Mobile mediation

Integration

10.07.2024

Mobile mediation. Integration. Version 2.0

Document build date: 10.07.2024

This volume is a part of Yandex technical documentation.

Copyright Disclaimer

Contact information

Yandex LLC

https://www.yandex.com

Тел.: +7 495 739 7000

Email: pr@yandex-team.ru

16 L'va Tolstogo St., Moscow, Russia 119021

Contents

Resources

Note:

Any call of the Mobile Ads SDK should be made from the main thread.

Set up mediation in the Yandex Partner interface and YAN interface.

SKAdNetwork support

Note:

SKAdNetwork is supported for SDK version 4.1.2 and higher.

Mobile Ads SDK supports tracking of app installations using the SKAdNetwork framework. Installation tracking works for any device, even if no access to IDFA was granted.

To enable this functionality, add the Yandex Advertising Network ID to the app's `Info.plist` file.

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

For more information, see Configuring a Source App in the Apple documentation.

Note

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

Integrating the Mobile Ads SDK

Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

Note:

1. To load ads of any type, Android 4.1 or later is required.
2. Video ads are only selected for devices with Android 5.0 or later.

Several integration methods are supported:

Integrating the SDK with adapters

You can enable all available adapters automatically using the `YandexMobileAdsMediation` shared mediation library.

1. Add the YandexMobileAdsMediation dependency to the `build.gradle` file in your app's module:

```
dependencies {
    ...
    implementation 'com.yandex.android:mobileads-mediation:5.10.0.0'
}
```

For each adapter, the latest compatible version is selected automatically.

2. Update the dependency on Koltin Gradle Plugin to the `build.gradle` file in your app's module:

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

3. Add Java 8 support to the `build.gradle` root file in your project:

```
android {

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

4. Add the following code to the `build.gradle` file in your app's module:

```
// IronSource
maven {
       url 'https://android-sdk.is.com/'
}

// Pangle
maven {
       url 'https://artifact.bytedance.com/repository/pangle'
}

// Tapjoy
maven {
       url "https://sdk.tapjoy.com/"
}

// Mintegral
maven {
       url "https://dl-maven-android.mintegral.com/repository/mbridge_android_sdk_oversea"
}
```

5. Set up permission to use the ad ID.

   A new permission has been made available in the Yandex Mobile Ads SDK version 4.5.0 and higher: `com.google.android.gms.permission.AD_ID`. It's written in the library's `AndroidManifest.xml` file. Because of this, you don't have to specify it in the application's main manifest. The permission allows you to use an ad ID to select relevant ads from advertising networks.

   You can delete the permission if necessary. For example, if a policy does not allow the use of an ID for ad selection, such as the Families Policy.

   To prevent the permission from being added to the application's main manifest, add the following line to `AndroidManifest.xml`:

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

Integrating the SDK

1. If you don't need to enable all the available adapters automatically, add the YandexMobileAdsMediation dependency to the build.gradle file in your app's module:

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

   To enable relevant adapters, follow their instructions.

2. Add Java 8 support to the `build.gradle` file in your app's module:

```
android {

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

3. Set up permission to use the ad ID.

   A new permission has been made available in the Yandex Mobile Ads SDK version 4.5.0 and higher: `com.google.android.gms.permission.AD_ID`. It's written in the library's `AndroidManifest.xml` file.

Because of this, you don't have to specify it in the application's main manifest. The permission allows you to use an ad ID to select relevant ads from advertising networks.

You can delete the permission if necessary. For example, if a policy does not allow the use of an ID for ad selection, such as the Families Policy.

To prevent the permission from being added to the application's main manifest, add the following line to `AndroidManifest.xml`:

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

Initializing the Mobile Ads SDK

Before loading ads, initialize the library using the initialize() method. Initialization makes ads load faster.

Note:

It's needed each time the app starts. That's why we recommend that you add the initialization code to the `onCreate` method of the `Application` class.

Initialization example:

```
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");
            }
        });
    }
}
```

See the SDK usage examples.

Ad formats

Banner ads

Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

A *banner* is a configurable ad that covers part of the screen and reacts to clicks.

