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

Внимание:

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

Примечание:

1. Для загрузки любого вида рекламы необходима версия Android 4.1 и выше.
2. Видеореклама подбирается только на устройства с версией Android 5.0 и выше.

Библиотека Yandex Mobile Ads предоставляется в формате AAR. Чтобы подключить Mobile Ads SDK:

1. Добавьте зависимость от Yandex Mobile Ads в `build.gradle` файл модуля вашего приложения:

```
dependencies {
    ...
    implementation 'com.yandex.android:mobileads:5.10.0'
}
```

2. Обновите зависимость от Koltin Gradle Plugin в корневом `build.gradle` файле вашего проекта:

```
dependencies {
    ...
    classpath("org.jetbrains.kotlin:kotlin-gradle-plugin:1.7.10")
}
```

3. Добавьте поддержку Java 8 в `build.gradle` файл модуля вашего приложения:

```
android {

    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}
```

4. Настройте разрешение на использование рекламного идентификатора.

   Рекламный идентификатор – уникальный идентификатор сервисов Google Play для показа рекламы пользователям, которые согласны видеть персонализированные объявления. Пользователь может отключить персонализацию рекламы или сбросить идентификатор в настройках. В таком случае рекламные сети не смогут использовать его для подбора релевантной рекламы.

   Версия Mobile Ads SDK 4.5.0 и выше

   В Yandex Mobile Ads SDK, начиная с версии 4.5.0, появилось новое разрешение `com.google.android.gms.permission.AD_ID`. Оно прописано в файле `AndroidManifest.xml` библиотеки. Поэтому указывать его дополнительно в основном манифесте приложения не нужно. Разрешение позволяет использовать рекламный идентификатор для подбора релевантной рекламы от рекламных сетей.

   При необходимости вы можете удалить разрешение. Например, если какие-то политики, как Families Policy, не позволяют использовать идентификатор для подбора рекламы.

   Чтобы разрешение не попало в основной манифест приложения, добавьте в файл `AndroidManifest.xml` строку:

```
<uses-permission android:name="com.google.android.gms.permission.AD_ID" tools:node="remove"/>
```

   Версия Mobile Ads SDK ниже 4.5.0

   Если ваше приложение использует версию Yandex Mobile Ads SDK ниже 4.5.0 и нет ограничений на использование рекламного идентификатора (например, Families Policy), добавьте разрешение в основной манифест приложения `AndroidManifest.xml`:

```
<uses-permission android:name="com.google.android.gms.permission.AD_ID"/>
```

   Отсутствие разрешения и доступа к идентификатору может снизить релевантность рекламы и, как следствие, ваш доход.

Инициализация Mobile Ads SDK

Перед загрузкой рекламы инициализируйте библиотеку с помощью метода initialize(). Инициализация ускорит загрузку объявлений.

Примечание:

Это необходимо делать во время каждого нового запуска приложения. Поэтому рекомендуется добавить код инициализации в метод `onCreate` класса `Application`.

Пример инициализации:

```
public class YandexApplication extends Application {

    private static final String YANDEX_MOBILE_ADS_TAG = "YandexMobileAds";

    @Override
    public void onCreate() {
        super.onCreate();

        MobileAds.initialize(this, new InitializationListener() {
            @Override
            public void onInitializationCompleted() {
                Log.d(YANDEX_MOBILE_ADS_TAG, "SDK initialized");
            }
        });
    }
}
```

Ознакомьтесь с примерами использования SDK.

Форматы рекламы

Процесс подключения баннерной рекламы

Внимание:

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

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

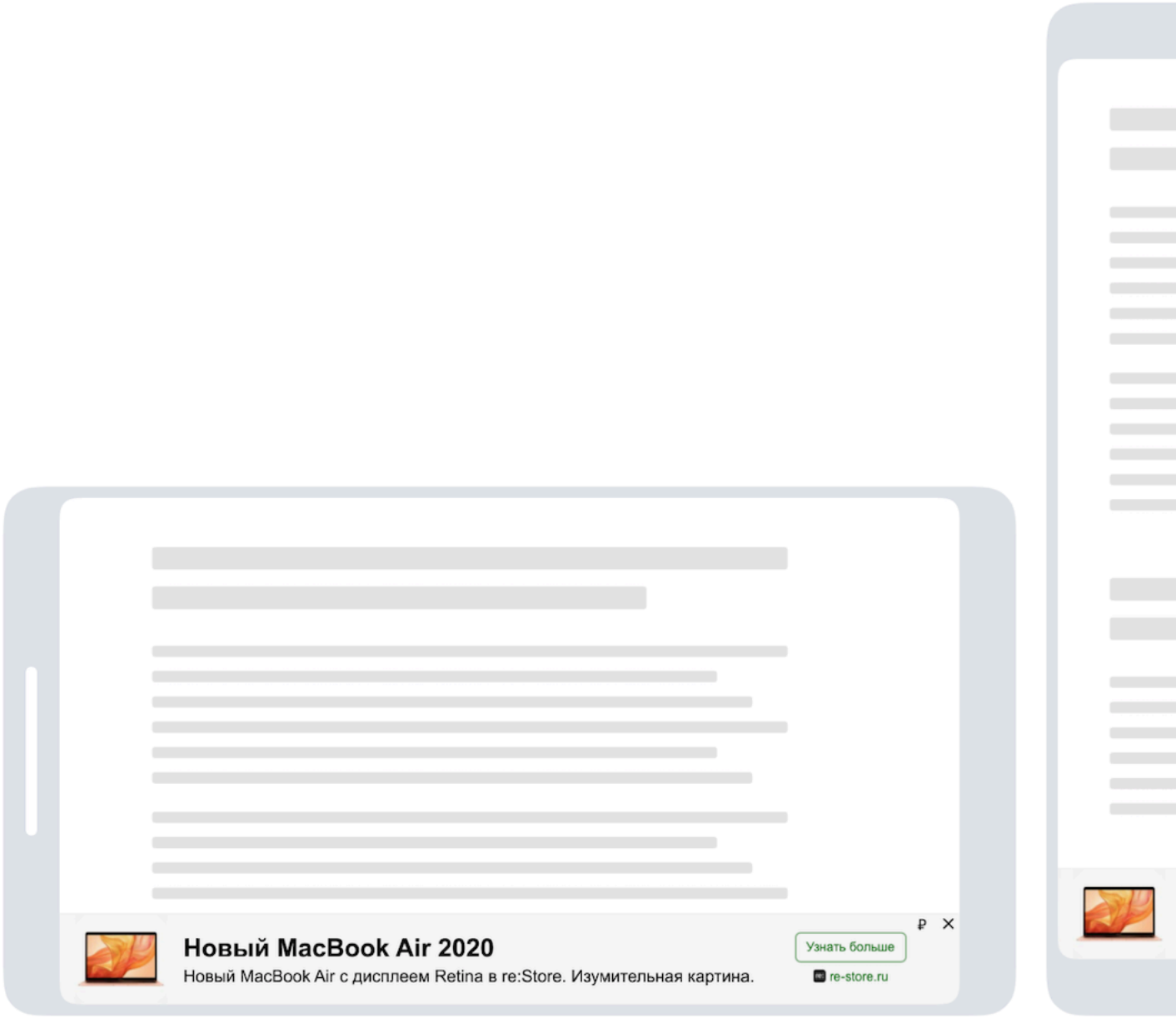
Типы баннера

Sticky баннер

Особенности:

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

Примеры отображения баннера:

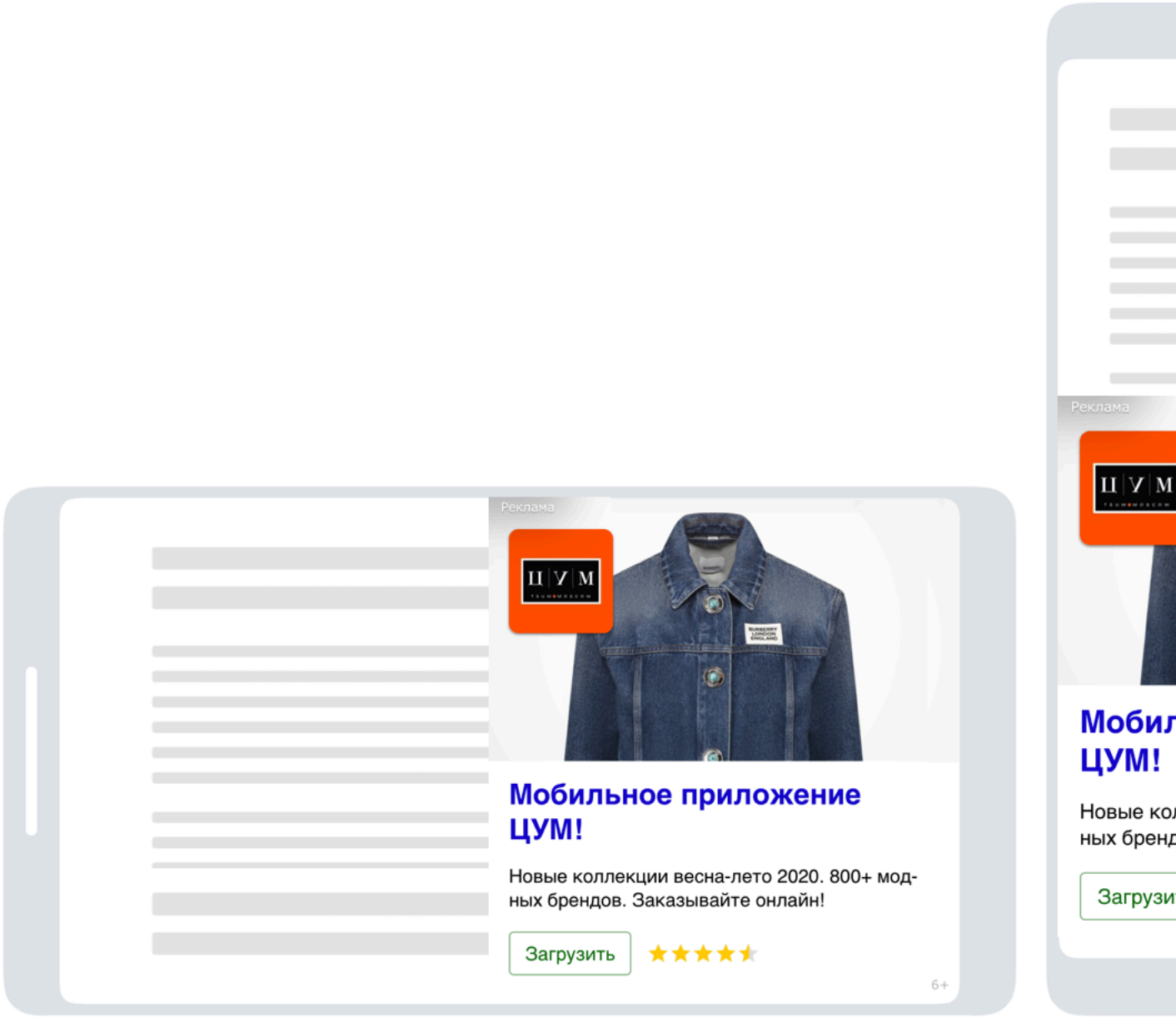


Flex баннер

Особенности:

1. Баннер заполняет весь блок, используя заданные максимальные размеры.
2. Ширина и высота баннера задается с помощью метода flexibleSize(int width, int height).

Примеры отображения баннера:



Подключение баннера

Создание BannerAdview

1. Добавьте объект класса `BannerAdView` в проект с помощью XML-файла или программно.

```
// Создание экземпляра mBannerAdView с помощью XML-файла.
        mBannerAdView = (BannerAdView) findViewById(R.id.banner_view);

// Создание экземпляра mBannerAdView программно.
        mBannerAdView = new BannerAdView(this);
```

2. Установите `AdUnitId`, используя метод setAdUnitId.

```
mBannerAdView.setAdUnitId(<AdUnitId>)
```

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

3. Установите размеры баннера, используя метод setAdSize.

   Sticky баннер

   Чтобы задать ширину баннера, вызовите метод stickySize(int width), где width — максимальная ширина баннера.

   ```
   mBannerAdView.setAdSize(AdSize.stickySize(width));
   ```

   Flex баннер

   Чтобы задать ширину и высоту баннера, вызовите метод flexibleSize(int width, int height).

   ```
   mBannerAdView.setAdSize(AdSize.flexibleSize(width, height));
   ```

   Ограничение: Требования к размерам баннера при отображении видеорекламы

   Минимальный размер баннера, в котором поддерживается воспроизведение видео: 300x160 или 160x300 в dp (density-independent pixels).
4. После создания и настройки объекта класса `BannerAdView` для отслеживания событий (открытие или закрытие рекламы, выход из приложения, успешная и неудачная загрузка рекламы) на объект рекламы можно установить слушатель AdEventListener.

Загрузка рекламы

Примечание:

Все вызовы Mobile Ads SDK следует выполнять с главного потока.

После создания и настройки объекта класса `BannerAdView`, рекламу необходимо загрузить. Для загрузки рекламы, используйте метод loadAd, принимающий в качестве параметра объект AdRequest (Builder, опционально принимающий данные для таргетирования рекламы).

