, в котором она отображается. Оформление нативной рекламы соответствует среде, в которую она встраивается. Такая реклама выглядит органично и дополняет приложение полезной информацией.

SDK также предоставляет набор готовых настраиваемых визуальных компонентов (шаблонов), которые позволяют использовать все преимущества отрисовки нативными средствами платформы и не требуют создания собственного дизайна.

Yandex Mobile Ads SDK поддерживает несколько типов рекламы: App Install, Content, Image.

Чтобы подключить рекламу:

1. Ознакомьтесь с требованиями к размещению рекламы и рекламных компонентов.
2. Загрузите рекламу.
3. Настройте внешнее оформление рекламы.

Понятия, связанные с данным

Классы и протоколы для работы с нативной рекламой

Отслеживание работы рекламы

Узнайте больше

YMANativeErrorCode

Пример рекламы

Загрузка рекламы

Внимание:

Это архивная версия документации. Актуальная документация по всем платформам находится здесь.

Загрузка рекламных объявлений

1. Создайте экземпляр класса YMANativeAdLoader для получения нативных объявлений.
2. Создайте конфигурацию запроса `nativeAdRequestConfiguration` с помощью класса YMANativeAdRequestConfiguration. В качестве параметров запроса можно передать идентификатор рекламного блока, способ загрузки изображений, возраст, гендерные признаки и другие данные, способные улучшить качество подбора рекламы.
3. Установите делегат для получения рекламы, который реализует протокол YMANativeAdLoaderDelegate.
4. Чтобы отслеживать процесс загрузки рекламы, реализуйте методы протокола YMANativeAdLoaderDelegate: -nativeAdLoader:didFailLoadingWithError:, -nativeAdLoader:didLoadAd:.
5. Отправьте загрузчику сообщение `loadAdWithRequestConfiguration:`, чтобы загрузить рекламу.

   С помощью класса YMAMutableNativeAdRequestConfiguration передайте код, полученный в интерфейсе Adfox (подробнее смотрите в помощи по Adfox).

   Swift

```
// Код из интерфейса Adfox для работы с прямыми кампаниями.
```

```
var parameters = [String: String]()
parameters["adf_ownerid"] = "<example>"
parameters["adf_p1"] = "<example>"
parameters["adf_p2"] = "<example>"
parameters["adf_pfc"] = "<example>"
parameters["adf_pfb"] = "<example>"
parameters["adf_pt"] = "<example>"
parameters["adf_pd"] = "<example>"
parameters["adf_pw"] = "<example>"
parameters["adf_pv"] = "<example>"
parameters["adf_prr"] = "<example>"
parameters["adf_pdw"] = "<example>"
parameters["adf_pdh"] = "<example>"
let requestConfiguration = YMAMutableNativeAdRequestConfiguration(adUnitID: "R-M-XXXXXX")
requestConfiguration.age = age
requestConfiguration.contextQuery = contextQuery
requestConfiguration.contextTags = contextTags
requestConfiguration.gender = gender
requestConfiguration.location = location
requestConfiguration.parameters = parameters

adLoader.loadAd(with: requestConfiguration)
```

Objective-C

```
// Код из интерфейса Adfox для работы с прямыми кампаниями.
NSMutableDictionary *parameters = [[NSMutableDictionary alloc] init];
parameters[@"adf_ownerid"] = @"<example>";
parameters[@"adf_p1"] = @"<example>";
parameters[@"adf_p2"] = @"<example>";
parameters[@"adf_pfc"] = @"<example>";
parameters[@"adf_pfb"] = @"<example>";
parameters[@"adf_pt"] = @"<example>";
parameters[@"adf_pd"] = @"<example>";
parameters[@"adf_pw"] = @"<example>";
parameters[@"adf_pv"] = @"<example>";
parameters[@"adf_prr"] = @"<example>";
parameters[@"adf_pdw"] = @"<example>";
parameters[@"adf_pdh"] = @"<example>";
YMAMutableNativeAdRequestConfiguration *requestConfiguration =
        [[YMAMutableNativeAdRequestConfiguration alloc] initWithAdUnitID:@"R-M-XXXXXX"];
requestConfiguration.age = age;
requestConfiguration.contextQuery = contextQuery;
requestConfiguration.contextTags = contextTags;
requestConfiguration.gender = gender;
requestConfiguration.location = location;
requestConfiguration.parameters = parameters;

[adLoader loadAdWithRequestConfiguration:requestConfiguration];
```

6. Если реклама загрузилась, будет вызван метод:

   Swift

   ```
   func nativeAdLoader(_ loader: YMANativeAdLoader, didLoad ad: YMANativeAd)
   ```

   Objective-C

   ```
   - (void)nativeAdLoader:(YMANativeAdLoader *)loader didLoadAd:(id<YMANativeAd>)ad
   ```

7. Если реклама не загрузилась, будет вызван следующий метод:

   Swift

   ```
   func nativeAdLoader(_ loader: YMANativeAdLoader, didFailLoadingWithError error: Error)
   ```

   Objective-C

   ```
   - (void)nativeAdLoader:(YMANativeAdLoader *)loader didFailLoadingWithError:(NSError *)error
   ```

   Подробно о возможных ошибках см. раздел YMANativeErrorCode.