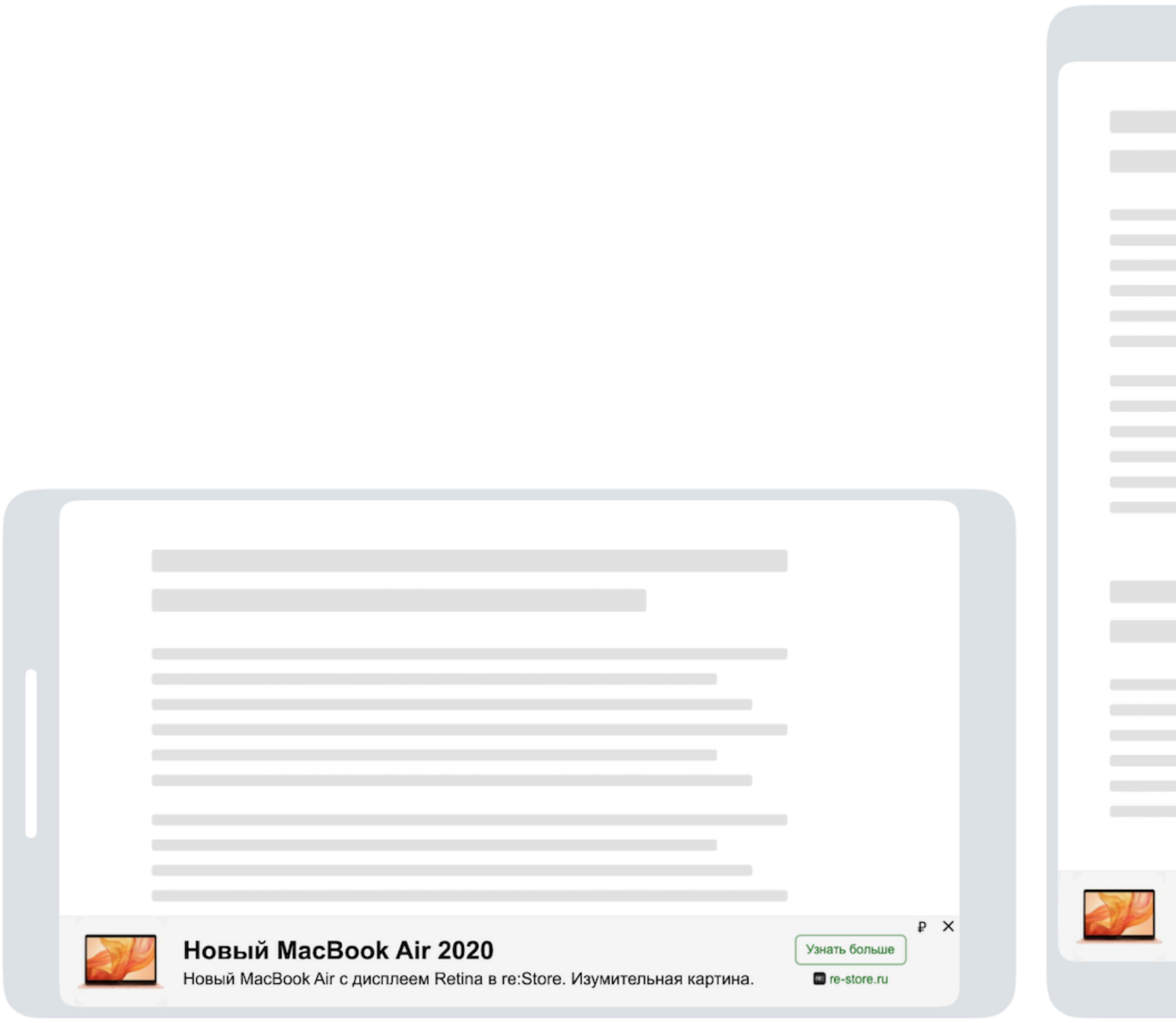
Banner types

Sticky banner

Features:

1. The specified banner width is used. The height is selected automatically.
2. The width of banners is set using the stickySize method.
3. The banner height shouldn't exceed 15% of the device height and should be at least 50 dp.

Examples of displaying banners:

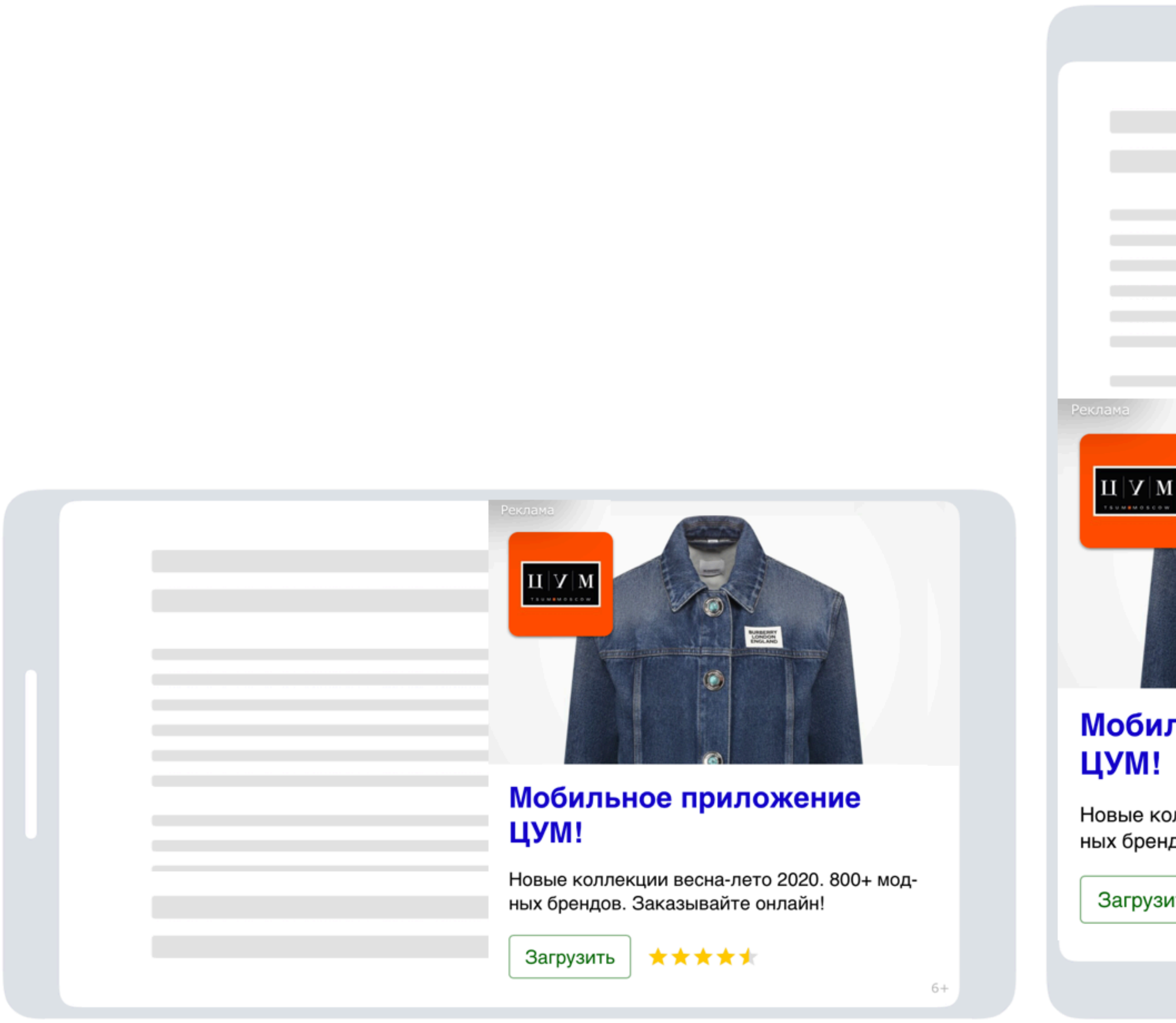

Новый MacBook Air 2020
Новый MacBook Air с дисплеем Retina в re:Store. Изумительная картина.
Узнать больше
re-store.ru