Пример работы с баннерной рекламой

Следующий код демонстрирует создание и настройку объекта `BannerAdView`, регистрацию слушателя, а также загрузку баннера:

```
...
<LinearLayout>
    ...
    <com.yandex.mobile.ads.banner.BannerAdView
        android:id="@+id/banner_ad_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

```
...
private BannerAdView mBannerAdView;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        ...
        // Создание экземпляра mAdView.
        mBannerAdView = (BannerAdView) findViewById(R.id.banner_ad_view);
        mBannerAdView.setAdUnitId(AdUnitId);
        mBannerAdView.setAdSize(AdSize.stickySize(maxBannerWidthDp));

        // Создание объекта таргетирования рекламы.
        final AdRequest adRequest = new AdRequest.Builder().build();

        // Регистрация слушателя для отслеживания событий, происходящих в баннерной рекламе.
        mBannerAdView.setAdEventListener(new BannerAdEventListener() {
            @Override
            public void onAdLoaded() {
                ...
            }

            @Override
            public void onAdFailedToLoad(AdRequestError adRequestError) {
                ...
            }

            @Override
            public void onLeftApplication() {
                ...
            }

            @Override
            public void onReturnedToApplication() {
```

```
                ...
            }
        });

        // Загрузка объявления.
        mBannerAdView.loadAd(adRequest);
    }
}
```

Если реклама подключена данным образом, после запуска приложения появится баннер:


10:45
BannerExample
Yandex Mobile Ads

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

- demo-banner-yandex

Узнайте больше

Пример рекламы

Адаптивный баннер

Адаптивный баннер — это баннер, органично вписывающийся в размеры блока, определенные пользователем. В зависимости от способа интеграции адаптивного баннера, для него подбирается оптимальная высота с заданной шириной, либо используются указанные размеры рекламного места.

Примечание:

Про создание рекламного блока для адаптивного баннера можно прочитать в справке Рекламной сети.

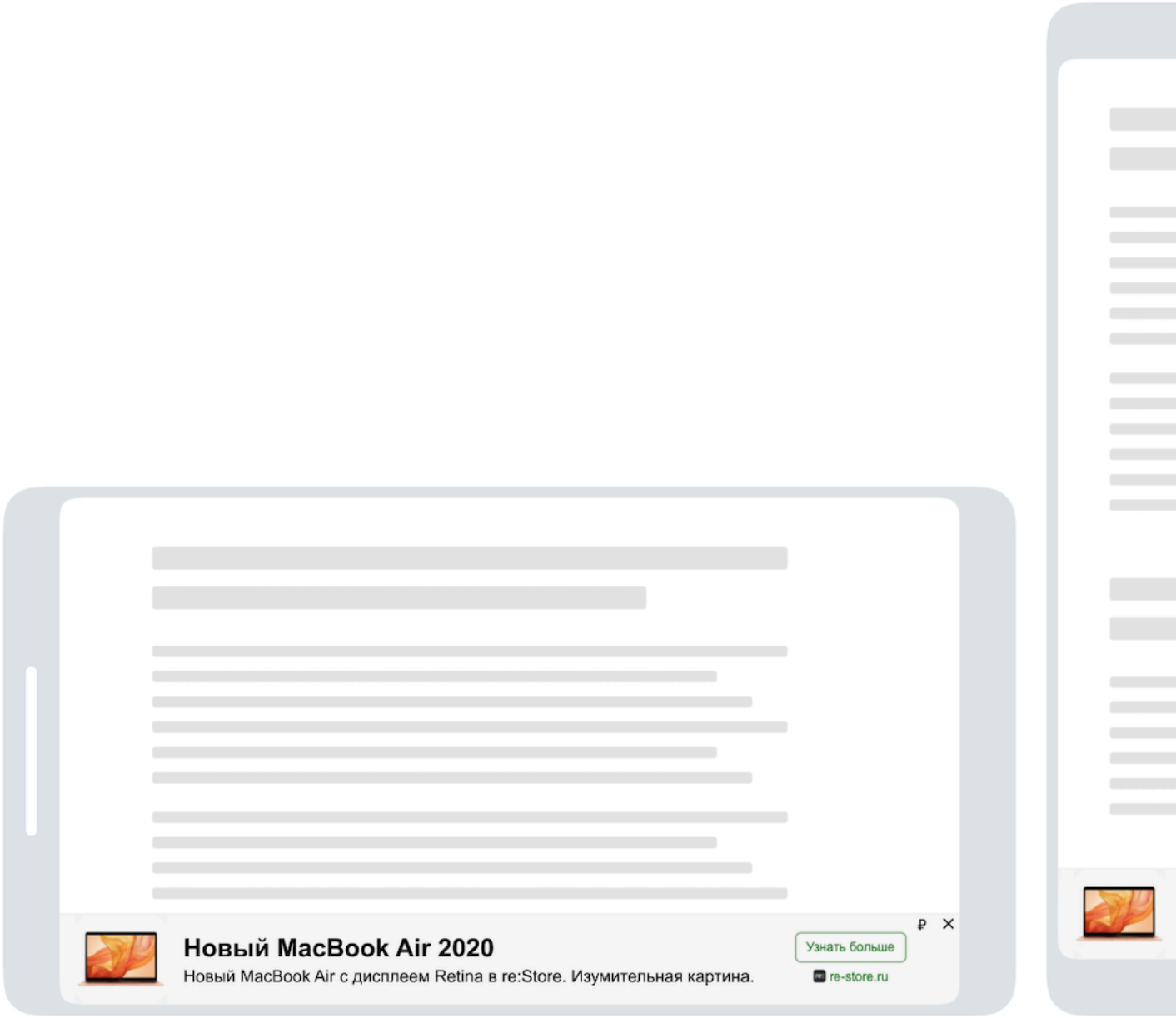
Типы адаптивного баннера

Баннер с заданной шириной

Особенности:

1. Альтернатива баннерам с размерами 320x50 (при определении высоты баннера сохраняется соотношение сторон 320x50).
2. Фиксация баннера в верхней или нижней части экрана (настраивается в приложении).
3. Использование заданной ширины баннера, а не ширины экрана устройства. Это позволяет учитывать особенности дисплея.
4. Ширина адаптивного баннера задается с помощью метода stickySize.

Примеры отображения адаптивного баннера:

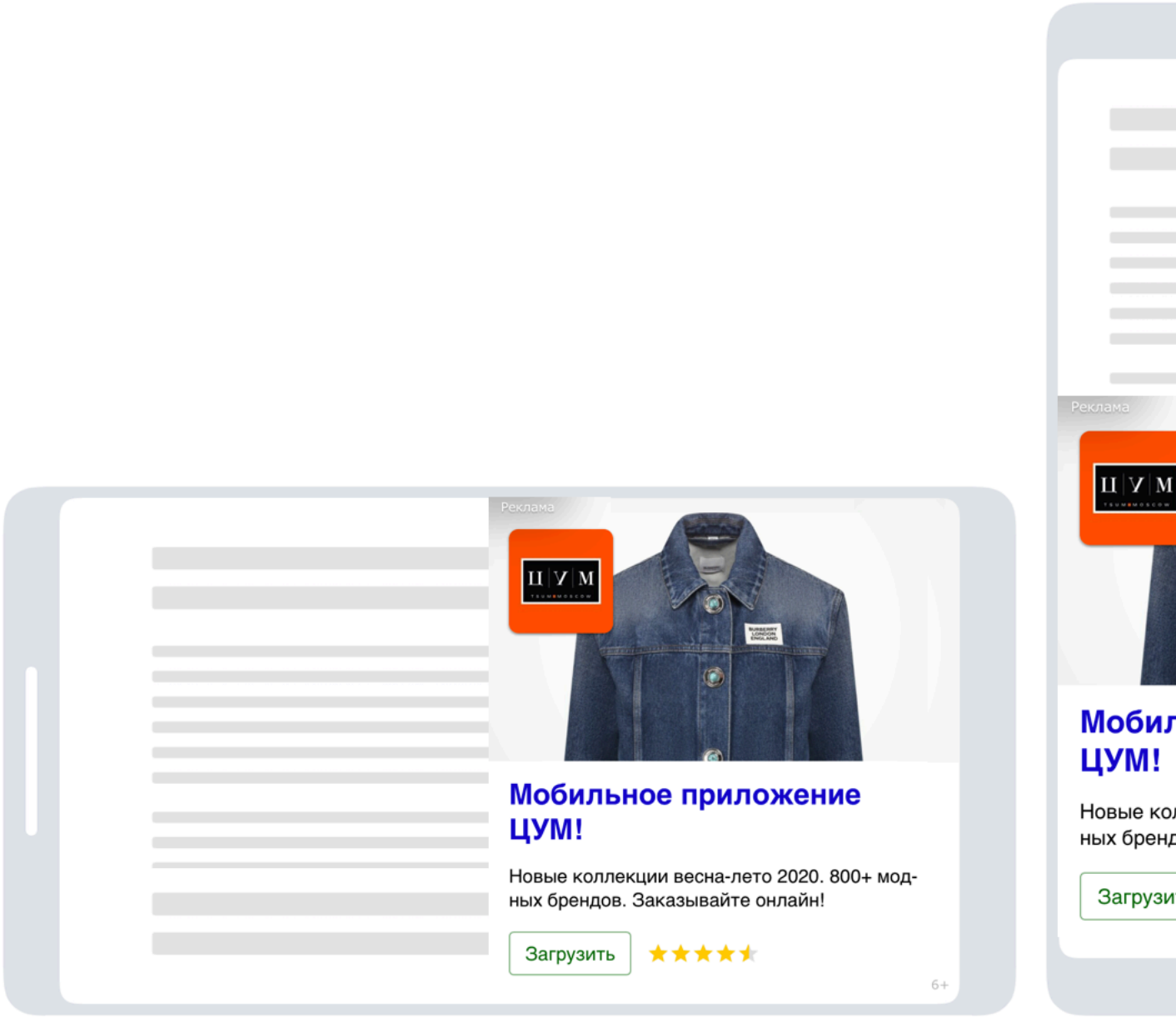


Баннер с заданной шириной и высотой

Особенности:

1. Адаптивный баннер заполняет весь блок, используя заданную ширину и высоту.
2. Ширина и высота адаптивного баннера задается с помощью метода flexibleSize(int width, int height).

Примеры отображения адаптивного баннера:



Создание BannerAdview

1. Добавьте объект класса `BannerAdView` в проект с помощью XML-файла или программно.

```
// Создание экземпляра mBannerAdView с помощью XML-файла.
        mBannerAdView = (BannerAdView) findViewById(R.id.banner_view);

// Создание экземпляра mBannerAdView программно.
        mBannerAdView = new BannerAdView(this);
```

2. Установите `AdUnitId`, используя метод setAdUnitId.

```
mBannerAdView.setAdUnitId(<AdUnitId>)
```

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

3. Установите размеры баннера, используя метод setAdSize.

   Баннер с заданной шириной

   Чтобы задать ширину адаптивного баннера, вызовите метод stickySize(int width).

   ```
   mBannerAdView.setAdSize(AdSize.stickySize(AdSize.FULL_WIDTH));
   ```

   Баннер с заданной шириной и высотой

   Чтобы задать ширину и высоту адаптивного баннера, вызовите метод flexibleSize(int width, int height).

   ```
   mBannerAdView.setAdSize(AdSize.flexibleSize(width, height));
   ```

4. После создания и настройки объекта класса `BannerAdView` для отслеживания событий (открытие или закрытие рекламы, выход из приложения, успешная и неудачная загрузка рекламы) на объект рекламы можно установить слушатель AdEventListener.

Загрузка рекламы

После создания и настройки объекта класса `BannerAdView`, рекламу необходимо загрузить. Для загрузки рекламы, используйте метод loadAd, принимающий в качестве параметра объект AdRequest (Builder, опционально принимающий данные для таргетирования рекламы).

Пример работы с адаптивным баннером

Следующий код демонстрирует создание и настройку объекта `AdView`, регистрацию слушателя, а также загрузку адаптивного баннера:

```
...
<LinearLayout>
    ...
    <com.yandex.mobile.ads.banner.BannerAdView
        android:id="@+id/banner_ad_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

```
...
private BannerAdView mBannerAdView;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        ...
        // Создание экземпляра mAdView.
        mBannerAdView = (BannerAdView) findViewById(R.id.banner_ad_view);
        mBannerAdView.setAdUnitId(AdUnitId);
        mBannerAdView.setAdSize(AdSize.stickySize(AdSize.FULL_WIDTH));

        // Создание объекта таргетирования рекламы.
        final AdRequest adRequest = new AdRequest.Builder().build();

        // Регистрация слушателя для отслеживания событий, происходящих в баннерной рекламе.
        mBannerAdView.setAdEventListener(new BannerAdEventListener() {
            @Override
            public void onAdLoaded() {
                ...
            }

            @Override
            public void onAdFailedToLoad(AdRequestError adRequestError) {
                ...
            }

            @Override
            public void onLeftApplication() {
                ...
            }

            @Override
            public void onReturnedToApplication() {
                ...
            }
        });

        // Загрузка объявления.
        mBannerAdView.loadAd(adRequest);
    }
}
```

Узнайте больше

Пример рекламы

Классы и интерфейсы для работы с баннерной рекламой

Внимание:

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

Классы

- AdSize
- BannerAdView

Интерфейсы

- BannerAdEventListener

Процесс подключения полноэкранной рекламы

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, необходимо:

Создать InterstitialAd
Загрузить рекламу
Отобразить рекламу

Создание InterstitialAd

1. Создайте объект класса InterstitialAd. Объект может быть создан только программно.

```
mInterstitialAd = new InterstitialAd(this);
```

2. Установите AdUnitId, используя метод setAdUnitId.

```
mInterstitialAd.setAdUnitId(AdUnitId);
```

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

3. После создания и настройки объекта класса InterstitialAd, для отслеживания событий (открытие или закрытие рекламы, выход из приложения, успешная и не успешная загрузка рекламы), на объект рекламы можно установить слушатель InterstitialEventListener интерфейса.

Загрузка рекламы

Примечание:

Все вызовы Mobile Ads SDK следует выполнять с главного потока.

После создания и настройки объекта класса InterstitialAd, рекламу необходимо загрузить. Для загрузки рекламы, используйте метод loadAd, принимающий в качестве параметра объект AdRequest (Builder, опционально принимающий данные для таргетирования рекламы).

Отображение рекламы

Загрузка полноэкранной рекламы происходит в фоновом потоке сразу после вызова метода loadAd. Чтобы показать полноэкранную рекламу необходимо вызвать метод show.

Рекомендуется предварительно проверить, что реклама действительно загружена. Для этого вызовите метод isLoaded.