8. Опционально, с помощью свойства `(nonatomic, copy, readonly) NSString *info` протокола YMANativeAd можно получить строку, которая была задана в интерфейсе Adfox, в поле **Дополнительный текст**.

Примеры с демонстрационным AdUnitID

Чтобы посмотреть, как реклама будет отображаться в приложении, используйте демонстрационный AdUnitID:

- для рекламы типа App Install — demo-native-app-yandex

- для рекламы типа Content — demo-native-content-yandex
- для рекламы типа Image — R-M-187883-1. Также необходимо передать список параметров:

Swift

```
// Код из интерфейса Adfox для работы с прямыми кампаниями.
parameters["adf_ownerid"] = "168627"
parameters["adf_p1"] = "bvyhu"
parameters["adf_p2"] = "fksh"
parameters["adf_pt"] = "b"
```

Objective-C

```
// Код из интерфейса Adfox для работы с прямыми кампаниями.
parameters[@"adf_ownerid"] = @"168627";
parameters[@"adf_p1"] = @"bvyhu";
parameters[@"adf_p2"] = @"fksh";
parameters[@"adf_pt"] = @"b";
```

Загрузка нескольких рекламных объявлений

Yandex Mobile Ads SDK предоставляет возможность загрузки нескольких рекламных объявлений одним запросом (до девяти объявлений).

1. Создайте экземпляр класса YMANativeBulkAdLoader для получения нативных объявлений.
2. Создайте конфигурацию запроса `nativeAdRequestConfiguration` с помощью класса YMANativeAdRequestConfiguration. В качестве параметров запроса можно передать идентификатор рекламного блока, способ загрузки изображений, возраст, гендерные признаки и другие данные, способные улучшить качество подбора рекламы.
3. Установите делегат для получения рекламы, который реализует протокол YMANativeBulkAdLoaderDelegate.
4. Чтобы отслеживать процесс загрузки рекламы, реализуйте методы протокола YMANativeBulkAdLoaderDelegate: -nativeBulkAdLoader:didLoadAds:, -nativeBulkAdLoader:didFailLoadingWithError:.
5. Отправьте загрузчику конфигурацию запроса и количество запрашиваемых объявлений (параметр `adsCount`).

   С помощью класса YMAMutableNativeAdRequestConfiguration передайте код, полученный в интерфейсе Adfox (подробнее смотрите в помощи по Adfox).

   Swift

```
// Код из интерфейса Adfox для работы с прямыми кампаниями.
var parameters = [String: String]()
parameters["adf_ownerid"] = "<example>"
parameters["adf_p1"] = "<example>"
parameters["adf_p2"] = "<example>"
parameters["adf_pfc"] = "<example>"
parameters["adf_pfb"] = "<example>"
parameters["adf_pt"] = "<example>"
parameters["adf_pd"] = "<example>"
parameters["adf_pw"] = "<example>"
parameters["adf_pv"] = "<example>"
parameters["adf_prr"] = "<example>"
parameters["adf_pdw"] = "<example>"
parameters["adf_pdh"] = "<example>"
let requestConfiguration = YMAMutableNativeAdRequestConfiguration(adUnitID: "R-M-XXXXXX")
requestConfiguration.age = age
requestConfiguration.contextQuery = contextQuery
requestConfiguration.contextTags = contextTags
requestConfiguration.gender = gender
requestConfiguration.location = location
requestConfiguration.parameters = parameters

adLoader.loadAds(with: requestConfiguration, adsCount: adsCount)
```

   Objective-C

```
// Код из интерфейса Adfox для работы с прямыми кампаниями.
NSMutableDictionary *parameters = [[NSMutableDictionary alloc] init];
parameters[@"adf_ownerid"] = @"<example>";
parameters[@"adf_p1"] = @"<example>";
parameters[@"adf_p2"] = @"<example>";
parameters[@"adf_pfc"] = @"<example>";
parameters[@"adf_pfb"] = @"<example>";
parameters[@"adf_pt"] = @"<example>";
parameters[@"adf_pd"] = @"<example>";
parameters[@"adf_pw"] = @"<example>";
```

```
parameters[@"adf_pv"] = @"<example>";
parameters[@"adf_prr"] = @"<example>";
parameters[@"adf_pdw"] = @"<example>";
parameters[@"adf_pdh"] = @"<example>";
YMAMutableNativeAdRequestConfiguration *requestConfiguration =
        [[YMAMutableNativeAdRequestConfiguration alloc] initWithAdUnitID:@"R-M-XXXXXX"];
requestConfiguration.age = age;
requestConfiguration.contextQuery = contextQuery;
requestConfiguration.contextTags = contextTags;
requestConfiguration.gender = gender;
requestConfiguration.location = location;
requestConfiguration.parameters = parameters;

[adLoader loadAdWithRequestConfiguration:requestConfiguration adsCount:adsCount];
```

Примечание:

Yandex Mobile Ads SDK не гарантирует, что будет загружено запрошенное количество объявлений. Полученный массив будет содержать от 0 до adsCount объектов NativeAd. Все полученные объекты рекламы можно показывать отдельно друг от друга, используя описанные выше способы внешнего оформления нативных объявлений.

Способы загрузки изображений

Автоматическая загрузка

Если приложение одновременно хранит ссылки на одно или небольшое количество объявлений, рекомендуется использовать автоматическую загрузку.