Flex banner

Features:

1. A banner fills up the entire unit using the set maximum sizes.
2. The width and height of a banner is set using the flexibleSize(int width, int height) method.

Examples of displaying banners:


Реклама
ЦУМ
Мобильное приложение ЦУМ!
Новые коллекции весна-лето 2020. 800+ модных брендов. Заказывайте онлайн!
Загрузить
6+

Enabling a banner

Creating Adview

1. Add an object of the `BannerAdView` class to the project using an XML file or programmatically.

```
// Creating an mBannerAdView instance using an XML file.
        mBannerAdView = (BannerAdView) findViewById(R.id.banner_view);

// Creating an mBannerAdView instance programmatically.
        mBannerAdView = new BannerAdView(this);
```

2. Set the `AdUnitId` using the setAdUnitId method.

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

`AdUnitId` is a unique identifier in R-M-XXXXXX-Y format, which is assigned in the Partner Interface.

3. Set the banner size using the setAdSize method.

   Sticky banner

   To set the width of a banner, call the stickySize(int width) method, where width is the maximum banner width.

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

   Flex banner

   To set the width and height of a banner, call the flexibleSize(int width, int height) method.

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

   Restriction: Banner size requirements when displaying video ads

   Minimum size of a banner that supports video playback is 300x160 or 160x300 dp (density-independent pixels).

4. After creating and configuring an instance of the `BannerAdView` class, you can set an AdEventListener on the ad object for tracking events (opening or closing the ad, exiting the app, and loading the ad successfully or unsuccessfully).

Loading ads

Note:

Any call of the Mobile Ads SDK should be made from the main thread.

After creating and configuring the `BannerAdView` class object, load the ads. To load an ad, use the loadAd method, which takes the AdRequest object as a parameter (or Builder, which optionally accepts ad targeting data).

Example of working with banner ads

The following code demonstrates creating and configuring the `AdView` object, registering a listener, and loading a banner:

```
...
<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) {
        ...
        // Creating an mAdView instance.
        mBannerAdView = (BannerAdView) findViewById(R.id.banner_ad_view);
        mBannerAdView.setAdUnitId(AdUnitId);
        mBannerAdView.setAdSize(AdSize.stickySize(AdSize.FULL_WIDTH));

        // Creating an ad targeting object.
        final AdRequest adRequest = new AdRequest.Builder().build();

        // Registering a listener for tracking events in the banner ad.
        mBannerAdView.setAdEventListener(new BannerAdEventListener() {
            @Override
            public void onAdLoaded() {
                ...
            }

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

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

            @Override
            public void onReturnedToApplication() {
                ...
```

```
            }
        });

        // Loading ads.
        mBannerAdView.loadAd(adRequest);
    }
}
```

If an ad is integrated this way, the banner appears after the app starts:

10:45
◀ BannerExample
Yandex Mobile Ads

To see how the banner ad will be displayed in the app, use a demo AdUnitId:

```
YANDEX_AD_UNIT_ID = "demo-banner-yandex"
ADCOLONY_AD_UNIT_ID = "demo-banner-adcolony"
ADMOB_AD_UNIT_ID = "demo-banner-admob"
APPLOVIN_AD_UNIT_ID = "demo-banner-applovin"
CHARTBOOST_AD_UNIT_ID = "demo-banner-chartboost"
MINTEGRAL_AD_UNIT_ID = "demo-banner-mintegral"
MYTARGET_AD_UNIT_ID = "demo-banner-mytarget"
STARTAPP_AD_UNIT_ID = "demo-banner-startapp"
VUNGLE_AD_UNIT_ID = "demo-banner-vungle"
```

See also

Classes and interfaces for working with banner ads

Related information

Ad example

Interstitial ads

Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

An *interstitial ad* is a configurable ad that covers the entire screen and responds to clicks.

To enable advertising:

- Create an InterstitialAd
- Load the ad
- Display the ad

Creating an InterstitialAd

1. Create an InterstitialAd class object. This object can only be created programmatically.

   ```
   mInterstitialAd = new InterstitialAd(this);
   ```
2. Set the AdUnitId using the setAdUnitId method.

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

   `AdUnitId` is a unique identifier in R-M-XXXXXX-Y format, which is assigned in the Partner Interface.
3. After creating and configuring an instance of the InterstitialAd class, you can set an InterstitialEventListener on the ad object for tracking events (opening or closing the ad, exiting the app, and loading the ad successfully or unsuccessfully).

Loading ads

Note:

Any call of the Mobile Ads SDK should be made from the main thread.

After creating and configuring an instance of the AdView class, load ads. To load an ad, use the loadAd method, which takes the AdRequest object as a parameter (or Builder, which optionally accepts ad targeting data).

Displaying ads

An interstitial ad is loaded in the background immediately after the loadAd call. To display an interstitial ad, you must call the show method.

We recommend checking whether the ad has actually loaded. To do this, call the isLoaded method.

