Yandex Mobile Ads

Integration

10.07.2024

Yandex

Yandex Mobile Ads. 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

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This is an archived version of the documentation.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.

The Yandex Mobile Ads library is provided in AAR format. To enable the Mobile Ads SDK:

1. Add the dependency on Yandex Mobile Ads to the `build.gradle` file in your app's module:

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

2. Update the dependency on Koltin Gradle Plugin to the `build.gradle` root file in your project:

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

3. 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
    }
}
```

4. Set up permission to use the ad ID.

   The ad ID is a unique identifier provided by Google Play services for displaying ads to users who opt in to personalized ads. Users can opt out of ad personalization or reset their ID in the settings. In this case, advertising networks won't be able to use the ID to select relevant ads.

   Mobile Ads SDK version 4.5.0 and later

   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"/>
```

   Mobile Ads SDK version below 4.5.0

   If your app uses a version of the Yandex Mobile Ads SDK below 4.5.0 and there are no restrictions on the use of an ad ID (for example, Families Policy), add the permission to the application's main manifest `AndroidManifest.xml`:

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

   Lack of this permission and access to the ID may reduce the relevance of ads and, as a result, your income.

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.

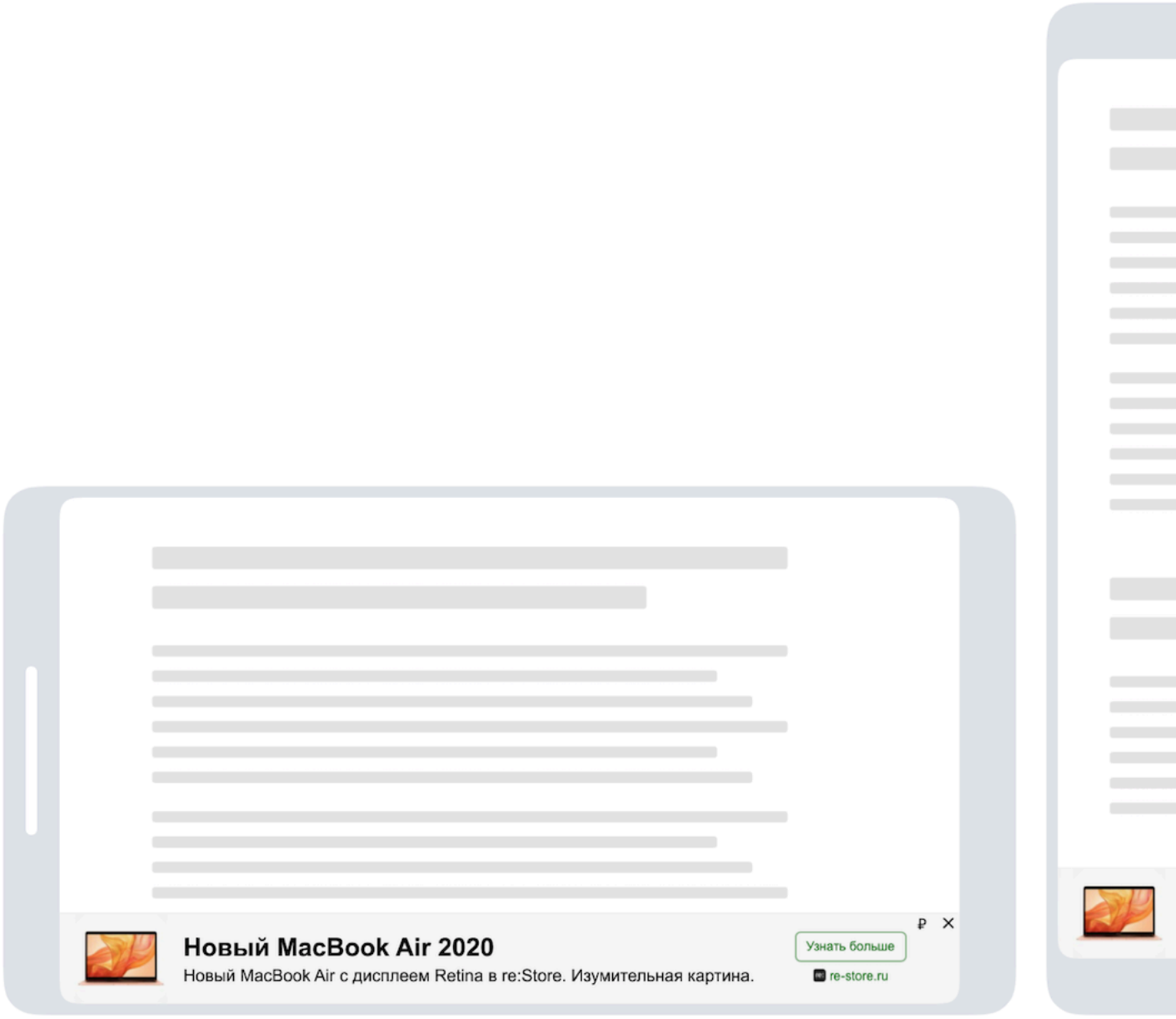
Banner types

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:

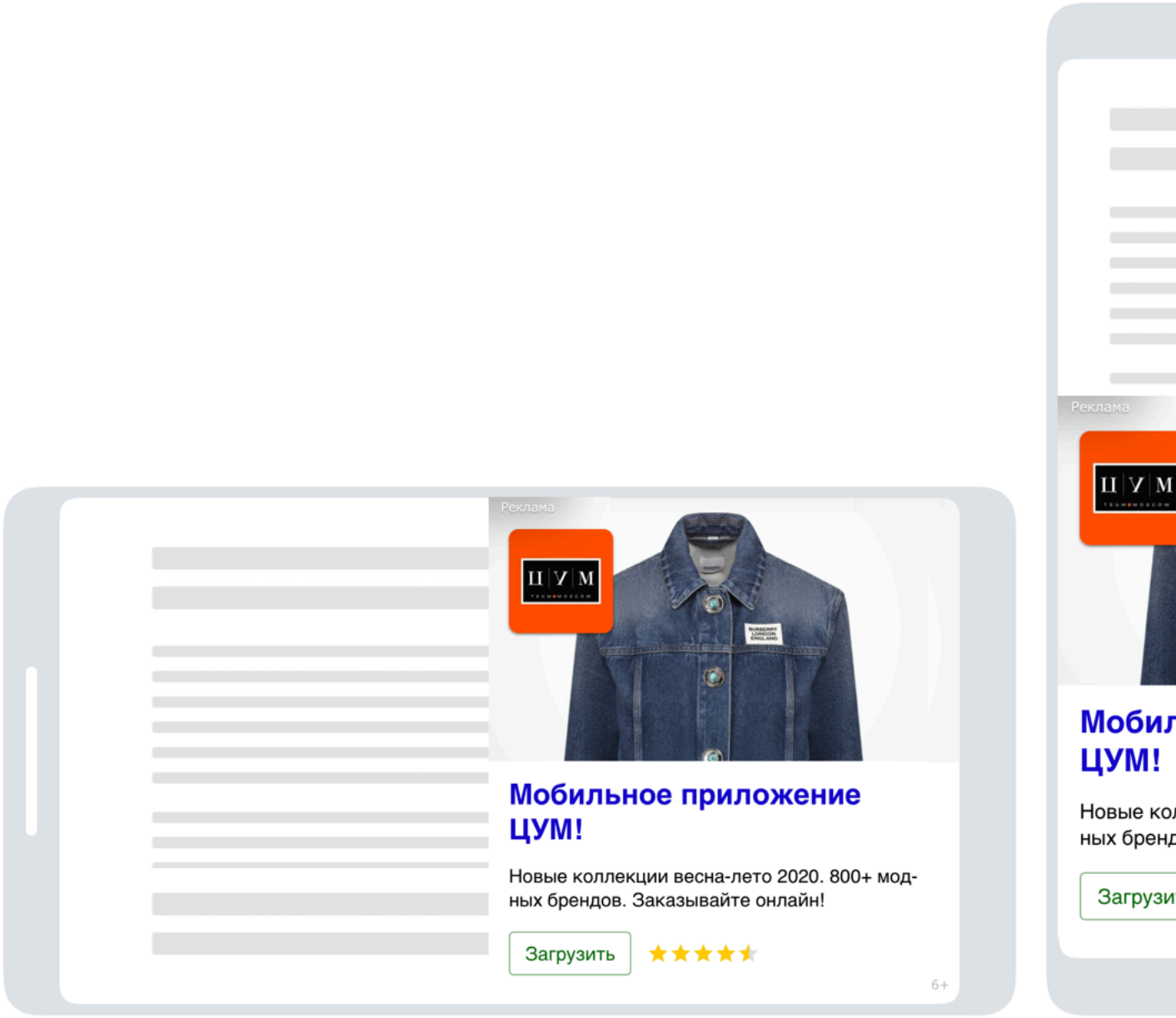


Flex banner

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:



Enabling a banner

Creating BannerAdview

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 an instance of the `AdView` class, you need to load an ad.

To load an ad, use the loadAd method, which takes an object of the AdRequest class as an argument:

```
mBannerAdView.loadAd(adRequest);
```

About loading ads

Use the `AdRequest` object to transmit the code received in the Adfox interface (for more information, see Help for Adfox):

```
// Code from the Adfox interface for working with direct campaigns.
Map<String, String> parameters = new HashMap<String, String>();
parameters.put("adf_ownerid", "example");
parameters.put("adf_p1", "example");
parameters.put("adf_p2", "example");
parameters.put("adf_pfc", "example");
parameters.put("adf_pfb", "example");
parameters.put("adf_plp", "example");
parameters.put("adf_pli", "example");
parameters.put("adf_pop", "example");
parameters.put("adf_pt", "example");
parameters.put("adf_pd", "example");
parameters.put("adf_pw", "example");
parameters.put("adf_pv", "example");
parameters.put("adf_prr", "example");
parameters.put("adf_pdw", "example");
parameters.put("adf_pdh", "example");
parameters.put("adf_puid1", "example");

final AdRequest adRequest = AdRequest.builder().withParameters(parameters).build();
```

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>
```

```
...
```

```
public class BannerExample extends Activity {
    ...
    private static final String adUnitId = "YOUR_adUnitId";
    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));

        // Code from the Adfox interface for working with direct campaigns.
        Map<String, String> parameters = new HashMap<String, String>();
        parameters.put("adf_ownerid", "example");
        parameters.put("adf_p1", "example");
        parameters.put("adf_p2", "example");
        parameters.put("adf_pfc", "example");