Примечание:

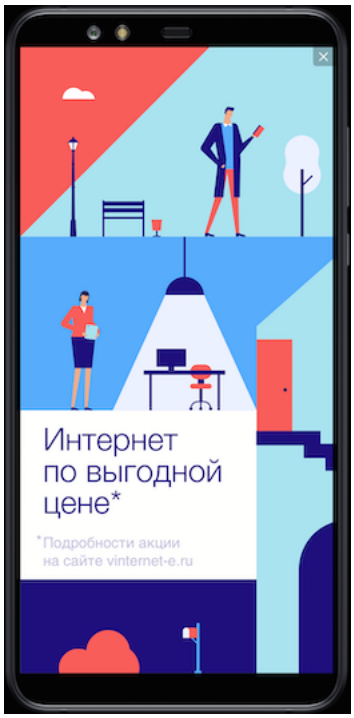
Такая проверка не требуется, если метод show вызывается после срабатывания callback об окончании загрузки onAdLoaded.

Пример работы с полноэкранной рекламой

Следующий код демонстрирует создание и настройку объекта InterstitialAd, регистрацию слушателя, а также загрузку и отображение полноэкранной рекламы:

```
...
public class InterstitialExample extends Activity {
    ...
    private static final String AdUnitId = "YOUR_AdUnitId";
    private InterstitialAd mInterstitialAd;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        ...
        // Создание экземпляра InterstitialAd.
        mInterstitialAd = new InterstitialAd(this);
        mInterstitialAd.setAdUnitId(AdUnitId);

        // Создание объекта таргетирования рекламы.
        final AdRequest adRequest = new AdRequest.Builder().build();

        // Регистрация слушателя для отслеживания событий, происходящих в рекламе.
        mInterstitialAd.setInterstitialAdEventListener(new InterstitialAdEventListener() {
            @Override
            public void onAdLoaded() {
                mInterstitialAd.show();
            }

            @Override
            public void onAdFailedToLoad(AdRequestError adRequestError) {
                ...
            }

            @Override
            public void onAdShown() {
                ...
            }

            @Override
            public void onAdDismissed() {
                ...
            }

            @Override
            public void onLeftApplication() {
                ...
            }

            @Override
            public void onReturnedToApplication() {
                ...
            }
        });

        // Загрузка объявления.
        mInterstitialAd.loadAd(adRequest);
    }
}
```

Если реклама подключена данным образом, после запуска приложения появится рекламный блок:



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

- demo-interstitial-yandex

Узнайте больше
Пример рекламы

Классы и интерфейсы для работы с полноэкранной рекламой

Внимание:

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

Классы

- InterstitialAd

Интерфейсы

- InterstitialAdEventListener

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:

Создать RewardedAd
Загрузить рекламу
Отобразить рекламу

Создание RewardedAd

1. Добавьте объект класса RewardedAd.

```
mRewardedAd = new RewardedAd(this);
```

2. Установите AdUnitId, используя метод setAdUnitId.

```
mRewardedAd.setAdUnitId(AdUnitId);
```

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

3. Если вы используете механизм награждения на стороне приложения («client-side reward»), реализуйте метод интерфейса onRewarded. Он вызывается, когда показ засчитан и пользователь может быть награжден за просмотр рекламы. Используйте этот момент, чтобы выдать награду пользователю приложения.

После создания и настройки объекта класса RewardedAd, для отслеживания событий (открытие или закрытие рекламы, выход из приложения, успешная и не успешная загрузка рекламы), на объект рекламы необходимо установить слушатель RewardedAdEventListener интерфейса.

Загрузка рекламы

Примечание:

Все вызовы Mobile Ads SDK следует выполнять с главного потока.

После создания и настройки объекта класса RewardedAd, рекламу необходимо загрузить. Для загрузки рекламы, используйте метод loadAd, принимающий в качестве параметра объект AdRequest (Builder, опционально принимающий данные для таргетирования рекламы).

Отображение рекламы

Загрузка рекламы с вознаграждением происходит в фоновом потоке сразу после вызова метода loadAd. Чтобы показать рекламу с вознаграждением необходимо вызвать метод show.

Рекомендуется предварительно проверить, что реклама действительно загружена. Для этого вызовите метод isLoaded.

Примечание:

Такая проверка не требуется, если метод show вызывается после срабатывания callback об окончании загрузки onAdLoaded.

Пример работы с рекламой с вознаграждением

Следующий код демонстрирует создание и настройку объекта RewardedAd, регистрацию слушателя, а также загрузку и отображение рекламы с вознаграждением:

```
...
public class RewardedAdExample extends Activity {
    ...
    private static final String AdUnitId = "YOUR_AdUnitId";
    private RewardedAd mRewardedAd;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        ...
        // Создание экземпляра RewardedAd.
        mRewardedAd = new RewardedAd(this);
        mRewardedAd.setAdUnitId(AdUnitId);

        // Создание объекта таргетирования рекламы.
        final AdRequest adRequest = new AdRequest.Builder().build();

        // Регистрация слушателя для отслеживания событий, происходящих в рекламе.
        mRewardedAd.setRewardedAdEventListener(new RewardedAdEventListener() {
            @Override
            public void onLoaded() {
                mRewardedAd.show();
            }

            @Override
            public void onRewarded(final Reward reward) {
```

```
                ...
            }

            @Override
            public void onAdFailedToLoad(final AdRequestError adRequestError) {
                ...
            }

            @Override
            public void onAdShown() {
                ...
            }

            @Override
            public void onAdDismissed() {
                ...
            }

            @Override
            public void onLeftApplication() {
                ...
            }

            @Override
            public void onReturnedToApplication() {
                ...
            }
        });

        // Загрузка объявления.
        mRewardedAd.loadAd(adRequest);
    }
}
```

Если реклама подключена данным образом, после запуска приложения появится рекламный блок.

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

- `demo-rewarded-yandex`

Классы и интерфейсы для работы с рекламой с вознаграждением

Внимание:

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

Классы

- RewardedAd

Интерфейсы

- Reward
- RewardedAdEventListener

Нативная реклама

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.

Есть три типа загрузки рекламы:

- Загрузка одного объявления
- Загрузка нескольких объявлений
- Слайдер рекламных объявлений

Загрузка и показ рекламных объявлений

Внимание:

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

Подготовка к работе

Для корректной работы Yandex Mobile Ads SDK выполните все шаги подключения рекламной библиотеки.

Загрузка рекламных объявлений

Примечание:

Все вызовы Mobile Ads SDK следует выполнять с главного потока.

1. Создайте экземпляр класса NativeAdLoader для получения нативных рекламных объявлений.
2. Создайте конфигурацию запроса `nativeAdRequestConfiguration` с помощью класса NativeAdRequestConfiguration.Builder. В качестве параметров запроса можно передать идентификатор рекламного блока, способ загрузки изображений, возраст, гендерные признаки и другие данные, способные улучшить качество подбора рекламы.
3. Для получения уведомлений (успешная загрузка рекламы или ошибка при загрузке рекламы), создайте экземпляр NativeAdLoadListener и установите его в качестве слушателя событий загрузчика рекламных объявлений.
4. Запустите процесс загрузки рекламы.

Если на блоке включены медиа форматы рекламы, следует запрашивать рекламу с указанием размеров контейнера.

Общий запрос за рекламой

```
final NativeAdLoader nativeAdLoader = new NativeAdLoader(this);
    nativeAdLoader.setNativeAdLoadListener(new NativeAdLoadListener() {
        @Override
        public void onAdLoaded(@NonNull final NativeAd nativeAd) {
            //bind nativeAd
        }

        @Override
        public void onAdFailedToLoad(@NonNull final AdRequestError error) {
            //log error
        }
    });

final NativeAdRequestConfiguration nativeAdRequestConfiguration =
    new NativeAdRequestConfiguration.Builder(AdUnitId).build();
nativeAdLoader.loadAd(nativeAdRequestConfiguration);
```

Совет:

Рекомендуется держать сильную ссылку на рекламу и ее загрузчик на всем протяжении жизни экрана, в рамках которого происходит работа с данными компонентами.

Показ рекламных объявлений

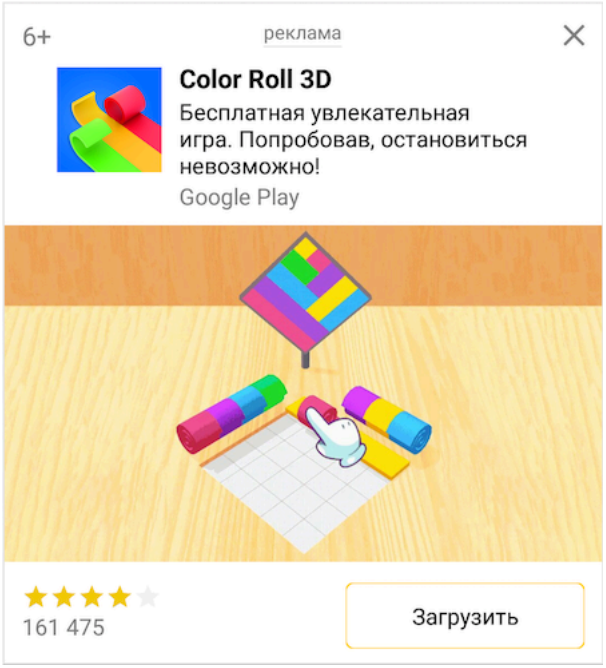
После окончания загрузки рекламы, все ее компоненты необходимо отобразить. Список доступных в объявлении компонентов можно получить из объекта рекламы NativeAd.

Существует два способа настройки внешнего оформления рекламы:

1. Оформление с помощью шаблона
2. Оформление без использования шаблона

Оформление с помощью шаблона

Пример шаблона нативной рекламы



Использование стандартного шаблона оформления — самый простой способ работы с нативной рекламой, который требует всего нескольких строк кода в базовом варианте.

В шаблоне уже добавлен весь набор необходимых рекламных компонентов и настроено их расположение относительно друг друга. Шаблон работает с любым поддерживаемым типом нативной рекламы.

```
final NativeBannerView nativeBannerView = new NativeBannerView(this);
nativeBannerView.setAd(nativeAd);
```

Шаблон нативной рекламы можно кастомизировать. Подробнее об этом в статье Пример оформления с помощью шаблона.

Оформление без использования шаблона

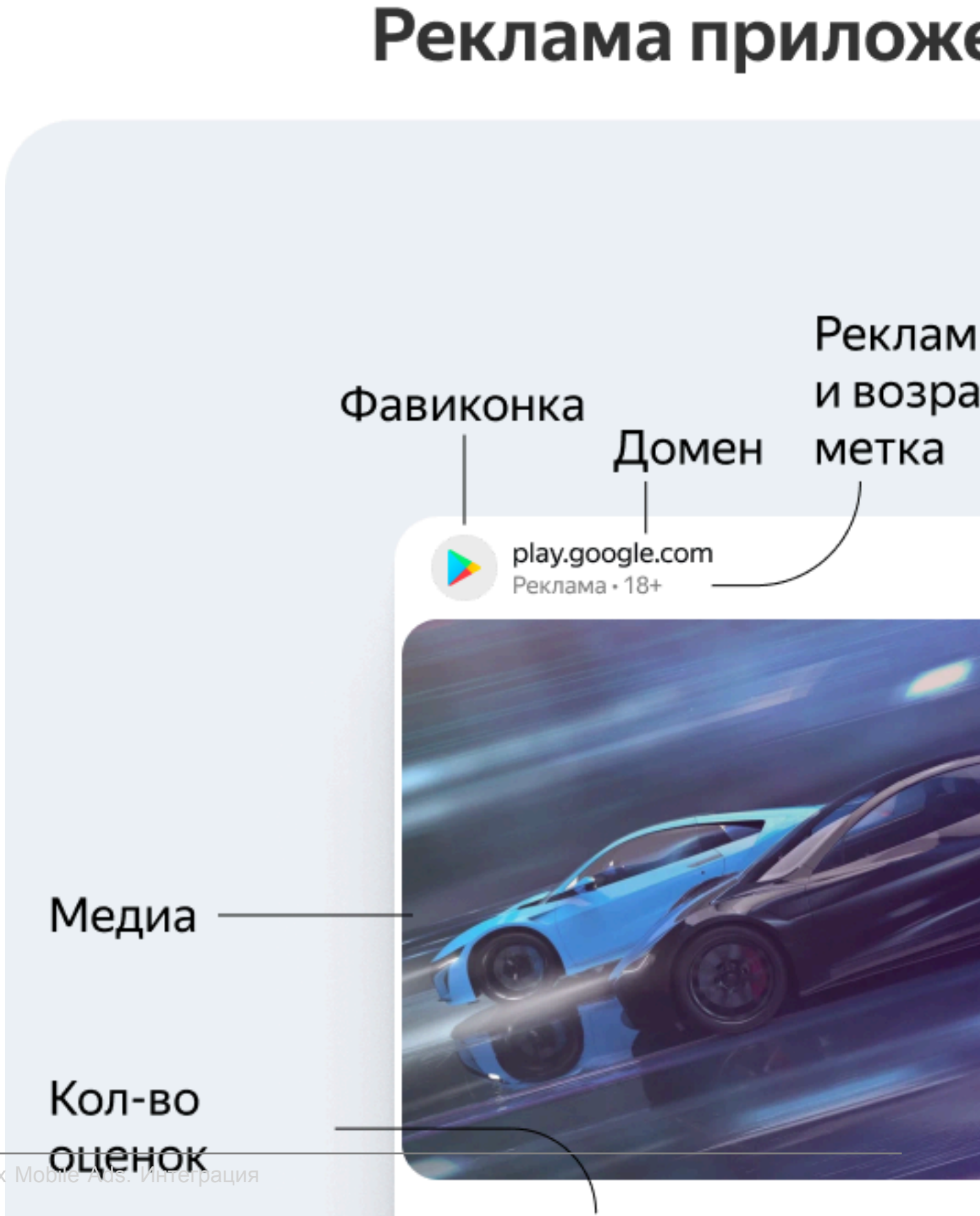
Ручная настройка внешнего оформления нативной рекламы используется в тех случаях, когда возможностей настройки шаблона недостаточно для получения желаемого результата.