Note:

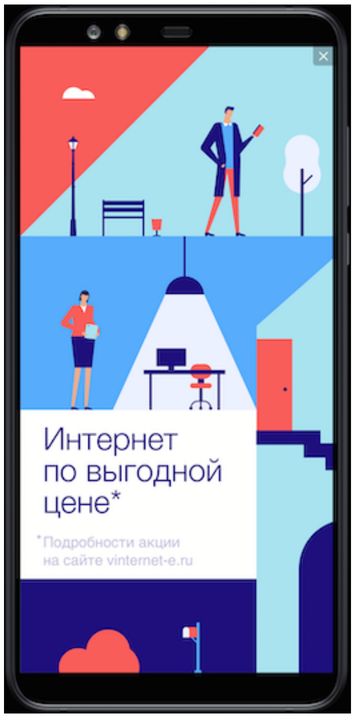
You don't need to check this if the show method is called after the onAdLoaded callback has been triggered.

Example of working with interstitial ads

The following code demonstrates creating and configuring the InterstitialAd object, registering a listener, and loading and displaying the interstitial ad:

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

        // Creating an ad targeting object.
        final AdRequest adRequest = new AdRequest.Builder().build();

        // Registering a listener to track events in the ad.
        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() {
                ...
            }
        });

        // Loading ads.
        mInterstitialAd.loadAd(adRequest);
    }
}
```

If an ad is integrated this way, the ad unit appears after the app starts:


Интернет по выгодной цене*
*Подробности акции на сайте vinternet-e.ru

To see how the ad will be displayed in the app, use a demo AdUnitId:

```
YANDEX_AD_UNIT_ID = "demo-interstitial-yandex"
ADCOLONY_AD_UNIT_ID = "demo-interstitial-adcolony"
ADMOB_AD_UNIT_ID = "demo-interstitial-admob"
APPLOVIN_AD_UNIT_ID = "demo-interstitial-applovin"
CHARTBOOST_AD_UNIT_ID = "demo-interstitial-chartboost"
IRONSOURCE_AD_UNIT_ID = "demo-interstitial-ironsource"
MINTEGRAL_AD_UNIT_ID = "demo-interstitial-mintegral"
MYTARGET_AD_UNIT_ID = "demo-interstitial-mytarget"
PANGLE_AD_UNIT_ID = "demo-interstitial-pangle"
STARTAPP_AD_UNIT_ID = "demo-interstitial-startapp"
TAPJOY_AD_UNIT_ID = "demo-interstitial-tapjoy"
UNITYADS_AD_UNIT_ID = "demo-interstitial-unityads"
VUNGLE_AD_UNIT_ID = "demo-interstitial-vungle"
```

See also

Classes and interfaces for working with interstitial ads

Related information

Ad example

Enabling rewarded ads

Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

A *rewarded ad* is a configurable full-screen ad. The user gets a reward for viewing the ad.

To enable advertising:

Create RewardedAd
Load the ad
Display the ad

Creating a rewarded ad

1. Add a RewardedAd class object.

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

2. Set the AdUnitId using the setAdUnitId method.

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

   `AdUnitId` is a unique identifier in R-M-XXXXXX-Y format, which is assigned in the Partner interface.
3. If you are assigning rewards on the application side (“client-side reward”), implement the onRewarded interface method. It is called when the impression is registered and the user can be rewarded for viewing the ad. Use this chance to give the reward to the app user.

After creating and configuring a RewardedAd class object, you need to install a listener of the RewardedAdEventListener interface for the ad object in order to track events (like opening or closing the ad, exiting the app, or loading the app successfully or unsuccessfully).

Loading ads

Note:

Any call of the Mobile Ads SDK should be made from the main thread.

After creating and configuring the RewardedAd class object, load the ads. To load an ad, use the loadAd method, which takes the AdRequest object as a parameter (or Builder, which optionally accepts ad targeting data).

Displaying ads

Rewarded ads are loaded in the background immediately after the loadAd method is called. To display rewarded ads, call the show method.

We recommend checking whether the ad has actually loaded. To do this, call the isLoaded method.

Note:

You don't need to check this if the show method is called after the onAdLoaded callback has been triggered.

Example of working with rewarded ads

The following code demonstrates how to create and configure a RewardedAd object, register the listener, and load and display the rewarded ad:

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

        // Creating an ad targeting object.
        final AdRequest adRequest = new AdRequest.Builder().build();

        // Registering a listener to track events in the ad.
        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() {
                ...
            }
        });

        // Loading ads.
        mRewardedAd.loadAd(adRequest);
    }
}
```

If advertising is integrated this way, the ad unit appears after the app starts.

To see how the ad will be displayed in the app, use a demo AdUnitId:

```
YANDEX_AD_UNIT_ID = "demo-rewarded-yandex"
ADCOLONY_AD_UNIT_ID = "demo-rewarded-adcolony"
ADMOB_AD_UNIT_ID = "demo-rewarded-admob"
APPLOVIN_AD_UNIT_ID = "demo-rewarded-applovin"
CHARTBOOST_AD_UNIT_ID = "demo-rewarded-chartboost"
IRONSOURCE_AD_UNIT_ID = "demo-rewarded-ironsource"
MINTEGRAL_AD_UNIT_ID = "demo-rewarded-mintegral"
MYTARGET_AD_UNIT_ID = "demo-rewarded-mytarget"
PANGLE_AD_UNIT_ID = "demo-rewarded-pangle"
STARTAPP_AD_UNIT_ID = "demo-rewarded-startapp"
TAPJOY_AD_UNIT_ID = "demo-rewarded-tapjoy"
UNITYADS_AD_UNIT_ID = "demo-rewarded-unityads"
VUNGLE_AD_UNIT_ID = "demo-rewarded-vungle"
```

Native ads

Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

Yandex Mobile Ads SDK lets you render ads using your own visual assets.