Передайте YES в качестве значения свойства shouldLoadImagesAutomatically при создании конфигурации:

Swift

```
let requestConfiguration = YMAMutableNativeAdRequestConfiguration(adUnitID: AdUnitID)
requestConfiguration.shouldLoadImagesAutomatically = true
```

Objective-C

```
YMAMutableNativeAdRequestConfiguration *requestConfiguration =
    [[YMAMutableNativeAdRequestConfiguration alloc] initWithAdUnitID:AdUnitID];
    requestConfiguration.shouldLoadImagesAutomatically = YES;
```

В полученном объявлении будут присутствовать готовые изображения. Они хранятся в памяти приложения до уничтожения объявления.

Ручная загрузка

Если приложение одновременно хранит ссылки на большое количество объявлений, рекомендуется использовать ручную загрузку изображений.

Передайте NO в качестве значения свойства shouldLoadImagesAutomatically при создании конфигурации:

Swift

```
let requestConfiguration = YMAMutableNativeAdRequestConfiguration(adUnitID: AdUnitID)
requestConfiguration.shouldLoadImagesAutomatically = false
```

Objective-C

```
YMAMutableNativeAdRequestConfiguration *requestConfiguration =
    [[YMAMutableNativeAdRequestConfiguration alloc] initWithAdUnitID:AdUnitID];
    requestConfiguration.shouldLoadImagesAutomatically = NO;
```

В полученном объявлении будут присутствовать только размеры изображений. Чтобы загрузить сами изображения, вызовите метод loadImages у полученного объявления.

Внимание:

Все изображения кэшируются, но могут быть удалены в любой момент, поэтому необходимо вызывать метод loadImages перед каждым показом объявления.

Swift

```
func showAd() {
    // Show ad: custom native view or template
    view.addSubview(adView)
    adView.ad?.loadImages()
}
```

Objective-C

```
- (void)showAd
{
    // Show ad: custom native view or template
    [self.view addSubview:self.adView];
    [self.adView.ad loadImages];
}
```

Уведомления о процессе загрузки изображений

Ограничение: Получение уведомлений работает только при использовании ручной загрузки изображений.

Включить уведомления

Чтобы включить уведомления о завершении загрузки изображений, вызовите метод -addImageLoadingObserver:

Swift

```
ad?.add(self)

...
func nativeAdDidFinishLoadingImages(_ ad: YMANativeAd) {
    print("Finished loading images")
}
```

Objective-C

```
[ad addImageLoadingObserver:self];
...
- (void)nativeAdDidFinishLoadingImages:(id<YMANativeAd>)ad
{
    NSLog(@"Finished loading images");
}
```

Отключить уведомления

Чтобы отключить уведомления о завершении загрузки изображений, вызовите метод -removeImageLoadingObserver:

Swift

```
ad?.remove(self)
```

Objective-C

```
[ad removeImageLoadingObserver:self];
```

Слайдер рекламных объявлений

Внимание:

Это архивная версия документации. Актуальная документация по всем платформам находится здесь.

Yandex Mobile Ads SDK предоставляет возможность показывать слайдер из связанных между собой рекламных объявлений. Подробнее про слайдер можно узнать в статье.

Слайдер реализован по принципу работы нативной рекламы. Внешний вид рекламы может настраиваться паблишером в зависимости от функций и дизайна приложения, в котором будет отображаться слайдер.

Загрузка слайдера

1. Создайте экземпляр класса YMANativeAdLoader для получения рекламных объявлений в рамках слайдера.
2. Создайте конфигурацию запроса `nativeAdRequestConfiguration` с помощью класса YMANativeAdRequestConfiguration. В качестве параметров запроса можно передать идентификатор рекламного блока, способ загрузки изображений, возраст, гендерные признаки и другие данные, способные улучшить качество подбора рекламы.
3. Установите делегат для получения рекламы, который реализует протокол YMANativeAdLoaderDelegate.

4. Чтобы отслеживать процесс загрузки рекламы, реализуйте методы протокола YMANativeAdLoaderDelegate: -nativeAdLoader:didFailLoadingWithError:, -nativeAdLoader:didLoadAd:.
5. Отправьте загрузчику сообщение `loadAdWithRequestConfiguration:`, чтобы загрузить рекламу.

   Swift

   ```
   adLoader.loadAd(with: requestConfiguration)
   ```

   Objective-C

   ```
   [self.adLoader loadAdWithRequestConfiguration:requestConfiguration];
   ```

6. Если реклама загрузилась, будет вызван метод:

   Swift

   ```
   func nativeAdLoader(_ loader: YMANativeAdLoader, didLoad ad: YMANativeAd)
   ```

   Objective-C

   ```
   - (void)nativeAdLoader:(YMANativeAdLoader *)loader didLoadAd:(id<YMANativeAd>)ad
   ```

7. Если реклама не загрузилась, будет вызван следующий метод:

   Swift

   ```
   func nativeAdLoader(_ loader: YMANativeAdLoader, didFailLoadingWithError error: Error)
   ```

   Objective-C

   ```
   - (void)nativeAdLoader:(YMANativeAdLoader *)loader didFailLoadingWithError:(NSError *)error
   ```

   Подробно о возможных ошибках см. раздел YMANativeErrorCode.