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.

Пример оформления без использования шаблона



Для каждого компонента рекламного объявления предоставьте View через экземпляр класса NativeAdViewBinder.Builder. Класс принимает в качестве аргумента контейнер NativeAdView. Все компоненты рекламы должны быть определены как subview данного контейнера.

Сформированный макет рекламного объявления необходимо связать с объектом нативной рекламы NativeAd.

```
final NativeAdViewBinder nativeAdViewBinder = new NativeAdViewBinder.Builder(mNativeAdView)
                    .setAgeView((TextView) findViewById(R.id.age))
                    .setBodyView((TextView) findViewById(R.id.body))
                    .setCallToActionView((TextView) findViewById(R.id.call_to_action))
                    .setDomainView((TextView) findViewById(R.id.domain))
                    .setFaviconView((ImageView) findViewById(R.id.favicon))
                    .setFeedbackView((TextView) findViewById(R.id.feedback))
                    .setIconView((ImageView) findViewById(R.id.icon))
                    .setMediaView((MediaView) findViewById(R.id.media))
                    .setPriceView((TextView) findViewById(R.id.price))
                    .setRatingView((MyRatingView) findViewById(R.id.rating))
                    .setReviewCountView((TextView) findViewById(R.id.review_count))
                    .setSponsoredView((TextView) findViewById(R.id.sponsored))
                    .setTitleView((TextView) findViewById(R.id.title))
                    .setWarningView((TextView) findViewById(R.id.warning))
                    .build();

                    try {
                    nativeAd.bindNativeAd(nativeAdViewBinder);
                    mNativeAdView.setVisibility(View.VISIBLE);
                    } catch (final NativeAdException exception) {
                    //log exception
                    }
```

Примечание: Требования к размерам mediaView при отображении видеорекламы

Минимальный размер экземпляра класса MediaView, в котором поддерживается воспроизведение видео: 300x160 или 160x300 в dp (density-independent pixels).

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

Загрузка нескольких рекламных объявлений

Примечание:

Все вызовы Mobile Ads SDK следует выполнять с главного потока.

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

Для балкового запроса за рекламой воспользуйтесь экземпляром класса NativeBulkAdLoader. Экземпляр класса принимает в методе loadAds аргумент COUNT, определяющий желаемое количество объявлений за один запрос.

```
final NativeBulkAdLoader nativeBulkAdLoader = new NativeBulkAdLoader(this);
    nativeBulkAdLoader.setNativeBulkAdLoadListener(new NativeBulkAdLoadListener() {
        @Override
        public void onAdsLoaded(@NonNull final List<NativeAd> nativeAds) {
            for (final NativeAd nativeAd : nativeAds) {
                //bind native ad
            }
        }

        @Override
        public void onAdsFailedToLoad(@NonNull final AdRequestError error) {
            //log error
        }
    });

    final NativeAdRequestConfiguration nativeAdRequestConfiguration =
    new NativeAdRequestConfiguration.Builder(AdUnitId).build();
    nativeBulkAdLoader.loadAds(nativeAdRequestConfiguration, COUNT);
```

Примечание:

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

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.

Подготовка к работе

Для корректной работы Yandex Mobile Ads SDK выполните все шаги подключения рекламной библиотеки.

Загрузка слайдера

Примечание:

Все вызовы Mobile Ads SDK следует выполнять с главного потока.

1. Создайте экземпляр класса SliderAdLoader для получения рекламных объявлений в рамках слайдера.
2. Создайте конфигурацию запроса `nativeAdRequestConfiguration` с помощью класса NativeAdRequestConfiguration.Builder. В качестве параметров запроса можно передать идентификатор рекламного блока, способ загрузки изображений, возраст, гендерные признаки и другие данные, способные улучшить качество подбора рекламы.
3. Для получения уведомлений (успешная загрузка слайдера или ошибка при загрузке), создайте экземпляр SliderAdLoadListener и установите его в качестве слушателя событий загрузчика рекламных объявлений.
4. Запустите процесс загрузки слайдера.

Пример кода:

```
final SliderAdLoader sliderAdLoader = new SliderAdLoader(this);
sliderAdLoader.setSliderAdLoadListener(new SliderAdLoadListener() {
    @Override
    public void onSliderAdLoaded(@NonNull final SliderAd sliderAd) {
        //bind sliderAd
    }

    @Override
    public void onSliderAdFailedToLoad(@NonNull final AdRequestError error) {
        //log error
    }
});

final NativeAdRequestConfiguration nativeAdRequestConfiguration =
        new NativeAdRequestConfiguration.Builder(AdUnitId).build();
sliderAdLoader.loadSlider(nativeAdRequestConfiguration);
```

Совет:

Рекомендуется держать сильную ссылку на slider и его загрузчик на всем протяжении жизни экрана, в рамках которого происходит работа с данными компонентами.

Показ слайдера

После окончания загрузки слайдера, все его компоненты необходимо отобразить.

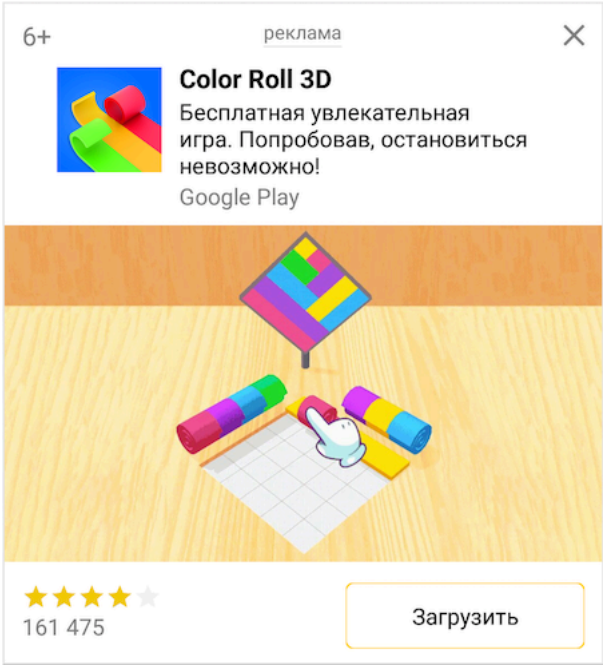
Успешно загруженный слайдер содержит одно или несколько связанных между собой нативных объявлений. Показ рекламных объявлений в рамках слайдера должен происходить в пределах одного общего контейнера, иначе показ объявлений не будет засчитан.

Существует два способа настройки внешнего оформления слайдера:

1. Оформление с помощью шаблона
2. Оформление без использования шаблона

Оформление с помощью шаблона

Пример шаблона нативной рекламы



Использование стандартного шаблона оформления — самый простой способ работы с нативной рекламой, который требует всего нескольких строк кода в базовом варианте.

В шаблоне уже добавлен весь набор необходимых рекламных компонентов и настроено их расположение относительно друг друга. Шаблон работает с любым поддерживаемым типом нативной рекламы.

Оформить слайдер с помощью шаблона очень просто. Привяжите объект SliderAd к корневому контейнеру NativeAdView, а все входящие в слайдер рекламные объявления свяжите с шаблоном:

```
@Override
public void onSliderAdLoaded(@NonNull final SliderAd sliderAd) {
    mNativeAdView = (NativeAdView) findViewById(R.id.native_slider_ad_container);
    try {
        final NativeAdViewBinder sliderViewBinder =
            new NativeAdViewBinder.Builder(mNativeAdView).build();
sliderAd.bindSliderAd(sliderViewBinder);
        final List<NativeAd> nativeAds = sliderAd.getNativeAds();
        for (final NativeAd nativeAd : nativeAds) {
            final NativeBannerView nativeBannerView = new NativeBannerView(this);
            nativeBannerView.setAd(nativeAd);
            mNativeAdView.addView(nativeBannerView);
        }
    } catch (final NativeAdException exception) {
        //log error
    }
}
```

Шаблон нативной рекламы можно кастомизировать. Подробнее об этом в статье Пример оформления с помощью шаблона.

Оформление без использования шаблона

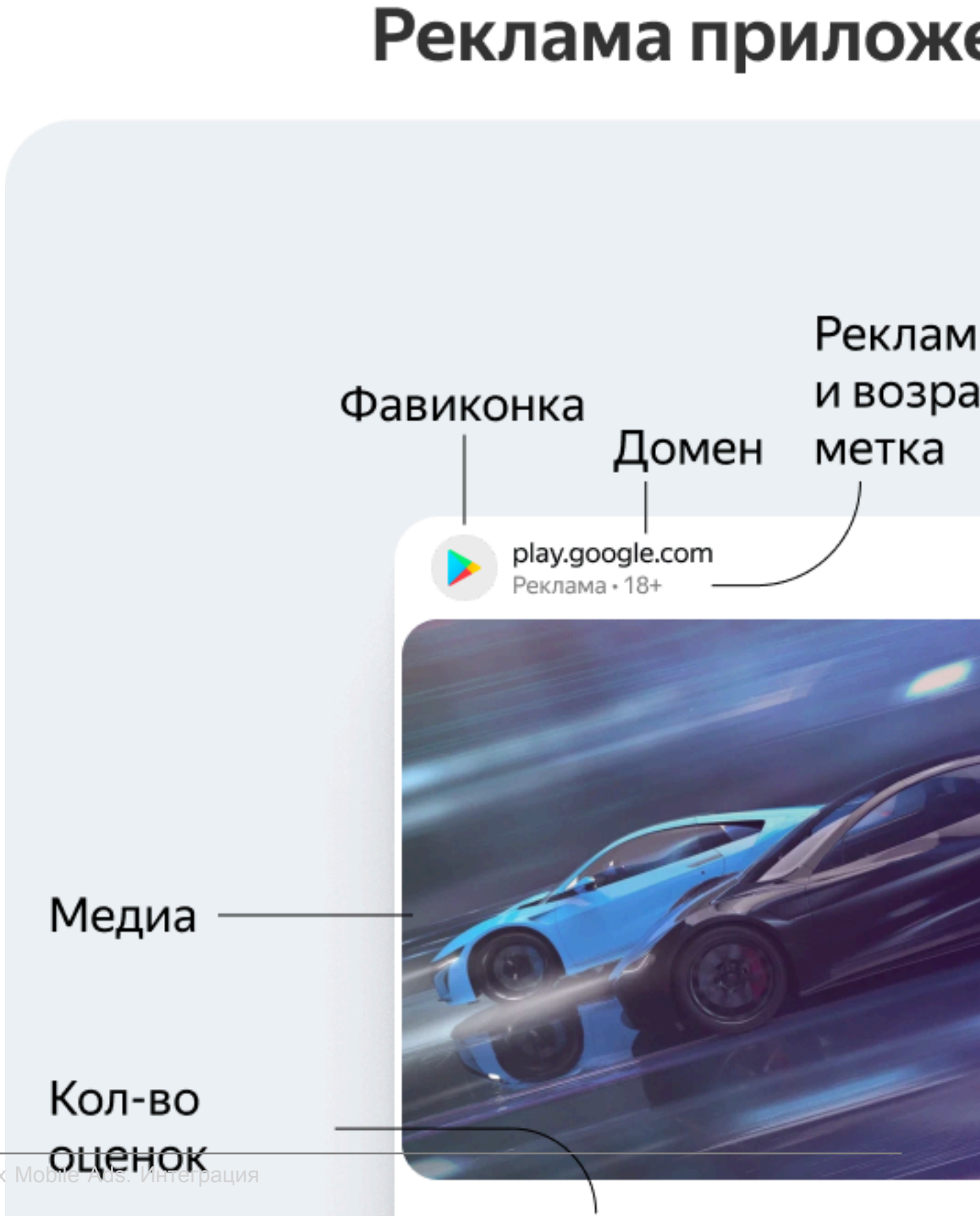
Ручная настройка внешнего оформления нативной рекламы используется в тех случаях, когда возможностей настройки шаблона недостаточно для получения желаемого результата.

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.

Пример оформления без использования шаблона



Привяжите объект SliderAd к корневому NativeAdView, а каждое объявление, входящее в слайдер, к дочернему NativeAdView.

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

```
@Override
public void onSliderAdLoaded(@NonNull final SliderAd sliderAd) {
    mNativeAdView = (NativeAdView) findViewById(R.id.native_slider_ad_container);
    try {
        final NativeAdViewBinder sliderViewBinder =
            new NativeAdViewBinder.Builder(mNativeAdView).build();
sliderAd.bindSliderAd(sliderViewBinder);
        final List<NativeAd> nativeAds = sliderAd.getNativeAds();
        for (final NativeAd nativeAd : nativeAds) {
            final NativeAdView nativeAdView = mLayoutInflater.inflate(R.layout.widget_native_ad, mSliderAdView,
 false);
            final NativeAdViewBinder viewBinder = new NativeAdViewBinder.Builder(nativeAdView)
                    .setAgeView((TextView) nativeAdView.findViewById(R.id.age))
                    .setBodyView((TextView) nativeAdView.findViewById(R.id.body))
                    .setCallToActionView((Button) nativeAdView.findViewById(R.id.call_to_action))
                    .setDomainView((TextView) nativeAdView.findViewById(R.id.domain))
                    .setFaviconView((ImageView) nativeAdView.findViewById(R.id.favicon))
                    .setFeedbackView((Button) nativeAdView.findViewById(R.id.feedback))
                    .setIconView((ImageView) nativeAdView.findViewById(R.id.icon))
                    .setMediaView((MediaView) nativeAdView.findViewById(R.id.media))
                    .setPriceView((TextView) nativeAdView.findViewById(R.id.price))
                    .setRatingView((MyRatingView) nativeAdView.findViewById(R.id.rating))
                    .setReviewCountView((TextView) nativeAdView.findViewById(R.id.review_count))
                    .setSponsoredView((TextView) nativeAdView.findViewById(R.id.sponsored))
                    .setTitleView((TextView) nativeAdView.findViewById(R.id.title))
                    .setWarningView((TextView) nativeAdView.findViewById(R.id.warning))
                    .build();

            nativeAd.bindNativeAd(viewBinder);
            mNativeAdView.addView(nativeBannerView);
        }
    } catch (final NativeAdException exception) {
        //log error
    }
}
```

Примечание: Требования к размерам mediaView при отображении видеорекламы

Минимальный размер экземпляра класса MediaView, в котором поддерживается воспроизведение видео: 300x160 или 160x300 в dp (density-independent pixels).