Native ads can adapt to the features and design of the app they are displayed in. The layout of a native ad matches the environment it is integrated into. This type of ad looks natural and contributes useful information to the app.

The SDK also provides a set of ready-made customizable visual assets (templates) that let you enjoy all the benefits of rendering from the platform's native tools without creating your own design.

There are two types of ad loading:

- Loading one ad
- Loading multiple ads

See also

Advertising requirements
Classes and interfaces for working with native ads

Loading and rendering ads

Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

Getting started

To ensure that the Yandex Mobile Ads SDK runs correctly, follow all the steps to attach an ad library.

Loading ads

1. Create an instance of the NativeAdLoader class to get native ads.
2. Create a configuration for the `nativeAdRequestConfiguration` request using the NativeAdRequestConfiguration.Builder class. As the request parameters, you can use the ad unit ID, method for loading images, age, gender, and other data that might improve the quality of ad selection.
3. To get notifications (ad loaded successfully or failed with an error), create an instance of NativeAdLoadListener and set it as an event listener for the ad loader.
4. Start the ad loading process.

Ad request with the size set

To get an ad of the correct size, pass the maximum container width and height to an ad request using the setParameters method:

```
final NativeAdLoader loader = new NativeAdLoader(this);
final HashMap<String, String> parameters = new HashMap<String, String>(){{
    put("preferable-height", "123");
    put("preferable-width", "321");
}};
final NativeAdRequestConfiguration nativeAdRequestConfiguration =
        new NativeAdRequestConfiguration.Builder("demo-native-app-yandex")
                .setParameters(parameters).build();
NativeAdLoader mNativeAdLoader = new NativeAdLoader(this);
mNativeAdLoader.loadAd(nativeAdRequestConfiguration);
```

General ad request

```
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);
```

Tip:

We recommend that you keep a strong reference to the ad and its loader throughout the lifecycle of the screen hosting the assets.

Examples with a demo AdUnit ID

To see how the ad will be displayed in the app, use a demo AdUnit ID:

```
YANDEX_AD_UNIT_ID = "demo-native-content-yandex"
ADMOB_AD_UNIT_ID = "demo-native-content-admob"
MYTARGET_AD_UNIT_ID = "demo-native-content-mytarget"
STARTAPP_AD_UNIT_ID = "demo-native-content-startapp"
```

Rendering ads

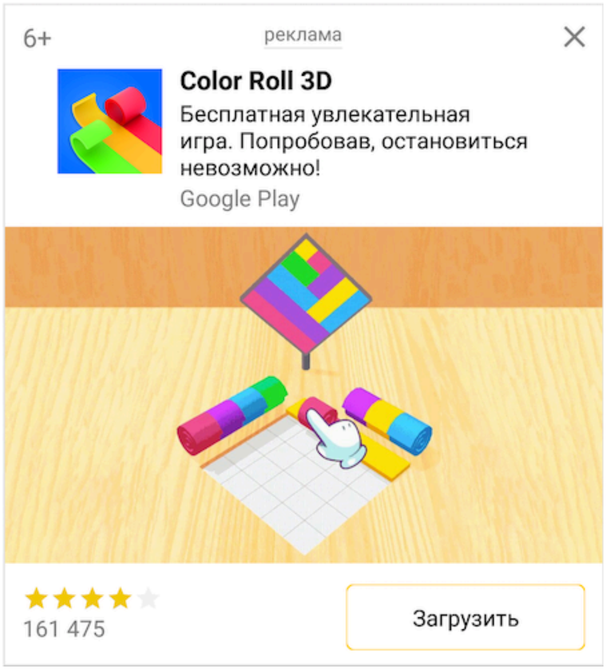
When the ad is loaded, you must render all of its assets. You can get the list of assets available in the ad from the NativeAd ad object.

There are two ways to configure the layout of an ad:

1. Layout using a template
2. Layout without a template

Layout using a template

Example of a native ad template


6+
реклама
Color Roll 3D
Бесплатная увлекательная игра. Попробовав, остановиться невозможно!
Google Play
161 475
Загрузить

The easiest way to work with native ads is to use a standard layout template: all you need is a few lines of code in the basic version.

The template already has the complete set of required assets and defines their arrangement relative to each other. The template works with any supported type of native ad.

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

You can customize the native ad template. Learn more in Layout using a template.