Общий запрос за рекламой

Swift

```
// Создание загрузчика
adLoader = YMANativeAdLoader()
adLoader.delegate = self

// Создание конфигурации запроса
let requestConfiguration = YMANativeAdRequestConfiguration(adUnitID: "<AdUnitID>")

// Передача загрузчику конфигурации запроса
adLoader.loadAd(with: requestConfiguration)

// Реализация методов делегата
....

func nativeAdLoader(_ loader: YMANativeAdLoader, didLoad ad: YMANativeAd) {
    // Отобразите рекламу
}
```

Objective-C

```
// Создание загрузчика
self.adLoader = [[YMANativeAdLoader alloc] init];
self.adLoader.delegate = self;

// Создание конфигурации запроса
YMANativeAdRequestConfiguration *requestConfiguration =
        [[YMANativeAdRequestConfiguration alloc] initWithAdUnitID:@"your_AdUnitID"];

// Передача загрузчику конфигурации запроса
[self.adLoader loadAdWithRequestConfiguration:requestConfiguration];

// Реализация методов делегата
....
- (void)nativeAdLoader:(YMANativeAdLoader *)loader didLoadAd:(id<YMANativeAd>)ad
{
     // Отобразите рекламу
}
```

Показ слайдера

После окончания загрузки слайдера, все его компоненты необходимо отобразить.

Успешно загруженный слайдер содержит одно или несколько связанных между собой нативных объявлений. Показ рекламных объявлений в рамках слайдера должен происходить в пределах одного общего контейнера, иначе показ объявлений не будет засчитан.

Оформление без использования шаблона

Ручная настройка внешнего оформления нативной рекламы используется в тех случаях, когда возможностей настройки шаблона недостаточно для получения желаемого результата.

Данный способ позволяет самостоятельно сверстать макет нативной рекламы, определить расположение элементов рекламы относительно друг друга. В объявлении могут присутствовать как обязательные, так и опциональные для показа компоненты. Полный их перечень можно найти в разделе Компоненты нативной рекламы.

Совет:

Рекомендуется использовать макет, который включает весь набор возможных компонентов. Как показывает практика, макеты, включающие весь набор компонентов, более кликабельные.

Вызовите метод bindAdToSliderView и передайте в него контейнер для слайдера объявлений.

Внешнее оформление каждого объявления, входящего в слайдер, настраивается одним из способов оформления стандартной нативной рекламы.

Swift

```
    func nativeAdLoader(_ loader: YMANativeAdLoader, didLoad ad: YMANativeAd) {
        self.ad = ad
        ad.delegate = self
        // Проверяем наличие вложенных объявлений
        if ad.ads.count != 0 {
            // Создаем контейнер для слайдера; Вместо YMANativeAdView должен быть ваш наследник этого класса
            let sliderAdView = YMANativeAdView()

            // Вызываем метод bindAd(toSliderView: _) и передаем в него контейнер
            do {
                try ad.bindAd(toSliderView: sliderAdView)
            } catch {
                // Проверяем сообщение об ошибке и исправляем проблему
                return
            }

            for subAd in ad.ads {
                // Подписываемся на делегат
                subAd.delegate = self

                // Создаем рекламное view для объявления
                // Вместо YMANativeAdView должен быть ваш наследник этого класса
                let subAdView = YMANativeAdView()

                // Вызываем метод bind(with: subAdView) для объявления
                do {
                    try subAd.bind(with: subAdView)
                } catch {
                    // Проверяем сообщение об ошибке и исправляем проблему
                    return
                }

                // Добавляем объявление в контейнер
                sliderAdView.addSubview(subAdView)
                // Располагаем объявление в контейнере
            }

        } else {
            // Обработать как обычную нативную рекламу
            // https://yandex.ru/dev/mobile-ads/doc/ios/quick-start/config.html
        }
    }
```

Objective-C

```
- (void)nativeAdLoader:(YMANativeAdLoader *)loader didLoadAd:(id<YMANativeAd>)ad
{
    self.ad = ad;
    self.ad.delegate = self;
    // Проверяем наличие вложенных объявлений
    if (ad.ads.count != 0) {
        // Создаем контейнер для слайдера
        YMANativeAdView *sliderAdView = [[YMANativeAdView alloc] init];
        // Вызываем метод bindAdToSliderView и передаем в него контейнер
        NSError *error = nil;
        [ad bindAdToSliderView:sliderAdView error:&error];
```

```
        // Проверяем успешность привязки
        if (error != nil) {
            // Проверяем сообщение об ошибке и исправляем проблему
            return;
        }

        for (id<YMANativeAd> subAd in ad.ads) {
            // Подписываемся на делегат
            subAd.delegate = self;
            // Создаем рекламное view для объявления
            // Вместо YMANativeAdView должен быть ваш наследник этого класса
            YMANativeAdView *subAdView = [[YMANativeAdView alloc] init];
            NSError *error = nil;
            // Вызываем метод bindWithAdView и передаем в него рекламное view
            [subAd bindWithAdView:subAdView error:&error];

            // Проверяем успешность привязки
            if (error != nil) {
                // Проверяем сообщение об ошибке и исправляем проблему
                continue;
            }

            // Добавляем объявление в контейнер
            [sliderAdView addSubview:subAdView];
            // Располагаем объявление в контейнере
        }

    } else {
        // Обработать как обычную нативную рекламу
        // https://yandex.ru/dev/mobile-ads/doc/ios/quick-start/config.html
    }
}
```

Настройка внешнего оформления рекламы

Внимание:

Это архивная версия документации. Актуальная документация по всем платформам находится здесь.