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

Требования к размещению рекламы и рекламных компонентов

Внимание:

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

Основные требования

Показ рекламы не будет засчитан, если хотя бы одно из перечисленных условий не выполнено:

1. Рекламная `View`, а также каждая из ее `superview` должны удовлетворять следующим условиям:
   - `View` должна быть видима;
   - `View` должна быть непрозрачна.
2. Все обязательные рекламные компоненты должны удовлетворять следующим условиям:
   - рекламный компонент, а также каждый из его `superview` должны быть полностью видимы;
   - рекламный компонент должен полностью находиться в пределах рекламной `View`;
   - рекламный компонент должен находиться в иерархии рекламной `View`;
   - рекламный компонент должен быть непрозрачен;
   - содержимое рекламного компонента должно соответствовать действительности.
3. Приложение должно находиться в активном состоянии (не в фоне).

4. В единый момент времени загруженное рекламное объявление может быть показано только в одной `View`. Одновременное отображение одного рекламного объявления в нескольких `View` может привести к потере показа.

Примечание: Если все условия выполнены, а показ не засчитан, то попытка его засчитать будет произведена позже.

Ограничения на изменение рекламных компонентов

1. Нельзя редактировать текстовое содержимое компонентов.
2. Нельзя редактировать содержимое изображения.
3. Растягивать изображение можно только с соблюдением пропорций.
4. Обрезать изображение можно не более чем на 20%. Допустимые инструменты: наложение маски, изменение размеров контейнера.

Компоненты нативной рекламы

Внимание:

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

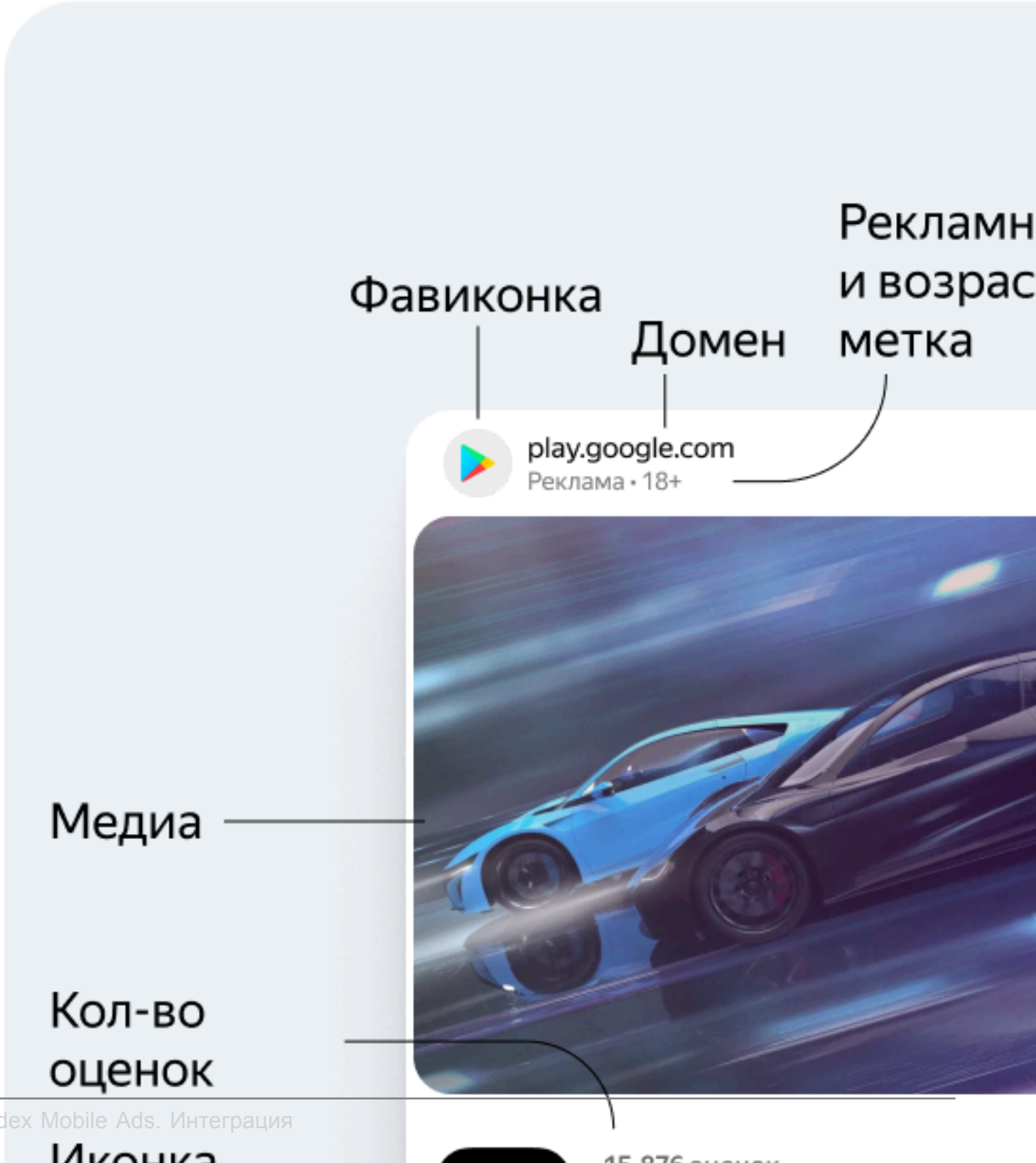
Примечание:

Ознакомьтесь с требованиями к размещению рекламы и рекламных компонентов.

Список компонентов

В библиотеке есть обязательные и опциональные компоненты. Правила размещения рекламных объявлений определяют необходимость отображения только обязательных компонентов. На практике, макеты с полным набором компонентов более кликабельные. Поэтому рекомендуется использовать дизайн, который включает весь набор возможных компонентов.

Реклама приложе



Элемент в объявлении	Компонент	Тип	Обязательность
Заголовок	title	TextView	Да
Домен	domain	TextView	Да
Предупреждение	warning	TextView	Да
Рекламная и возрастная метка	sponsored	TextView	Да
Значок меню	feedback	ImageView	Да
Кнопка действия	callToAction	TextView	Да
Медиа	media	MediaView	Да
Иконка приложения	icon	ImageView	Да, для РМП
Цена	price	TextView	Да, для РМП
Фавиконка	favicon	ImageView	Нет
Количество оценок	reviewCount	TextView	Нет
Рейтинг	rating	View implements Rating interface	Нет
Текст	body	TextView	Нет

Список обязательных компонентов описывает набор данных, для которых, в случае наличия этих компонентов, должна быть предоставлена View для отображения.

Совет:

Рекомендуется использовать макет, который сможет отобразить весь набор обязательных и опциональных компонентов рекламы.

Оформление компонентов

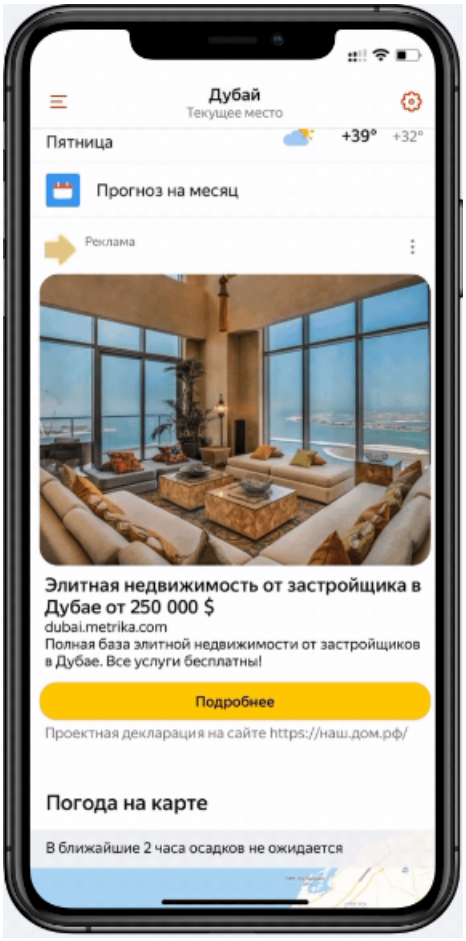
feedback

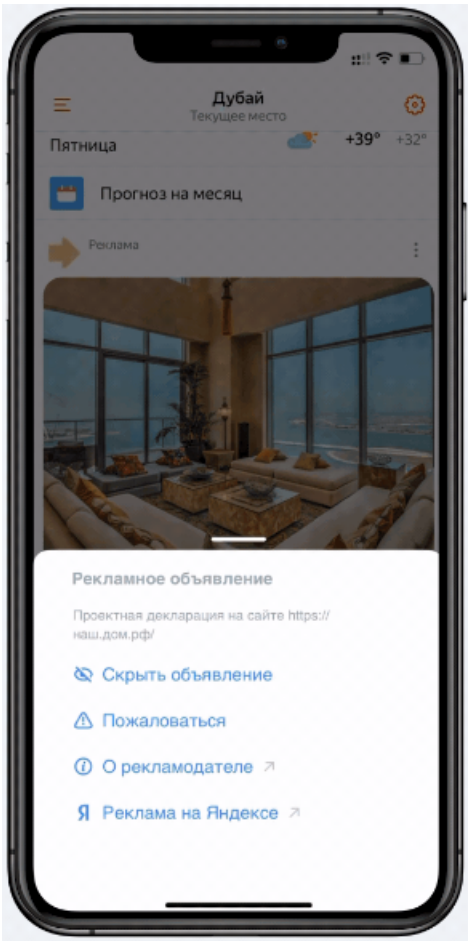
Значок меню. Обязательный компонент. С помощью значка меню пользователь может скрыть объявление или пожаловаться на него.

Значок меню добавляется в правый верхний угол объявления.

Примечание:

Разработчику заранее необходимо определить что делать с объявлением после выбора причины закрытия (например, скрыть объявление или показать какую-то надпись). Если дальнейшее действие не определено, SDK учтет причину закрытия, но объявление не будет скрыто.





Варианты оформления:

feedback_dark_dots_with_background

Белый значок меню, с темными точками и полупрозрачной подложкой. Значение по умолчанию.

feedback_light_dots

Значок меню без фона, со светлыми точками.

Пример кода:

```
final NativeAdLoader loader = new NativeAdLoader(this);
final HashMap<String, String> parameters = new HashMap<String, String>(){{
 put("feedback_image", "feedback_light_dots");
}};
final NativeAdRequestConfiguration nativeAdRequestConfiguration =
 new NativeAdRequestConfiguration.Builder("demo-native-app-yandex")
  .setParameters(parameters)
mNativeAdLoader.loadAd(nativeAdRequestConfiguration);
```

feedback_dark_dots

Значок меню без фона, с темными точками.

Пример кода:

```
final NativeAdLoader loader = new NativeAdLoader(this);
final HashMap<String, String> parameters = new HashMap<String, String>(){{
 put("feedback_image", "feedback_dark_dots");
}};
final NativeAdRequestConfiguration nativeAdRequestConfiguration =
 new NativeAdRequestConfiguration.Builder("demo-native-app-yandex")
  .setParameters(parameters)
mNativeAdLoader.loadAd(nativeAdRequestConfiguration);
```

Пример оформления с помощью шаблона

Внимание:

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

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

Создание своего оформления на основе шаблона

1. Создайте объект — экземпляр класса NativeBannerView и установите для него предпочитаемые настройки:

```
final NativeBannerView nativeBannerView =
      new NativeBannerView(getApplicationContext());
final NativeTemplateAppearance nativeTemplateAppearance =
      new NativeTemplateAppearance.Builder()
                .withBannerAppearance(new BannerAppearance.Builder()
                        .setBackgroundColor(Color.GRAY).build())
                .withTitleAppearance(new TextAppearance.Builder()
                        .setTextColor(Color.BLUE).build())
                ...
                .build();
nativeBannerView.applyAppearance(nativeTemplateAppearance);
```

Примечание:

Экземпляр класса NativeBannerView может быть создан как программно, так и с помощью XML-файла.