Layout without a template

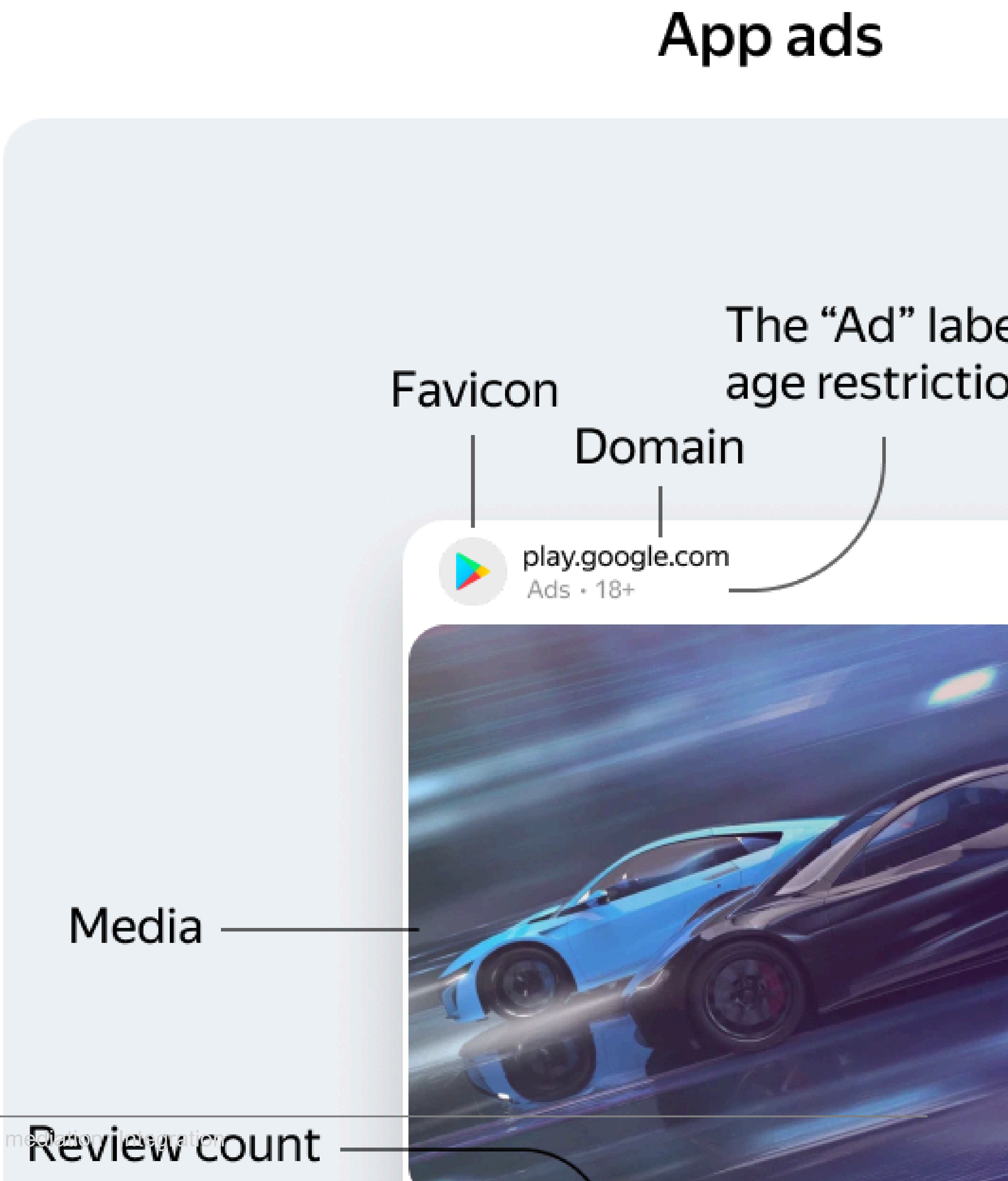
When the template settings aren't enough to get the desired effect, you can configure native ads manually.

With this method, you can lay out native ads yourself by positioning ad elements in respect of each other. Your ad may contain both mandatory and optional display assets. You can find their full list in Native ad assets.

Tip:

We recommend that you use a layout that includes the complete set of possible assets. Experience has shown that layouts with a complete set of assets are more clickable.

Example of a layout without using a template



Provide a View for each ad asset using the NativeAdViewBinder.Builder class instance. The class accepts the NativeAdView container as an argument. All the ad components must be defined as a subview of this container.

Link the generated ad layout with the NativeAd native ad object.

```
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
                    }
```

Note: mediaView size requirements when displaying video ads

Minimum size of an instance of the MediaView class that supports video playback: 300x160 or 160x300 dp (density-independent pixels).

To support video playback in native ad templates, we recommend setting the width for NativeBannerView to at least 300 dp. The correct height for mediaView will be calculated automatically based on the width to height ratio.

Loading multiple ads

Note:

Any call of the Mobile Ads SDK should be made from the main thread.

Yandex Mobile Ads SDK provides the option to load multiple ads in a single request (up to nine ads).

To make a bulk ad request, use the NativeBulkAdLoader class instance. The class instance provides the loadAds method with the COUNT argument where you can define the desired number of ads per request.

```
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);
```

Note:

Yandex Mobile Ads SDK doesn't guarantee that the requested number of ads will be loaded. The resulting array will contain from 0 to COUNT NativeAd objects. You can render all the received ad objects separately from each other using the above methods for laying out native ads.

Mobile mediation adapters

Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

Mobile mediation is a platform for automatically selecting ads from multiple ad networks. Each advertising network offers an ad to be displayed and the mediation platform chooses the most profitable one.

The mobile mediation platform integrates adapters from most of the major ad networks listed below.

List of supported ad networks

Yandex Advertising Network	Adapter version	Ad network SDK version	Banner Ad	Interstitial Ad	Rewarded Ad	Native Ad
AdMob	22.1.0.0	22.1.0	✓	✓	✓	✓
myTarget	5.17.0.0	5.17.0	✓	✓	✓	✓
Start.io	4.11.0.1	4.11.0	✓	✓	✓	✓
UnityAds	4.7.1.2	4.7.1	✓	✓	✓	✗
AppLovin	11.10.1.1	11.10.1	✓	✓	✓	✗
IronSource	7.3.1.1.0	7.3.1.1	✗	✓	✓	✗
AdColony	4.8.0.5	4.8.0	✓	✓	✓	✗
ChartBoost	9.3.0.2	9.3.0	✓	✓	✓	✗
Pangle	5.3.0.4.0	5.3.0.4	✗	✓	✓	✗
Tapjoy	13.0.1.1	13.0.1	✗	✓	✓	✗
Vungle	6.12.1.2	6.12.1	✓	✓	✓	✗
Mintegral	16.4.71.0	16.4.71	✓	✓	✓	✓

Enabling AdMob

Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.10.0.

Maximum supported version: 6.0.0 (up to but not including).

GoogleMobileAdsSDK

Minimum supported version: 22.1.0.