Существует два способа настройки внешнего оформления рекламы:

1. Оформление с помощью шаблона.

   Использование стандартного шаблона оформления — это самый простой способ работы с нативной рекламой. В шаблоне уже добавлен набор обязательных рекламных компонентов и настроено их расположение относительно друг друга. Шаблон работает с любым поддерживаемым типом рекламы.
2. Оформление без использования шаблона.

   Оформление без шаблона используется в тех случаях, когда для настройки шаблона недостаточно функций, чтобы реклама выглядела органично и соответствовала дизайну приложения.

Оформление с помощью шаблона

Для настройки внешнего оформления можно использовать стандартный шаблон оформления или создать свое оформление на основе стандартного шаблона.

Использование стандартного шаблона оформления

Примечание:

После настройки оформления задайте местоположение и размеры рекламы относительно экрана устройства.

1. Создайте объект — экземпляр класса YMANativeBannerView и установите для него загруженную рекламу:

   Swift

   ```
   let bannerView = YMANativeBannerView()
   bannerView.ad = ad
   view.addSubview(bannerView)
   ```

   Objective-C

   ```
   YMANativeBannerView *bannerView = [[YMANativeBannerView alloc] init];
   bannerView.ad = ad;
   [self.view addSubview:bannerView];
   ```

2. Чтобы получать уведомления о взаимодействии пользователя с рекламой (открытие рекламы, выход из приложения), установите для нее делегат YMANativeAdDelegate, реализующий методы:

   - -nativeAd:didDismissScreen:;
   - -nativeAd:willPresentScreen:;
   - -nativeAdWillLeaveApplication:;
   - -viewControllerForPresentingModalView.

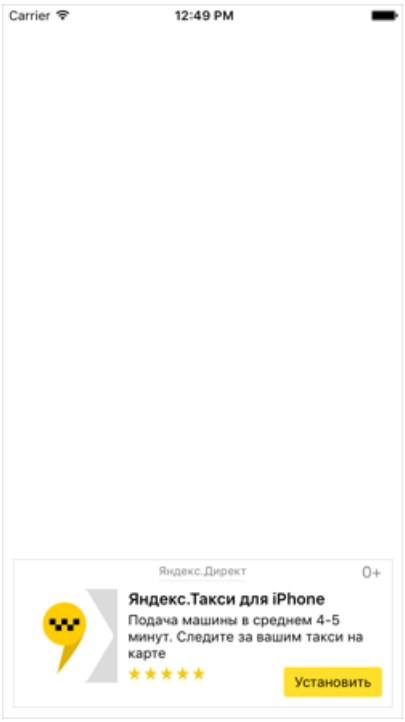
   Swift

   ```
   ad.delegate = self
   ```

   Objective-C

   ```
   ad.delegate = self;
   ```

3. Пример использования стандартного шаблона оформления:



Примечание: Если вашему приложению не подходит стандартное оформление, его можно изменить. Подробнее смотрите в разделе Создание своего оформления на основе шаблона.

Создание своего оформления на основе шаблона

Примечание:

После настройки оформления задайте местоположение и размеры рекламы относительно экрана устройства.

1. Создайте объект — экземпляр класса YMANativeBannerView и установите для него загруженную рекламу:

   Swift

   ```
   let bannerView = YMANativeBannerView()
   bannerView.ad = ad
   view.addSubview(bannerView)
   ```

   Objective-C

   ```
   YMANativeBannerView *bannerView = [[YMANativeBannerView alloc] init];
   bannerView.ad = ad;
   [self.view addSubview:bannerView];
   ```

2. Чтобы получать уведомления о взаимодействии пользователя с рекламой (открытие рекламы, выход из приложения), установите для нее делегат YMANativeAdDelegate, реализующий методы:
   - -nativeAd:didDismissScreen:;
   - -nativeAd:willPresentScreen:;
   - -nativeAdWillLeaveApplication:;
   - -viewControllerForPresentingModalView.

   Swift

   ```
   ad.delegate = self
   ```

   Objective-C

   ```
   ad.delegate = self;
   ```

3. Запросите настройки стандартного шаблона оформления:

   Swift

   ```
   let appearance = YMAMutableNativeTemplateAppearance.default()
   ```

   Objective-C

   ```
   YMAMutableNativeTemplateAppearance *appearance = [[YMANativeTemplateAppearance defaultAppearance]
    mutableCopy];
   ```

4. Задайте предпочитаемые настройки.
5. Чтобы применить настройки к шаблону, вызовите метод -applyAppearance::