Пример настроек оформления

```
final NativeBannerView nativeBannerView = new NativeBannerView(getApplicationContext());
final NativeTemplateAppearance nativeTemplateAppearance =
      new NativeTemplateAppearance.Builder()
              // Задаем цвет для рамки рекламного объявления.
              .withBannerAppearance(new BannerAppearance.Builder()
                      .setBorderColor(Color.YELLOW).build())

              // Задаем параметры кнопки.
              .withCallToActionAppearance(new ButtonAppearance.Builder()
              // задаем цвет и размер шрифта для надписи на кнопке с действием.
                      .setTextAppearance(new TextAppearance.Builder()
                              .setTextColor(Color.BLUE)
                              .setTextSize(14f).build())

                      // Задаем цвет кнопки для обычного и нажатого состояния.
                      .setNormalColor(Color.TRANSPARENT)
                      .setPressedColor(Color.GRAY)
                      // Задаем цвет и толщину обводки кнопки.
                      .setBorderColor(Color.BLUE)
                      .setBorderWidth(1f).build())

              // Задаем ширину изображения и правило формирования размера.
              .withImageAppearance(new ImageAppearance.Builder()
                      .setWidthConstraint(new SizeConstraint(SizeConstraint
                      .SizeConstraintType.FIXED, 60f)).build())

             // Задаем размер и цвет шрифта для надписи с возрастным ограничением.
             .withAgeAppearance(new TextAppearance.Builder()
                      .setTextColor(Color.GRAY)
                      .setTextSize(12f).build())

             // Задаем размер и цвет шрифта для основного рекламного текста.
             .withBodyAppearance(new TextAppearance.Builder()
                      .setTextColor(Color.GRAY)
                      .setTextSize(12f).build())

             // Задаем цвет для закрашенных звезд в рейтинге.
             .withRatingAppearance(new RatingAppearance.Builder()
                      .setProgressStarColor(Color.CYAN).build())

             // Задаем размер и цвет шрифта для заголовка рекламного объявления.
             .withTitleAppearance(new TextAppearance.Builder()
                      .setTextColor(Color.BLACK)
                      .setTextSize(14f).build())

             .build();

// Применяем настройки.
nativeBannerView.applyAppearance(nativeTemplateAppearance);
```

Примечание:

При создании своего оформления на основе шаблона необязательно задавать предпочитаемые настройки для всех визуальных компонентов. Компоненты, для которых не установлены предпочитаемые настройки, будут сконфигурированы значениями по умолчанию.

Классы и интерфейсы для работы с нативной рекламой

Внимание:

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

Классы

- BannerAppearance
- BannerAppearance.Builder
- ButtonAppearance
- ButtonAppearance.Builder
- HorizontalOffset
- ImageAppearance
- ImageAppearance.Builder
- NativeAdAssets
- NativeAdImage
- NativeAdLoader
- NativeAdRequestConfiguration
- NativeAdRequestConfiguration.Builder
- NativeAdMedia
- NativeAdViewBinder
- NativeAdViewBinder.Builder
- NativeBannerView
- NativeTemplateAppearance
- NativeTemplateAppearance.Builder
- MediaView
- RatingAppearance
- RatingAppearance.Builder
- SizeConstraint
- TextAppearance
- TextAppearance.Builder

Интерфейсы

- NativeAdEventListener
- NativeAdImageLoadingListener
- Rating

InStream реклама

Внимание:

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

InStream — формат рекламы, который позволяет монетизировать приложение через показ рекламы во время воспроизведения видеоконтента.

InStream объявление состоит из сценария с несколькими видеоблоками. Тип видеоблока в сценарии InStream определяет, как должен проигрываться рекламный ролик относительно основного видеоконтента.

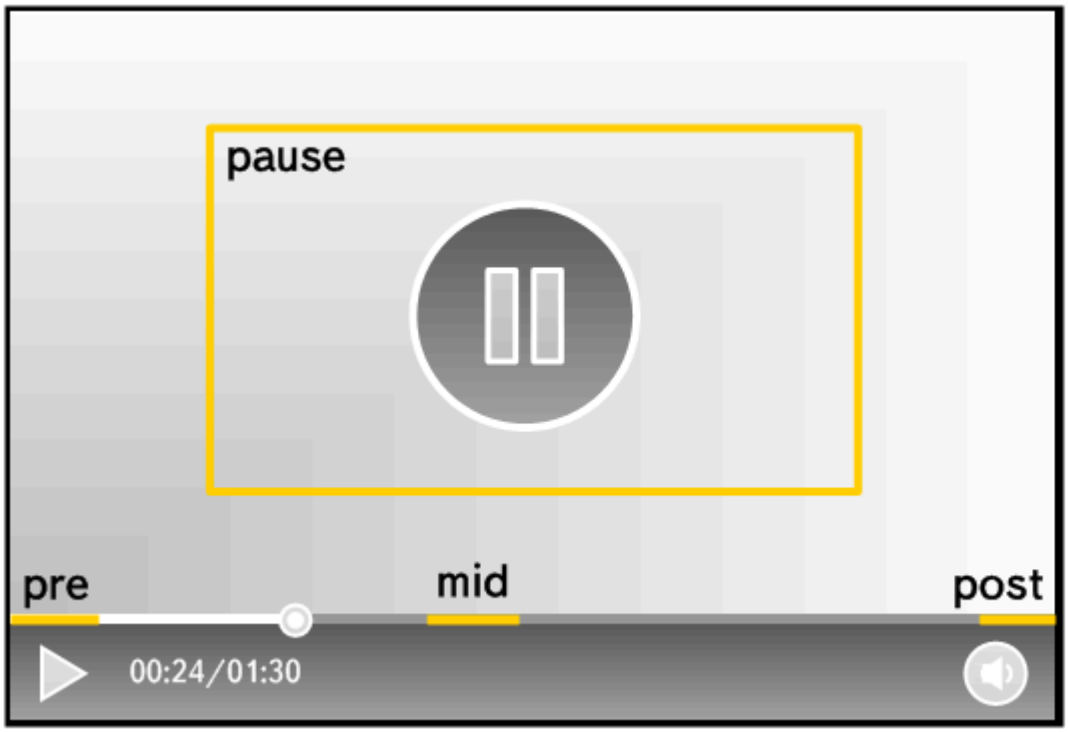
Об InStream

Чтобы настроить сценарий InStream, создайте видеоресурс в Партнерском интерфейсе. После создания видеоресурсу будет присвоен уникальный идентификатор (`Page ID`). Этот идентификатор необходимо использовать в Mobile Ads SDK.

Для увеличения рекламного дохода, есть возможность настроить показ нескольких рекламных объявлений в рамках одной рекламной вставки, так называемый AdPod. Настроить AdPod для рекламной вставки можно в Партнерском Интерфейсе.

Типы видеоблоков, которые поддерживает Mobile Ads SDK:

- Pre-Roll — рекламный ролик проигрывается перед основным контентом;
- Mid-Roll — рекламный ролик проигрывается по времени внутри основного контента;
- Post-Roll — рекламный ролик проигрывается после основного контента;
- Pause-Roll — рекламный ролик проигрывается при нажатии кнопки паузы;
- In-Roll — рекламный ролик проигрывается в любом месте видео при достижении определенной отметки.



1. In-roll

API для работы с InStream

Для работы с InStream рекламой есть несколько API:

ExoPlayer AdsLoader API

API для базовой интеграции InStream рекламы в плеер, использующий ExoPlayer. Позволяет быстро интегрировать показ InStream рекламы. API поддерживает рекламные вставки с типом Pre-Roll, Mid-Roll, Post-Roll.

Ограничение:

1. Необходима определенная версия ExoPlayer.
2. Отсутствует поддержка In-Roll и Pause-Roll.

InStream API

API для расширенной интеграции InStream рекламы. Позволяет поддержать проигрывание всех типов рекламных вставок, а также использовать свою реализацию плеера. InStream реклама состоит из рекламных вставок, которые проигрываются автоматически и вручную.

Для автоматического проигрывания вставок с типом Pre-Roll, Mid-Roll, Post-Roll используется InstreamAdBinder API. Для ручного запуска рекламных вставок с типом In-Roll и Pause-Roll используются In-Roll API и Pause-Roll API соответственно.

Примечание:

Доступно одновременное использование InstreamAdBinder API, In-Roll API и Pause-Roll API при соблюдении определенных условий:

1. Используйте разные инстансы рекламного плеера.
2. Не запускайте Pause-Roll и In-Roll API для воспроизведения, если через InStreamAdBinder API было приостановлено основное видео.

Базовая интеграция (ExoPlayer AdsLoader API)

Внимание:

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

YandexAdsLoader — упрощенный API для интеграции InStream рекламы через ExoPlayer. YandexAdsLoader поддерживает воспроизведение рекламных вставок с типом Pre-Roll, Mid-Roll, Post-Roll.

Данный тип интеграции подходит для приложений, которым не нужны расширенные возможности InStream API.

Подключение с использованием Gradle

В файл build.gradle добавьте следующие зависимости на уровне приложения:

```
dependencies {
    ...
    implementation 'com.yandex.android:mobileads:5.10.0'
    implementation 'com.google.android.exoplayer:exoplayer-core:2.18.1'
}
```

Показ рекламных объявлений

1. Добавьте в layout приложения com.google.android.exoplayer2.ui.PlayerView

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <com.google.android.exoplayer2.ui.PlayerView
        android:id="@+id/player_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center" />

</FrameLayout>
```

2. Создайте конфигурацию запроса `instreamAdRequestConfiguration` с помощью класса InstreamAdRequestConfiguration.Builder. В качестве параметров запроса передайте идентификатор страницы (`Page ID`) из Партнерского интерфейса.
3. Создайте экземпляр класса YandexAdsLoader для показа и загрузки InStream рекламы.
4. Создайте экземпляр класса `DefaultMediaSourceFactory`, установите в него созданный экземпляр YandexAdsLoader и `PlayerView`.

```
val userAgent = Util.getUserAgent(this, getString(R.string.app_name))
val dataSourceFactory = DefaultDataSourceFactory(this, userAgent)
val mediaSourceFactory = DefaultMediaSourceFactory(dataSourceFactory)
    .setAdsLoaderProvider { yandexAdsLoader }
    .setAdViewProvider(playerView)
```

5. Создайте экземпляр `ExoPlayer`, установите в него созданные `mediaSourceFactory`.
6. Установите созданный экземпляр `ExoPlayer` в `PlayerView` и YandexAdsLoader.

```
val player = SimpleExoPlayer.Builder(this)
    .setMediaSourceFactory(mediaSourceFactory).build()
playerView.player = player
yandexAdsLoader.setPlayer(player)
```

7. Создайте экземпляр `MediaItem`, установите в него ссылку на видео для воспроизведения и `YandexAdsLoader.AD_TAG_URI`.

```
val contentVideoUrl = getString(R.string.content_url_for_instream_ad)
val mediaItem = MediaItem.Builder()
    .setUri(contentVideoUrl)
    .setAdTagUri(YandexAdsLoader.AD_TAG_URI).build()
```

8. Запустите проигрывание видео с созданным экземпляром `MediaItem`.

```
player.apply {
    setMediaItem(mediaItem)
    prepare()
    playWhenReady = true
}
```

Расширенная интеграция (InStream API)

Внимание:

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

InStream API — расширенный API для настройки, управления загрузкой и воспроизведением InStream рекламы. Позволяет поддержать проигрывание всех типов рекламных вставок, а также использовать свою реализацию плеера. InStream реклама состоит из рекламных вставок, которые проигрываются автоматически и вручную.

Для автоматического проигрывания вставок с типом Pre-Roll, Mid-Roll, Post-Roll используется InstreamAdBinder API. Для ручного запуска рекламных вставок с типом In-Roll и Pause-Roll используются In-Roll API и Pause-Roll API соответственно.

Примечание:

Доступно одновременное использование InstreamAdBinder API, In-Roll API и Pause-Roll API при соблюдении определенных условий:

1. Используйте разные инстансы рекламного плеера.
2. Не запускайте Pause-Roll и In-Roll API для воспроизведения, если через InStreamAdBinder API было приостановлено основное видео.

Принципы работы

Загруженный объект InStream рекламы содержит в себе расписание показа рекламных вставок. Каждая рекламная вставка описывается объектом InstreamAdBreak. Рекламная вставка может иметь один из следующих типов: Pre / Mid / Post / In / Pause-Roll. Показать Pre/MidPost-Roll вставки можно через InstreamAdBinder API. Показать In / Pause-Roll вставки можно через In-Roll API и Pause-Roll API соответственно.

Для взаимодействия с основным видеоконтентом используется интерфейс VideoPlayer, а для воспроизведения рекламного видео внутри рекламной вставки — интерфейс InstreamAdPlayer.

Показ Pre/Mid/Post-Roll

InstreamAdBinder отслеживает ход воспроизведения основного видео и показывает рекламные вставки на основе настроек видеоресурса в Партнерском Интерфейсе.

InstreamAdBinder не управляет непосредственно отрисовкой рекламного видео в PlayerView. Воспроизводить рекламное видео необходимо со стороны приложения основываясь на сигналах от интерфейсов плееров, переданных в InstreamAdBinder. InstreamAdBinder сообщает о начале проигрывания рекламной вставки через вызов VideoPlayer.pauseVideo() и об окончании проигрывания рекламной вставки через вызов VideoPlayer.resumeVideo().

В момент вызова VideoPlayer.pauseVideo() со стороны приложения необходимо скрыть контролы управления основным видео, приостановить основное видео и начать воспроизведение рекламного видео. Со стороны рекламного SDK после вызова метода будут показаны рекламные контролы внутри контейнера InstreamAdView и будет вызван метод InstreamAdPlayer.playAd() для старта воспроизведения рекламного видео.

В момент вызова VideoPlayer.resumeVideo() со стороны приложения необходимо вернуть контролы управления основным видео и продолжить воспроизведение основного видео. Со стороны рекламного SDK до вызова метода будут убраны рекламные контролы внутри контейнера InstreamAdView.

Показ In / Pause-Roll

In / Pause-Roll API не управляет непосредственно отрисовкой рекламного видео в PlayerView. Воспроизводить рекламное видео необходимо со стороны приложения основываясь на сигналах от интерфейсов плееров, переданных в In / Pause-Roll. In / Pause-Roll сообщает о начале проигрывания рекламной вставки через вызов InstreamAdBreakEventListener.onInstreamAdBreakStarted() и об окончании проигрывания рекламной вставки через вызов InstreamAdBreakEventListener.onInstreamAdBreakCompleted() или InstreamAdBreakEventListener.onInstreamAdBreakError().

В момент вызова InstreamAdBreakEventListener.onInstreamAdBreakStarted() со стороны приложения необходимо скрыть контролы управления основным видео, приостановить основное видео. Со стороны рекламного SDK после вызова метода будут показаны рекламные контролы внутри контейнера InstreamAdView и будет вызван метод InstreamAdPlayer.playAd() для старта воспроизведения рекламного видео.

В момент вызова InstreamAdBreakEventListener.onInstreamAdBreakCompleted() или InstreamAdBreakEventListener.onInstreamAdBreakError() со стороны приложения необходимо вернуть контролы управления основным видео и продолжить воспроизведение основного видео. Со стороны рекламного SDK до вызова методов будут убраны рекламные контролы из контейнера InstreamAdView.

Загрузка рекламных объявлений

1. Создайте экземпляр класса InstreamAdLoader для получения InStream рекламы.
2. Настройте получение уведомлений (успешная загрузка рекламы или ошибка при загрузке рекламы): создайте экземпляр InstreamAdLoadListener и установите его в качестве слушателя событий загрузчика рекламных объявлений.
3. Создайте конфигурацию запроса `instreamAdRequestConfiguration` с помощью класса InstreamAdRequestConfiguration.Builder. В качестве параметров запроса передайте идентификатор страницы (`Page ID`) из Партнерского интерфейса.
4. Загрузите рекламу с помощью метода InstreamAdLoader.loadInstreamAd, передайте в него `Context` и `instreamAdRequestConfiguration`.

Пример кода:

Для тестирования корректности интеграции используйте демонстрационный Page ID: demo-instream-vmap-yandex.