        parameters.put("adf_pfb", "example");
        parameters.put("adf_plp", "example");
        parameters.put("adf_pli", "example");
        parameters.put("adf_pop", "example");
        parameters.put("adf_pt", "example");
        parameters.put("adf_pd", "example");
        parameters.put("adf_pw", "example");
        parameters.put("adf_pv", "example");
        parameters.put("adf_prr", "example");
        parameters.put("adf_pdw", "example");
        parameters.put("adf_pdh", "example");
        parameters.put("adf_puid1", "example");

        // Creating an ad targeting object.
        final AdRequest adRequest = AdRequest.builder().withParameters(parameters).build();

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

        // 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:

- demo-banner-yandex

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:

```
mInterstitialAd.loadAd(adRequest);
```

About loading ads

Use the `AdRequest` object to transmit the code received in the Adfox interface (for more information, see Help for Adfox):

```
// Code from the Adfox interface for working with direct campaigns.
Map<String, String> parameters = new HashMap<String, String>();
parameters.put("adf_ownerid", "example");
parameters.put("adf_p1", "example");
parameters.put("adf_p2", "example");
parameters.put("adf_pfc", "example");
parameters.put("adf_pfb", "example");
parameters.put("adf_plp", "example");
parameters.put("adf_pli", "example");
parameters.put("adf_pop", "example");
parameters.put("adf_pt", "example");
parameters.put("adf_pd", "example");
```

```
parameters.put("adf_pw", "example");
parameters.put("adf_pv", "example");
parameters.put("adf_prr", "example");
parameters.put("adf_pdw", "example");
parameters.put("adf_pdh", "example");
parameters.put("adf_puid1", "example");

final AdRequest adRequest = AdRequest.builder().withParameters(parameters).build();
```

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.