   Swift

   ```
   bannerView.apply(appearance)
   ```

   Objective-C

   ```
   [bannerView applyAppearance:appearance];
   ```

Пример настроек оформления

Swift

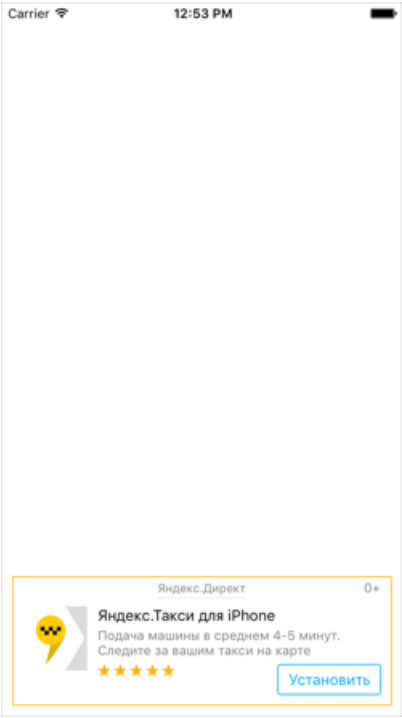
```
// Определяем собственные цветовые решения для работы.
let orangeColor = UIColor(red: 1, green: 176.0/255, blue: 32.0/255, alpha: 1)
let blueColor = UIColor(red: 0, green: 170.0/255, blue: 1, alpha: 1)

// Создаем копию с настройками стандартного шаблона оформления.
let appearance = YMAMutableNativeTemplateAppearance.default()

// Начинаем изменять стандартные настройки шаблона.

// Задаем цвет для рамки рекламного объявления.
appearance.borderColor = orangeColor

// Создаем копию с настройками рейтинга.
let ratingAppearance = appearance.ratingAppearance?.mutableCopy() as? YMAMutableRatingAppearance

// Задаем цвет для закрашенных звезд в рейтинге.
ratingAppearance?.filledStarColor = orangeColor
appearance.ratingAppearance = ratingAppearance

// Задаем цвет и размер шрифта для надписи на кнопке с действием.
let callToActionTextAppearance = YMALabelAppearance(font: .systemFont(ofSize: 14), textColor: blueColor)

// Задаем цвет кнопки для обычного и нажатого состояния, цвет и толщину обводки кнопки.
let callToActionAppearance = YMAButtonAppearance(
    textAppearance: callToActionTextAppearance,
    normalColor: .clear,
    highlightedColor: .gray,
    borderColor: blueColor,
    borderWidth: 1
)
appearance.callToActionAppearance = callToActionAppearance

// Задаем размер и цвет шрифта для надписи с возрастным ограничением.
appearance.ageAppearance = YMALabelAppearance(font: .systemFont(ofSize: 12), textColor: .gray)

// Задаем размер и цвет шрифта для заголовка рекламного объявления.
appearance.titleAppearance = YMALabelAppearance(font: .systemFont(ofSize: 14), textColor: .black)
```

```
// Задаем размер и цвет шрифта для основного рекламного текста.
appearance.bodyAppearance = YMALabelAppearance(font: .systemFont(ofSize: 12), textColor: .gray)

// Задаем ширину изображения и правило формирования размера.
let imageConstraint = YMASizeConstraint(type: .fixed, value: 60)

// Применяем настройки к изображению.
appearance.imageAppearance = YMAImageAppearance(widthConstraint: imageConstraint)
```

Objective-C

```
// Определяем собственные цветовые решения для работы.
UIColor *orangeColor =
  [UIColor colorWithRed:255.f / 255.f green:176.f / 255.f blue:32.f / 255.f alpha:1.f];
UIColor *blueColor =
  [UIColor colorWithRed:0.f / 255.f green:170.f / 255.f blue:255.f / 255.f alpha:1.f];

// Создаем копию с настройками стандартного шаблона оформления.
YMAMutableNativeTemplateAppearance *appearance = [[YMANativeTemplateAppearance defaultAppearance] mutableCopy];

// Начинаем изменять стандартные настройки шаблона.

// Задаем цвет для рамки рекламного объявления.
appearance.borderColor = orangeColor;

// Создаем копию с настройками рейтинга.
YMAMutableRatingAppearance *ratingAppearance = [appearance.ratingAppearance mutableCopy];

// Задаем цвет для закрашенных звезд в рейтинге.
ratingAppearance.filledStarColor = orangeColor;
appearance.ratingAppearance = ratingAppearance;

// Задаем цвет и размер шрифта для надписи на кнопке с действием.
YMALabelAppearance *callToActionTextAppearance =
    [YMALabelAppearance appearanceWithFont:[UIFont systemFontOfSize:14.f]
                                 textColor:blueColor];

// Задаем цвет кнопки для обычного и нажатого состояния, цвет и толщину обводки кнопки.
YMAButtonAppearance *callToActionAppearance =
    [YMAButtonAppearance appearanceWithTextAppearance:callToActionTextAppearance
                                          normalColor:[UIColor clearColor]
                                     highlightedColor:[UIColor grayColor]
                                          borderColor:blueColor
                                          borderWidth:1.f];
appearance.callToActionAppearance = callToActionAppearance;

// задаем размер и цвет шрифта для надписи с возрастным ограничением.
appearance.ageAppearance =
  [YMALabelAppearance appearanceWithFont:[UIFont systemFontOfSize:12.f]
                               textColor:[UIColor grayColor]];

// Задаем размер и цвет шрифта для заголовка рекламного объявления.
appearance.titleAppearance =
  [YMALabelAppearance appearanceWithFont:[UIFont systemFontOfSize:14.f]
                               textColor:[UIColor blackColor]];

// Задаем размер и цвет шрифта для основного рекламного текста.
appearance.bodyAppearance =
  [YMALabelAppearance appearanceWithFont:[UIFont systemFontOfSize:12.f]
                               textColor:[UIColor grayColor]];

// Задаем ширину изображения и правило формирования размера.
YMASizeConstraint *imageConstraint =
  [YMASizeConstraint constraintWithType:YMASizeConstraintTypeFixed value:60.f];

// Применяем настройки к изображению.
appearance.imageAppearance =
  [YMAImageAppearance appearanceWithWidthConstraint:imageConstraint];
```

Получаем собственный дизайн на основе шаблона:



Настройка местоположения и размера рекламы

Ограничение: Требования к размерам mediaView при отображении видеорекламы

Минимальный размер экземпляра класса YMANativeMediaView, в котором поддерживается воспроизведение видео: 300x160 или 160x300.