```
final InstreamAdLoader instreamAdLoader = new InstreamAdLoader(context);
instreamAdLoader.setInstreamAdLoadListener(new InstreamAdLoadListener() {
        @override
        public void onInstreamAdLoaded(@NonNull final InstreamAd instreamAd) {
        ...
        }

        @override
        public void onInstreamAdFailedToLoad(@NonNull final String reason) {
        ...
        }
    });

final InstreamAdRequestConfiguration instreamAdRequestConfiguration =
    new InstreamAdRequestConfiguration.Builder(PAGE_ID).build();
instreamAdLoader.loadInstreamAd(this, instreamAdRequestConfiguration);
```

Показ рекламных объявлений

Показ Pre / Mid / Post-Roll

1. Реализуйте интерфейсы InstreamAdPlayer и VideoPlayer.

   В разделах справочника Package com.yandex.mobile.ads.instream.player.ad и Package com.yandex.mobile.ads.instream.player.content приведена подробная информация по работе методов и их реализации. Дополнительно рекомендуется ориентироваться на тестовый пример реализации.

   Внимание:

   Для упрощения реализации, рекомендуется использовать разные инстансы плееров для воспроизведения рекламы и контентного видео.

2. Добавьте в layout приложения InstreamAdView. InstreamAdView должна содержать в себе PlayerView, в которой будут проигрываться рекламные ролики.

   Пример кода:

   Ограничение:

   Размер контейнера должен быть не меньше 300dp x 160dp.

```
<com.yandex.mobile.ads.instream.player.ad.InstreamAdView
    android:id="@+id/instream_ad_view"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

        <PlayerView
            android:id="@+id/player_view"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>

</com.yandex.mobile.ads.instream.player.ad.InstreamAdView>
```

3. Создайте объект InstreamAdBinder: передайте в конструктор `Context`, объект загруженной рекламы InstreamAd и реализации плееров InstreamAdPlayer, VideoPlayer.

   Настройте получение уведомлений о ходе воспроизведения рекламы (готовность к проигрыванию видеорекламы, завершение воспроизведения или ошибка в процессе воспроизведения): создайте экземпляр InstreamAdListener и установите его в качестве слушателя событий объекта InstreamAdBinder.

```
mInstreamAdBinder = new InstreamAdBinder(context, mInstreamAd, mYandexAdPlayer, mContentVideoPlayer);
mInstreamAdBinder.setInstreamAdListener(...);
```

4. Для ускорения старта рекламной вставки с типом Pre-Roll, заранее предзагрузите ее через вызов метода InstreamAdBinder.prepareAd().

```
private void preparePrerollAd(@NonNull final InstreamAdBinder instreamAdBinder) {
    instreamAdBinder.setInstreamAdListener(new InstreamAdListener() {
        ...
        public void onInstreamAdPrepared() {
            addInstreamAdBinderToPreloadedAdQueue(instreamAdBinder);
        }
        ...
    });
    instreamAdBinder.prepareAd();
}
```

5. Вызовите метод InstreamAdBinder.bind(instreamAdView) у созданного объекта InstreamAdBinder. В качестве параметра передайте InstreamAdView. После этого InStream SDK начнет автоматически отслеживать прогресс воспроизведения основного видео и возьмет на себя управление проигрыванием рекламных роликов.

```
mInstreamAdBinder.bind(mInstreamAdView);
```

6. При реализации проигрывания InStream рекламы в списке, необходимо использовать метод InStreamBinder.unbind() при инвалидации ячейки с рекламой в списке. При реализации переиспользуемого пула плееров для скролла, необходимо вызывать InstreamAdbinder.invalidateAdPlayer() при переиспользовании плеера рекламы привязанного к InstreamAdBinder, а при переиспользовании плеера основного контента InstreamAdBinder.invalidateVideoPlayer().
7. При остановке использования InStreamAdBinder необходимо очищать состояние.

```
public void onDestroy() {
    instreamAdBinder.unbind();
    instreamAdBinder.invalidateVideoPlayer();
    instreamAdBinder.invalidateAdPlayer();
    instreamAdBinder.setInstreamAdListener(null);
    instreamAdBinder.setVideoAdPlaybackListener(null);

    super.onDestroy();
}
```

Показ In / Pause-Roll

Примечание:

Показ рекламной вставки Pause-Roll настраивается по аналогии с In-Roll. Для этого замените In-Roll классы / методы на Pause-Roll.

1. Реализуйте интерфейс InstreamAdPlayer.

   В разделе справочника Package com.yandex.mobile.ads.instream.player.ad приведена подробная информация по работе методов и их реализации. Дополнительно рекомендуется ориентироваться на тестовый пример реализации.

   Внимание:

   Для упрощения реализации, рекомендуется использовать разные инстансы плееров для воспроизведения рекламы и контентного видео.
2. Добавьте в layout приложения InstreamAdView. InstreamAdView должна содержать в себе PlayerView, в которой будут проигрываться рекламные ролики.

   Пример кода:

   Ограничение:

   Размер контейнера должен быть не меньше 300dp x 160dp.

   ```
   <com.yandex.mobile.ads.instream.player.ad.InstreamAdView
       android:id="@+id/instream_ad_view"
       android:layout_width="match_parent"
       android:layout_height="wrap_content">

           <PlayerView
               android:id="@+id/player_view"
               android:layout_width="match_parent"
               android:layout_height="wrap_content"/>

   </com.yandex.mobile.ads.instream.player.ad.InstreamAdView>
   ```
3. Загрузите через InstreamAdLoader объект InstreamAd с помощью идентификатора страницы (`Page ID`) из Партнерского интерфейса.
4. InstreamAd содержит в себе набор рекламных вставок разных типов. Чтобы получить рекламные вставки In-Roll, используйте InrollQueueProvider. Очередь InrollQueueProvider позволяет получать объекты In-Roll в необходимом для показа порядке.

   ```
   public void onInstreamAdLoaded(@NonNull final InstreamAd instreamAd) {
       final InrollQueueProvider inrollQueueProvider = new InrollQueueProvider(context, instreamAd);
       mInstreamAdBreakQueue = inrollQueueProvider.getQueue();
   }
   ```
5. Для запуска полученного In-Roll, необходимо его подготовить. Без подготовки In-Roll не запустится.

   Чтобы подготовить In-Roll, вызовите Inroll.prepare(instreamAdPlayer) интерфейса и передайте в него инстанс созданной реализации InstreamAdPlayer. Для отслеживания состояния рекламной вставки установите слушатель InstreamAdBreakEventListener

   ```
   public void prepare(instreamAdPlayer) {
      currentInroll = mInstreamAdBreakQueue.poll();
      currentInroll.setListener(new InrollListener());
      currentInroll.prepare(instreamAdPlayer);
   }
   ```
6. По окончанию подготовки In-Roll будет вызван InstreamAdBreakEventListener.onInstreamAdBreakPrepared(). Подготовленный In-Roll готов к показу.

   Совет:

   Показывайте объявления в порядке их получения из InstreamAdBreakQueue. Показ полученных In-Roll в другом порядке может снизить монетизацию вашего приложения.
7. Чтобы показать подготовленный In-Roll, вызовите Inroll.play() и передайте в качестве параметра InstreamAdView.

   ```
   public void onInstreamAdBreakPrepared() {
       if (currentInroll != null) {
           currentInroll.play(instreamAdView);
       }
   }
   ```
8. После начала проигрывания рекламной вставки, будет вызван метод InstreamAdBreakEventListener.onInstreamAdBreakStarted(). По вызову этого метода необходимо поставить на паузу проигрывание основного видео и скрыть контролы управления основного видео.

   ```
   public void onInstreamAdBreakPrepared() {
       if (contentVideoPlayer != null) {
           contentVideoPlayer.pauseVideo();
       }
   }
   ```

9. После проигрывания рекламной вставки необходимо продолжить воспроизведение основного видео. Проигрывание рекламной вставки может завершиться успешно или с ошибкой, эти две ситуации необходимо обрабатывать.

```
@Override
public void onInstreamAdBreakCompleted() {
    handleAdBreakCompleted();
}

@Override
public void onInstreamAdBreakError(@NonNull final String reason) {
    handleAdBreakCompleted();
}

private void handleAdBreakCompleted() {
    currentInroll = null;
    if (contentVideoPlayer != null) {
        contentVideoPlayer.resumeVideo();
    }
}
```

10. После завершения проигрывания текущего InRoll проверьте очередь воспроизведения на наличие следующего In-Roll в InstreamAdBreakQueue.

```
public void prepareNextAd() {
   currentInroll = mInstreamAdBreakQueue.poll();
   if (currentInroll != null) {
       prepareInroll(currentInroll);
   }
}
```

11. При остановке использования In-Roll необходимо очищать его состояние.

```
public void onDestroy() {
    if (currentInroll != null) {
        currentInroll.invalidate();
        currentInroll.setListener(null);
    }
    super.onDestroy();
}
```

Классы и интерфейсы для работы с InStream рекламой

Внимание:

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

- Package com.yandex.mobile.ads.instream
- Package com.yandex.mobile.ads.instream.exoplayer
- Package com.yandex.mobile.ads.instream.inroll
- Package com.yandex.mobile.ads.instream.pauseroll
- Package com.yandex.mobile.ads.instream.player.ad
- Package com.yandex.mobile.ads.instream.player.ad.error
- Package com.yandex.mobile.ads.instream.player.content

GDPR

Внимание:

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

Общие сведения

Весной 2018 года в силу вступил общий регламент по защите данных (General Data Protection Regulation, сокращенно GDPR). Регламент регулирует сбор и обработку информации о физических лицах — гражданах Европейской экономической зоны и Швейцарии. Он призван усилить защиту

конфиденциальных данных и сделать прозрачными все элементы сбора, хранения и обработки информации в интернете.

GDPR имеет экстерриториальное действие и применяется ко всем компаниям, которые обрабатывают персональные данные граждан Европейской экономической зоны и Швейцарии, независимо от местонахождения такой компании.

Начиная с версии 2.80, Yandex Mobile Ads SDK позволит ограничить сбор данных пользователей, расположенных в Европейской экономической зоне и Швейцарии, при отсутствии их согласия на это.

Краткое руководство

Согласие пользователя на обработку персональных данных необходимо передавать в SDK при каждом запуске приложения.

1. Подключите Mobile Ads SDK по инструкции.
2. Отобразите для пользователя диалог, предлагающий принять пользовательское соглашение на обработку персональных данных (подробнее в примере).

   Внимание:

   Данный код является демонстрацией, а не руководством к действию.

```
...
public class GdprDialogFragment extends DialogFragment {
  ...
  // Код демонстрирует создание диалога.
  public Dialog onCreateDialog(Bundle savedInstanceState) {
        final Context context = getContext();

        AlertDialog.Builder builder = new AlertDialog.Builder(context);
        builder.setTitle(R.string.gdpr_title)
                .setMessage(R.string.gdpr_message)
                .setPositiveButton(R.string.accept, new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        onButtonClicked(context, true);
                    }
                })
                .setNeutralButton(R.string.about, new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(final DialogInterface dialog, final int which) {
                        openPrivacyPolicy();
                    }
                })
                .setNegativeButton(R.string.decline, new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        onButtonClicked(context, false);
                    }
                });
        return builder.create();
    }

    private void openPrivacyPolicy() {
        final String url = getString(R.string.privacy_policy_url);
        final Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
        startActivity(intent);
    }

    private void onButtonClicked(final Context context,
                                 final boolean userConsent) {
        final SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(context);
        preferences.edit()
                .putBoolean(SettingsFragment.USER_CONSENT_KEY, userConsent)
                .putBoolean(SettingsFragment.DIALOG_SHOWN_KEY, true)
                .apply();

        mNoticeDialogListener.onDialogClick();
    }
}
```

3. Передайте полученное значение в Mobile Ads SDK с помощью метода setUserConsent. Данные пользователей, находящихся в GDPR-регионе, будут обрабатываться только при наличии согласия пользователя на обработку данных.

COPPA

Внимание:

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

Общие сведения

21 апреля 2000 года в силу вступил Закон о защите конфиденциальности детей в Интернете (Children's Online Privacy Protection Act, сокращенно COPPA). Закон применяется к сбору персональной информации от детей младше 13 лет лицами или организациями под юрисдикцией США. Согласно закону, администрация приложения должна включить в политику конфиденциальности способы получения согласия родителей или опекунов и ответственности администрации за защиту конфиденциальности и безопасности детей в Интернете, включая ограничения в маркетинге.

Данные о возрасте пользователя необходимо передавать в SDK при каждом запуске приложения.

Начиная с версии 5.4.0, Yandex Mobile Ads SDK позволит ограничить сбор данных от детей младше 13 лет.

Краткое руководство

Данные о возрасте пользователя необходимо передавать в SDK при каждом запуске приложения.

1. Подключите Mobile Ads SDK по инструкции.
2. Отобразите для пользователя диалог, предлагающий принять пользовательское соглашение на обработку персональных данных (подробнее в примере).
3. Передайте полученное значение в Mobile Ads SDK с помощью метода setAgeRestrictedUser. По умолчанию считается, что пользователь не является ребенком младше 13 лет.

Учет контекстных данных

Внимание:

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

Примечание: Чтобы Mobile Ads SDK могла учитывать контекст приложения, подключите AppMetrica SDK версии 3.14.3 и выше.

Учет контекстных работает с 20 августа 2020 г.

Чтобы повысить эффективность монетизации, Mobile Ads SDK автоматически учитывает контекст приложения: тексты интерфейса, их тематику и взаимодействие пользователя с контентом. За счет этого подбирается более релевантная реклама.

При этом:

- вы можете ограничить учет контекстных данных, например, в местах, где пользователи указывают конфиденциальную информацию — на экранах оплаты или в личной переписке;
- вы можете полностью отключить учет контекстных данных;
- SDK учитывает только обезличенные данные и соответствует стандарту ISO.

Требования для работы с функциональностью

1. Функциональность доступна только для платформы Android.

2. Минимальная поддерживаемая версия AppMetrica SDK 3.14.3 и выше.
3. Минимальная поддерживаемая версия Mobile Ads SDK Android 2.160 и выше.

Как включить учет контекстных данных

Чтобы включить автоматический учет контекстных данных приложения, инициализируйте библиотеку AppMetrica SDK версии 3.14.3 и выше.

Как отключить учет контекстных данных

Отключить автоматический учет можно для разных сущностей: Application, Activity, View:

Application

Чтобы отключить автоматический учет контекстных данных для всего приложения, в файле `AndroidManifest.xml` на уровне application добавьте код:

```
<meta-data
    android:name="@string/yandex_ads_context"
    android:value="@string/yandex_ads_context_do_not_parse"/>
```

Пример кода:

```
<application
    android:name="com.yandex.appmetrica.autotests.agent.AgentApplication"
    ...>
    <meta-data
        android:name="@string/yandex_ads_context"
        android:value="@string/yandex_ads_context_do_not_parse"/>
</application>
```

Activity

Чтобы отключить автоматический учет контекстных данных для конкретной activity, в файле `AndroidManifest.xml` на уровне activity добавьте код:

```
<meta-data
    android:name="@string/yandex_ads_context"
    android:value="@string/yandex_ads_context_do_not_parse"/>
```

Пример кода:

```
<activity
    android:name=".NoContextActivity"
    ...>
    <meta-data
        android:name="@string/yandex_ads_context"
        android:value="@string/yandex_ads_context_do_not_parse"/>
</activity>
```

View

Отключить автоматический учет контекстных данных для конкретной view можно одним из способов:

В ресурсах Android проекта

```
<TextView ...>
    <tag android:id="@id/yandex_ads_context"
         android:value="@string/yandex_ads_context_do_not_parse"/>
</TextView>
```

Программно

```
view.setTag(R.id.yandex_ads_context, getString(R.string.yandex_ads_context_do_not_parse))
```

TCF v2.0 Consent

Внимание:

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

Если вы реализовали CMP (Consent Management Platform), которая соответствует IAB TCF v2.0 (Transparency & Consent Framework) для пользовательского сценария выдачи разрешения на передачу данных (user consent flow), Yandex Mobile Ads SDK поддерживает отправку TCF v2.0 значений.

Yandex Mobile Ads SDK считывает TCF v2.0 значения разрешения (consent strings) из `SharedPreferences` по следующим ключам:

IABTCF_TCString	String	Полная закодированная TC строка
IABTCF_gdprApplies	Number	• 1 — GDPR применим в текущем контексте; • 0 — GDPR не применим в текущем контексте; • Unset — не определено (значение по умолчанию перед инициализацией).
IABTCF_CmpSdkID	Number	Целое число без знака, идентификатор (ID) используемого CMP SDK
IABTCF_PurposeConsents	Binary String	«0» или «1» на позиции n, где отсчет n начинается с 0 — отображает статус разрешения (consent) для цели (purpose) с идентификатором (ID) n +1. «0» соответствует `false`, «1» — `true`. Например, «1» на позиции 0 — это выданное разрешение для цели с идентификатором 1.
IABTCF_VendorConsents	Binary String	«0» или «1» на позиции n, где отсчет n начинается с 0 — отображает статус разрешения (consent) для вендора с идентификатором (ID) n +1. «0» соответствует `false`, «1» — `true`. Например, «1» на позиции 0 — это выданное разрешение для вендора с идентификатором 1.

Подробное описание можно найти в документации IAB.

Передача значений в `SharedPreferences`

Передать значения по ключам необходимо перед походами за рекламой. Сделать это можно одним из способов:

Вручную

Самостоятельно положите значения по ключам из таблицы выше в `SharedPreferences`. Пример:

Kotlin

```
val sharedPref = activity?.getPreferences(Context.MODE_PRIVATE) ?: return
with (sharedPref.edit()) {
    putInt("IABTCF_gdprApplies", 1)
    apply()
}
```

Java

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
```

```
SharedPreferences.Editor editor = sharedPref.edit();
editor.putInt("IABTCF_gdprApplies", newHighScore);
editor.apply();
```

Примечание:

Подробности в документации. Параметры зависят от страны, настроек пользователя и устройства.

Через SDK для управления CMP

Используйте готовые SDK для управления CMP. Такие SDK автоматически передадут необходимые значения в `SharedPreferences` и Yandex Mobile SDK сможет их использовать.

Таргетирование рекламы

Внимание:

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

С помощью данных классов можно настраивать работу SDK и задавать настройки для таргетирования рекламы.

- MobileAds
- AdRequest
- AdRequest.Builder
- Gender

Руководство по миграции на версию 5

Внимание:

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

Дополнения

Пакет com.yandex.mobile.ads.banner

Интерфейс BannerAdEventListener

- Новый метод `void onImpression(ImpressionData impressionData)`, который вызывается, когда был засчитан рекламный показ.
- Новый метод `void onAdClicked()`, который вызывается, когда пользователь кликнул по баннеру.

Пакет com.yandex.mobile.ads.instream.exoplayer

Добавлен новый пакет для интеграции InStream рекламы через ExoPlayer.

Пакет com.yandex.mobile.ads.instream.player.ad

Интерфейс InstreamAdPlayerListener

- Новый метод `void onAdBufferingFinished(VideoAd videoAd)`. Вызывается, когда InstreamAdPlayer завершил буферизацию рекламного ролика.
- Новый метод `void onAdBufferingStarted(VideoAd videoAd)`. Вызывается, когда InstreamAdPlayer начал буферизацию рекламного ролика.

Пакет com.yandex.mobile.ads.instream.player.ar.error

Добавлен новый пакет для работы с ошибками в InstreamAdPlayer.

Пакет com.yandex.mobile.ads.interstitial

Интерфейс InterstitialAdEventListener

- Новый метод `void onImpression(ImpressionData impressionData)`, который вызывается, когда был засчитан рекламный показ.
- Новый метод `void onAdClicked()`, который вызывается, когда пользователь кликнул по рекламному объявлению.

Пакет com.yandex.mobile.ads.nativeads

Интерфейс NativeAdEventListener

- Новый метод `void onImpression(ImpressionData impressionData)`, который вызывается, когда был засчитан рекламный показ.
- Новый метод `void onAdClicked()`, который вызывается, когда пользователь кликнул по рекламному объявлению.

Пакет com.yandex.mobile.ads.rewarded

Интерфейс RewardedAdEventListener

- Новый метод `void onImpression(ImpressionData impressionData)`, который вызывается, когда был засчитан рекламный показ.
- Новый метод `void onAdClicked()`, который вызывается, когда пользователь кликнул по рекламному объявлению.

Изменения

Пакет com.yandex.mobile.ads.banner

Класс BannerAdView

- Метод `void setBlockId` переименован в `void setAdUnitId`.

Пакет com.yandex.mobile.ads.interstitial

Класс InterstitialAd

- Метод `void setBlockId` переименован в `void setAdUnitId`.

Пакет com.yandex.mobile.ads.nativeads

Класс NativeAdRequestConfiguration.Builder

- Метод `public Builder(@NonNull java.lang.String blockId)` переименован в `public Builder(@NonNull java.lang.String adUnitId)`.

Пакет com.yandex.mobile.ads.rewarded

Класс RewardedlAd

- Метод `void setBlockId` переименован в `void setAdUnitId`.

Удаления

Пакет com.yandex.mobile.ads.banner

Класс AdSize

Удалены методы:

- `static AdSize flexibleSize()`
- `static AdSize flexibleSize(int width)`

Пакет com.yandex.mobile.ads.instream

- Интерфейс `InstreamAdSkipInfo` удален.

Пакет com.yandex.mobile.ads.instream.model

Пакет удален.

Пакет com.yandex.mobile.ads.nativeads

- Класс `SliderAdView` удален.

Рекомендации

Версия 4.X.Y	Версия 5
`final AdSize flexibleAdSize = AdSize.flexibleSize(width);`	Удалено, альтернативный вариант: `final AdSize flexibleAdSize = AdSize.flexibleSize(width, height);`
`final AdSize flexibleAdSize = AdSize.flexibleSize();`	Удалено, альтернативный вариант: `final AdSize flexibleAdSize = AdSize.flexibleSize(width, height);`

Версия 4.X.Y	Версия 5
`mBannerAdView.setBlockId(<BlockId>);`	Параметр `BlockId` переименован в `AdUnitId`: `mBannerAdView.setAdUnitId(<AdUnitId>);`
`mInterstitialAd.setBlockId(<BlockId>);`	Параметр `BlockId` переименован в `AdUnitId`: `mInterstitialAd.setAdUnitId(<AdUnitId>);`
`mRewardedAd.setBlockId(<AdUnitID>);`	Параметр `BlockId` переименован в `AdUnitId`: `mRewardedAd.setAdUnitId(<AdUnitId>);`
`final NativeAdRequestConfiguration nativeAdRequestConfiguration = new NativeAdRequestConfiguration.Builder(BlockId).build();`	Параметр `BlockId` переименован в `AdUnitId`: `final NativeAdRequestConfiguration nativeAdRequestConfiguration = new NativeAdRequestConfiguration.Builder(AdUnitId).build();`
`bindSliderAd(@NonNull final SliderAdView sliderAdView);`	Изменен способ установки View: `bindSliderAd(@NonNull final SliderAdViewBinder viewBinder);`
`void onError(@NonNull final VideoAd videoAd);`	`void onError(@NonNull final VideoAd videoAd, @NonNull final InstreamAdPlayerError error);`

Режим отладки интеграции

Внимание:

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

Режим отладки интеграции помогает заранее выявить ошибки при подключении рекламной SDK или понять, что интеграция прошла успешно.

В режиме отладки доступны:

1. Индикатор корректной интеграции нативной рекламы
2. Режим проверки интеграции актуальной версии рекламной SDK

Индикатор корректной интеграции нативной рекламы

С помощью индикатора можно определить, успешно ли прошла интеграция нативной рекламы или получить отладочную информацию, чтобы понять причину ошибки.

Чтобы включить отображение индикатора в отладочном режиме, вызовите метод enableDebugErrorIndicator со значением `true`:

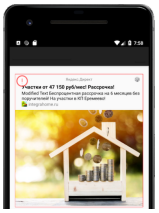
```
MobileAds.enableDebugErrorIndicator(true)
```

Успешная интеграция

Если интеграция прошла успешно, то в отладочном режиме, поверх рекламного объявления появится рамка светлого зеленого цвета.

Интеграция с ошибкой

Если при интеграции нативной рекламы допущена ошибка, то в отладочном режиме, 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.



Чтобы выключить отображение индикатора в отладочном режиме, вызовите метод enableDebugErrorIndicator со значением `false`:

```
MobileAds.enableDebugErrorIndicator(false)
```

Режим проверки интеграции актуальной версии рекламной SDK

В Yandex Mobile Ads SDK добавлена проверка интеграции актуальной версии SDK. Режим проверки интеграции актуальной версии рекламной SDK включает в себя:

- индикатор устаревшей версии библиотеки;
- lint проверка использования актуальной версии SDK.

Индикатор устаревшей версии библиотеки

Индикатор устаревшей версии рекламной SDK отображается в виде тоста во время инициализации библиотеки или загрузки объявления.

Более подробная информация о проблеме дублируется в логи приложения.

Пример логов:

```
********************************************************************************
* The integrated version of the Yandex Mobile Ads SDK is outdated.             *
* Please update com.yandex.android:mobileads to the latest version.            *
* Learn more about the latest version of the SDK here:                         *
* https://yandex.ru/dev/mobile-ads/doc/android/quick-start/android-ads-component.html *
* Changelog: https://yandex.ru/dev/mobile-ads/doc/intro/changelog-android.html *
********************************************************************************
```

Чтобы выключить отображение индикатора в отладочном режиме, вызовите метод enableDebugErrorIndicator со значением `false`:

```
MobileAds.enableDebugErrorIndicator(false)
```

Совет:

Не рекомендуется отключать валидацию индикатора устаревшей версии SDK. Старайтесь всегда использовать самую актуальную версию библиотеки, чтобы получать максимальный доход от рекламы.

Lint проверка использования актуальной версии SDK

Lint проверка использования актуальной версии SDK запускается во время сборки релизной версии приложения.

Lint проверка осуществляется за счет вызова gradle task `lintVitalRelease`. Таск падает, если используемая версия рекламной SDK более неактуальна.

Данная проверка не даст собраться приложению, если уже была выпущена более новая версия sdk.

Чтобы отключить проверку использования актуальной версии SDK, можно воспользоваться стандартным способом отключения lint проверок при помощи gradle-кода (подробнее в документации Android).

```
android {
    lintOptions {
        disable 'MobileAdsSdkOutdatedVersion'
    }
}
```