Maximum supported version: 22.2.0 (up to but not including).

Supported ad formats

- Banner ads
- Interstitial ads
- Rewarded ads
- Native ads

Integration

Enable using Gradle

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Add the following dependencies to the `build.gradle` file at the application level:

```
implementation 'com.yandex.android:mobileads:5.10.0'
implementation 'com.yandex.ads.mediation:mobileads-admob:22.1.0.0'
```

3. Add your `AdMob ID` to the AndroidManifest.xml file using a `<meta-data>` tag named com.google.android.gms.ads.APPLICATION_ID (learn more about how to find out the AdMob ID).

```
<manifest>
    <application>
        <meta-data
            android:name="com.google.android.gms.ads.APPLICATION_ID"
            android:value="ca-app-pub-xxxxxxxxxxxxxxxx~yyyyyyyyyy"/>
    </application>
</manifest>
```

Integration examples

- Banner ads
- Interstitial ads
- Rewarded ads
- Native ads

Enabling myTarget

Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.10.0.

Maximum supported version: 6.0.0 (up to but not including).

myTargetSDK

Minimum supported version: 5.17.0.

Maximum supported version: 5.18.0 (up to but not including).

Supported ad formats

- Banner ads
- Interstitial ads
- Rewarded ads
- Native ads

Integration

Enable using Gradle

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Add the following dependencies to the `build.gradle` file at the application level:

```
implementation 'com.yandex.android:mobileads:5.10.0'
implementation 'com.yandex.ads.mediation:mobileads-mytarget:5.17.0.0'
```

Enable manually

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Download the available 5.17.0 myTarget Android SDK version and add it to your project.
3. Download the latest version of the myTarget adapter (a file in the `.aar` format) from the repository.

Integration examples

- Banner ads
- Interstitial ads
- Rewarded ads
- Native ads

Enabling Start.io

Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.10.0.

Maximum supported version: 6.0.0 (up to but not including).

Start.io SDK

Minimum supported version: 4.11.0.

Maximum supported version: 4.12.0 (up to but not including).

Supported ad formats

- Banner ads
- Interstitial ads
- Rewarded ads
- Native ads

Integration

Note:

Before getting started, the Start.io mediation network displays a pop-up window requesting the user's consent to show them personalized ads (for more information, see the Start.io network documentation). To hide the pop-up window, pass the user's consent using the setUserConsent method.

Enable using Gradle

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Add the following dependencies to the `build.gradle` file at the application level:

```
implementation 'com.yandex.android:mobileads:5.10.0'
implementation 'com.yandex.ads.mediation:mobileads-startapp:4.11.0.1'
```

Enable manually

1. Set up mediation in the Yandex Partner interface and YAN interface.

2. Download the available 4.11.0 Start.io SDK version and add it to your project.
3. Download the latest version of the Start.io adapter (an `.aar` file) from the repository.

Integration examples

- Banner ads
- Interstitial ads
- Rewarded ads
- Native ads

Enabling UnityAds

Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.10.0.

Maximum supported version: 6.0.0 (up to but not including).

UnityAdsSDK

Minimum supported version: 4.7.1.

Maximum supported version: 4.8.0 (up to but not including).

Supported ad formats

- Banner ads
- Interstitial ads
- Rewarded ads

Integration

Enable using Gradle

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Add the following dependencies to the `build.gradle` file at the application level:

```
implementation 'com.yandex.android:mobileads:5.10.0'
implementation 'com.yandex.ads.mediation:mobileads-unityads:4.7.1.2'
```

Enable manually

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Download the available 4.7.1 UnityAds version and add it to your project.
3. Download the latest version of the UnityAds adapter (a file in the `.aar` format) from the repository.

Integration examples

- Interstitial ads
- Rewarded ads

Enabling AppLovin

Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.10.0.

Maximum supported version: 6.0.0 (up to but not including).

AppLovinSDK

Minimum supported version: 11.10.1.

Maximum supported version: 11.11.0 (up to but not including).

Supported ad formats

- Banner ads
- Interstitial ads
- Rewarded ads

Integration

Enable using Gradle

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Add the following dependencies to the `build.gradle` file at the application level:

```
implementation 'com.yandex.android:mobileads:5.10.0'
implementation 'com.yandex.ads.mediation:mobileads-applovin:11.10.1.1'
```

Integration examples

- Banner ads
- Interstitial ads
- Rewarded ads

Enabling IronSource

Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.10.0.

Maximum supported version: 6.0.0 (up to but not including).

IronSourceSDK

Minimum supported version: 7.3.1.1.

Maximum supported version: 7.4.0 (up to but not including).

Supported ad formats

- Interstitial ads
- Rewarded ads

Integration

Enable using Gradle

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Add the following dependencies to the `build.gradle` file at the application level:

```
implementation 'com.yandex.android:mobileads:5.10.0'
implementation 'com.yandex.ads.mediation:mobileads-ironsource:7.3.1.1.0'
```

Enable manually

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Download the available 7.3.1.1 IronSource SDK version and add it to your project.
3. Download the latest version of the IronSource adapter (a file in the `.aar` format) from the repository.

Integration examples

- Interstitial ads
- Rewarded ads

Enabling AdColony

Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.10.0.

Maximum supported version: 6.0.0 (up to but not including).

AdColonySDK

Minimum supported version: 4.8.0.

Maximum supported version: 4.9.0 (up to but not including).

Supported ad formats

- Banner ads
- Interstitial ads
- Rewarded ads

Integration

Enable using Gradle

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Add the following dependencies to the `build.gradle` file at the application level:

```
implementation 'com.yandex.android:mobileads:5.10.0'
implementation 'com.yandex.ads.mediation:mobileads-adcolony:4.8.0.5'
```

Enable manually

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Download the available 4.8.0 AdColony SDK version and add it to your project.
3. Download the latest version of the AdColony adapter (an `.aar` file) from the repository.