Для поддержки воспроизведения видео в шаблонах нативной рекламы рекомендуется выставить ширину для YMANativeBannerView не менее 300. Корректная высота для `mediaView` будет вычислена автоматически, с учетом соотношения ширины и высоты.

Управлять размерами и местоположением можно одним из двух способов:

1. с помощью системного механизма `AutoLayout`;

   Примечание:

   При работе с `AutoLayout` задайте `constraint` для ширины `UIView`. Высота будет определена автоматически на основании указанной ширины.
2. вручную, задавая все размеры.

Настройка размеров вручную

Примечание:

Не рекомендуется задавать ширину объекта более 420 логических пикселей.

Задайте для объекта `YMANativeBannerView` ширину и высоту. Высота определяется с помощью метода +heightWithAd:width:appearance:, в который необходимо передать рекламное объявление, ширину и объект `YMANativeTemplateAppearance` (используется для настроек внешнего вида рекламного объявления).

Swift

```
// Задаем отступы слева и справа относительно экрана.
let inset: CGFloat = 50

// Задаем ширину.
let width = view.frame.width - 2 * inset

// Задаем высоту.
let height = YMANativeBannerView.height(with: ad, width: width, appearance: nil)

// Задаем позицию по вертикали.
let y = view.frame.maxY - height - inset

// Задаем координаты и размеры для frame.
```

```
bannerView.frame = CGRect(x: inset, y: y, width: width, height: height)
```

Objective-C

```
// Задаем отступы слева и справа относительно экрана.
CGFloat inset = 50.f;

// Задаем ширину.
CGFloat width = CGRectGetWidth(self.view.frame) - 2 * inset;

// Задаем высоту.
CGFloat height = [YMANativeBannerView heightWithAd:ad width:width appearance:nil];

// Задаем позицию по вертикали.
CGFloat y = CGRectGetMaxY(self.view.frame) - height - inset;

// Задаем координаты и размеры для frame.
bannerView.frame = CGRectMake(inset, y, width, height);
```

Оформление без использования шаблона

Внимание:

Это архивная версия документации. Актуальная документация по всем платформам находится здесь.

Для отображения нативной рекламы объекты `YMANativeAd` должны быть привязаны к определенной `View`. Эта `View` должна определяться как корневой элемент для набора `subview`, к элементам которого будут привязываться данные, содержащиеся в нативной рекламе.

Ограничение: Требования к размерам mediaView при отображении видеорекламы

Минимальный размер экземпляра класса YMANativeMediaView, в котором поддерживается воспроизведение видео: 300x160 или 160x300.

Для поддержки воспроизведения видео в нативной рекламе рекомендуется выставить ширину для `mediaView` не менее 300. Чтобы вычислить соответствующее значение высоты `mediaView`, используйте значение свойства aspectRatio.

Оформление нативной рекламы

1. Ознакомьтесь со списком обязательных и опциональных subview для нативной рекламы.
2. Установите значения всех обязательных subview. Это можно сделать с помощью `nib` или непосредственно в коде. Для этого создайте объект класса YMANativeAdView и определите значения `subview` в `initWithFrame:`

   Пример регистрации набора `subview`:

   Swift

```
override init(frame: CGRect) {
    super.init(frame: frame)
    let titleLabel = createLabel()
    let bodyLabel = createLabel()
    let ageLabel = createLabel()
    let warningLabel = createLabel()
    let sponsoredByLabel = createLabel()
    let priceLabel = createLabel()
    let starRatingView = createStarRatingView()
    let button = createButton()
    let iconImageView = createIconAssetImageView()
    let mediaView = createMediaAssetView()
    addSubview(titleLabel)
    addSubview(bodyLabel)
    addSubview(ageLabel)
    addSubview(warningLabel)
    addSubview(sponsoredByLabel)
    addSubview(priceLabel)
    addSubview(starRatingView)
    addSubview(button)
    addSubview(iconImageView)
    addSubview(mediaView)
    self.titleLabel = titleLabel
    self.bodyLabel = bodyLabel
    self.ageLabel = ageLabel
    self.warningLabel = warningLabel
    self.sponsoredLabel = sponsoredByLabel
    self.callToActionButton = callToActionButton
    self.priceLabel = priceLabel
    self.ratingView = ratingView
    self.iconImageView = iconImageView
    self.mediaView = mediaView
```

```
}
```