Integration examples

- Banner ads
- Interstitial ads
- Rewarded ads

Enabling ChartBoost

Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.10.0.

Maximum supported version: 6.0.0 (up to but not including).

ChartBoostSDK

Minimum supported version: 9.3.0.

Maximum supported version: 9.4.0 (up to but not including).

Supported ad formats

- Banner ads
- Interstitial ads
- Rewarded ads

Integration

Enable using Gradle

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Add the following dependencies to the `build.gradle` file at the application level:

```
implementation 'com.yandex.android:mobileads:5.10.0'
implementation 'com.yandex.ads.mediation:mobileads-chartboost:9.3.0.2'
```

Enable manually

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Download the available 9.3.0 ChartBoost SDK version and add it to your project.
3. Download the latest version of the ChartBoost adapter (an `.aar` file) from the repository.

Integration examples

- Banner ads
- Interstitial ads
- Rewarded ads

Enabling Pangle

Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.10.0.

Maximum supported version: 6.0.0 (up to but not including).

PangleSDK

Minimum supported version: 5.3.0.4.

Maximum supported version: 5.4.0.0 (up to but not including).

Supported ad formats

- Interstitial ads
- Rewarded ads

Integration

Note:

When setting up ad orientation in the Pangle network, make sure that the selected orientation matches the one in your app. Otherwise, the ad block won't be monetized.

Enable using Gradle

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Add the following dependencies to the `build.gradle` file at the application level:

```
implementation 'com.yandex.android:mobileads:5.10.0'
implementation 'com.yandex.ads.mediation:mobileads-pangle:5.3.0.4.0'
```

Enable manually

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Download the available 5.3.0.4 Pangle SDK version and add it to your project.
3. Download the latest version of the Pangle adapter (an `.aar` file) from the repository.

Integration examples

- Interstitial ads
- Rewarded ads

Enabling Tapjoy

Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.10.0.

Maximum supported version: 6.0.0 (up to but not including).

TapjoySDK

Minimum supported version: 13.0.1.

Maximum supported version: 13.1.0 (up to but not including).

Supported ad formats

- Interstitial ads
- Rewarded ads

Integration

Enable using Gradle

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Add the following dependencies to the `build.gradle` file at the application level:

```
implementation 'com.yandex.android:mobileads:5.10.0'
implementation 'com.yandex.ads.mediation:mobileads-tapjoy:13.0.1.1'
```

Enable manually

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Download the available 13.0.1 Tapjoy SDK version and add it to your project.
3. Download the latest version of the Tapjoy adapter (an `.aar` file) from the repository.

Integration examples

- Interstitial ads
- Rewarded ads

Enabling Vungle

Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.10.0.

Maximum supported version: 6.0.0 (up to but not including).

VungleSDK

Minimum supported version: 6.12.1.

Maximum supported version: 6.13.0 (up to but not including).

Supported ad formats

- Banner ads
- Interstitial ads
- Rewarded ads

Integration

Enable using Gradle

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Add the following dependencies to the `build.gradle` file at the application level:

```
implementation 'com.yandex.android:mobileads:5.10.0'
implementation 'com.yandex.ads.mediation:mobileads-vungle:6.12.1.2'
```

Enable manually

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Download the available 6.12.1 Vungle SDK version and add it to your project.
3. Download the latest version of the Vungle adapter (an `.aar` file) from the repository.

Integration examples

- Banner ads
- Interstitial ads
- Rewarded ads

Enabling Mintegral

Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.10.0.

Maximum supported version: 6.0.0 (up to but not including).

MintegralSDK

Minimum supported version: 16.4.71.

Maximum supported version: 16.5.0 (up to but not including).

Supported ad formats

- Banner ads
- Interstitial ads
- Rewarded ads
- Native ads

Integration

Enable using Gradle

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Add the following dependencies to the `build.gradle` file at the application level:

```
implementation 'com.yandex.android:mobileads:5.10.0'
implementation 'com.yandex.ads.mediation:mobileads-mintegral:16.4.71.0'
```

3. If you have problems loading the library, add the following code to the `build.gradle` file as described in the Mintegral documentation:

```
//Non-listed GP market applications# Android X Version
 maven {
        url "https://dl-maven-android.mintegral.com/repository/mbridge_android_sdk_china"
 }

 //Launch GP market application# Android X Version
 maven {
        url  "https://dl-maven-android.mintegral.com/repository/mbridge_android_sdk_oversea"
 }

 //Non-listed GP market applications, not the Android X version
 maven {
        url  "https://dl-maven-android.mintegral.com/repository/mbridge_android_sdk_support/"
 }
```

Enable manually

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Download the available 16.4.71 Mintegral SDK version and add it to your project.
3. Download the latest version of the Mintegral adapter (an `.aar` file) from the repository.

Integration examples

- Banner ads
- Interstitial ads
- Rewarded ads
- Native ads

Enabling BigoAds

Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found here.

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.10.0.

Maximum supported version: 6.0.0 (up to but not including).

BigoAdsSDK

Minimum supported version: 2.9.0.

Maximum supported version: 2.10.0 (up to but not including).

Supported ad formats

- Banner ads

- Interstitial ads
- Rewarded ads

Integration

Enable using Gradle

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Add the following dependencies to the `build.gradle` file at the application level:

```
implementation 'com.yandex.android:mobileads:5.10.0'
implementation 'com.yandex.ads.mediation:mobileads-bigoads:2.9.0.2'
```

Enable manually

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Download the available 2.9.0 BigoAds SDK version and add it to your project.
3. Download the latest version of the BigoAds adapter (an `.aar` file) from the repository.