Objective-C

```
- (instancetype)initWithFrame:(CGRect)frame
{
    self = [super initWithFrame:frame];
    if (self != nil) {
        UILabel *titleLabel = [self label];
        UILabel *bodyLabel = [self label];
        UILabel *ageLabel = [self secondaryLabel];
        UILabel *warningLabel = [self label];
        UILabel *sponsoredByLabel = [self label];
        UILabel *priceLabel = [self label];
        StarRatingView *ratingView = [self starRatingView];
        UIButton *callToActionButton = [self button];
        UIImageView *iconImageView = [self iconAssetImageView];
        YMANativeMediaView *mediaView = [self mediaAssetView];
        [self addSubview:titleLabel];
        [self addSubview:bodyLabel];
        [self addSubview:ageLabel];
        [self addSubview:warningLabel];
        [self addSubview:sponsoredByLabel];
        [self addSubview:callToActionButton];
        [self addSubview:priceLabel];
        [self addSubview:ratingView];
        [self addSubview:iconImageView];
        [self addSubview:mediaView];
        self.titleLabel = titleLabel;
        self.bodyLabel = bodyLabel;
        self.ageLabel = ageLabel;
        self.warningLabel = warningLabel;
        self.sponsoredLabel = sponsoredByLabel;
        self.callToActionButton = callToActionButton;
        self.priceLabel = priceLabel;
        self.ratingView = ratingView;
        self.iconImageView = iconImageView;
        self.mediaView = mediaView;
    }
    return self;
}
```

3. Чтобы получать уведомления о взаимодействии пользователя с рекламой (открытие и закрытие рекламы, выход из приложения), установите для нее делегат YMANativeAdDelegate, реализующий методы:
   - -closeNativeAd:;
   - -nativeAd:didDismissScreen:;
   - -nativeAd:willPresentScreen:;
   - -nativeAdWillLeaveApplication:;
   - -viewControllerForPresentingModalView.
4. Запросите значения компонентов нативной рекламы с помощью метода -adAssets. Это поможет заранее рассчитать расположение и размеры этих компонентов.

   Swift

   ```
   let assets = ad.adAssets()
   ```

   Objective-C

   ```
   YMANativeAdAssets *assets = [ad adAssets];
   ```

   Пример получения размеров изображения, соотношения сторон медиа и текста заголовка объявления

   Swift

   ```
   let image = assets.image
   let title = assets.title
   let media = assets.media
   print("Image size: \(image?.size ?? .zero)")
   print("Title: \(title ?? "")")
   print(String(format: "Media aspect ratio: %.2f", media?.aspectRatio ?? 0))
   ```

   Objective-C

   ```
   YMANativeAdImage *image = assets.image;
   NSString *title = assets.title;
   YMANativeAdMedia *media = assets.media;
   NSLog(@"Image size: %@", NSStringFromCGSize(image.size));
   NSLog(@"Title: %@", title);
   NSLog(@"Media aspect ratio: %.2f", media.aspectRatio);
   ```

5. Вызовите метод -bindWithAdView:error:, чтобы связать контент с объектом нативной рекламы.

Swift

```
// ...
adView = YMANativeAdView(frame: frame)
//configure content ad view
// ...
func nativeAdLoader(_ loader: YMANativeAdLoader, didLoad ad: YMANativeAd) {
    ad.delegate = self
    do {
        try ad.bind(with: contentAdView)
    } catch {
        print("Error: \(error)")
    }
}
```

Objective-C

```
// ...
self.adView = [[YMANativeAdView alloc] initWithFrame:frame];
//configure content ad view
// ...
- (void)nativeAdLoader:(YMANativeAdLoader *)loader didLoadAd:(id<YMANativeAd>)ad
{
    ad.delegate = self;
    NSError *error = nil;
    BOOL result = [ad bindWithAdView:self.contentAdView error:error];
    if (error != nil) {
        NSLog(@"Error: %@", error);
    }
}
```

Примечание:

Если для обязательного элемента нативной рекламы соответствующее свойство `YMANativeAdView` принимает значение `nil`, привязка не осуществится — реклама не будет показана. Детали в этом случае можно получить из `error`.

Отладка

Внимание:

Это архивная версия документации. Актуальная документация по всем платформам находится здесь.

Логирование

Опционально можно включить логирование с помощью метода +enableLogging. Если показ рекламы не был засчитан, в консоли появится сообщение.

Индикатор некорректной интеграции для нативной рекламы

Если при интеграции нативной рекламы допущена ошибка, то в режиме симулятора, поверх рекламного объявления, появится индикатор. По нажатию на индикатор вы увидите сообщение с отладочной информацией, по которой можно понять причину ошибки. Повторное нажатие на индикатор скрывает сообщение.

Внимание:

По умолчанию индикатор отображается только в режиме симулятора (тип устройства `YMADeviceTypeSimulator`). Типы устройства можно посмотреть в YMADeviceType.

Чтобы включить индикатор также и для реальных устройств, передайте значение `YMADeviceTypeHardware | YMADeviceTypeSimulator` в методе enableVisibilityErrorIndicatorForDeviceType::

Swift

```
YMAMobileAds.enableVisibilityErrorIndicator(for: [.hardware, .simulator])
```

Objective-C

```
[YMAMobileAds enableVisibilityErrorIndicatorForDeviceType:YMADeviceTypeHardware | YMADeviceTypeSimulator]
```

Чтобы выключить индикатор, передайте значение `YMADeviceTypeNone` в методе enableVisibilityErrorIndicatorForDeviceType::

Swift

```
YMAMobileAds.enableVisibilityErrorIndicator(for: [])
```

Objective-C

```
[YMAMobileAds enableVisibilityErrorIndicatorForDeviceType:YMADeviceTypeNone]
```