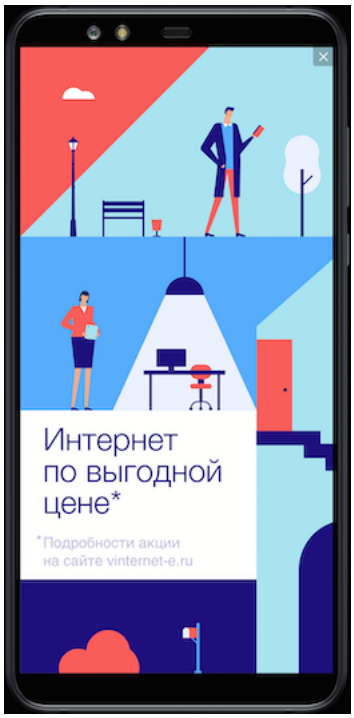
Example of working with interstitial ads

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 adUnitAd = "YOUR_adUnitId";
    private InterstitialAd mInterstitialAd;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        ...
        // Creating an InterstitialAd instance.
        mInterstitialAd = new InterstitialAd(this);
        mInterstitialAd.setAdUnitId(adUnitId);

        // Code from the Adfox interface for working with direct campaigns.
        Map<String, String> parameters = new HashMap<String, String>();
        parameters.put("adf_ownerid", "example");
        parameters.put("adf_p1", "example");
        parameters.put("adf_p2", "example");
        parameters.put("adf_pfc", "example");
        parameters.put("adf_pfb", "example");
        parameters.put("adf_plp", "example");
        parameters.put("adf_pli", "example");
        parameters.put("adf_pop", "example");
        parameters.put("adf_pt", "example");
        parameters.put("adf_pd", "example");
        parameters.put("adf_pw", "example");
        parameters.put("adf_pv", "example");
        parameters.put("adf_prr", "example");
        parameters.put("adf_pdw", "example");
        parameters.put("adf_pdh", "example");
        parameters.put("adf_puid1", "example");

        // Creating an ad targeting object.
        final AdRequest adRequest = AdRequest.builder().withParameters(parameters).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:



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

- demo-interstitial-yandex

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.

   `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:

- `demo-rewarded-yandex`

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 three types of ways to load ads (ad loading):

- Loading one ad
- Loading multiple ads
- Ad slider

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

Note:

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

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.

Rendering ads

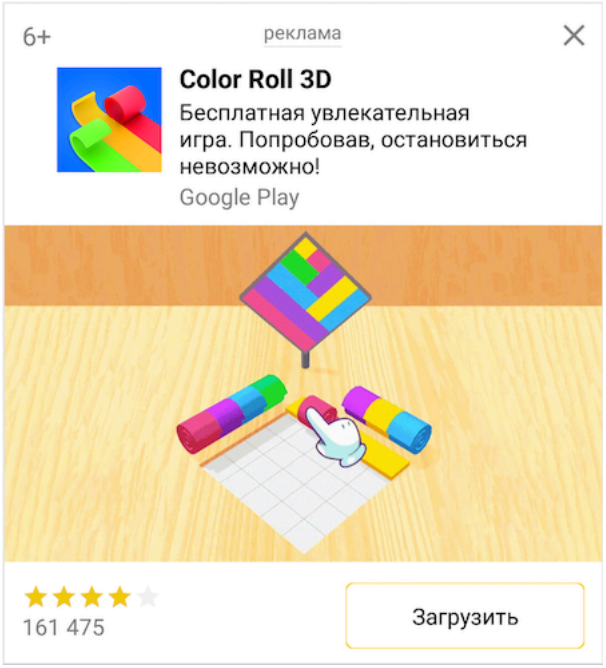
When the ad is loaded, you must render all of its assets. You can get the list of assets available in the ad from the NativeAd ad object.

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



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

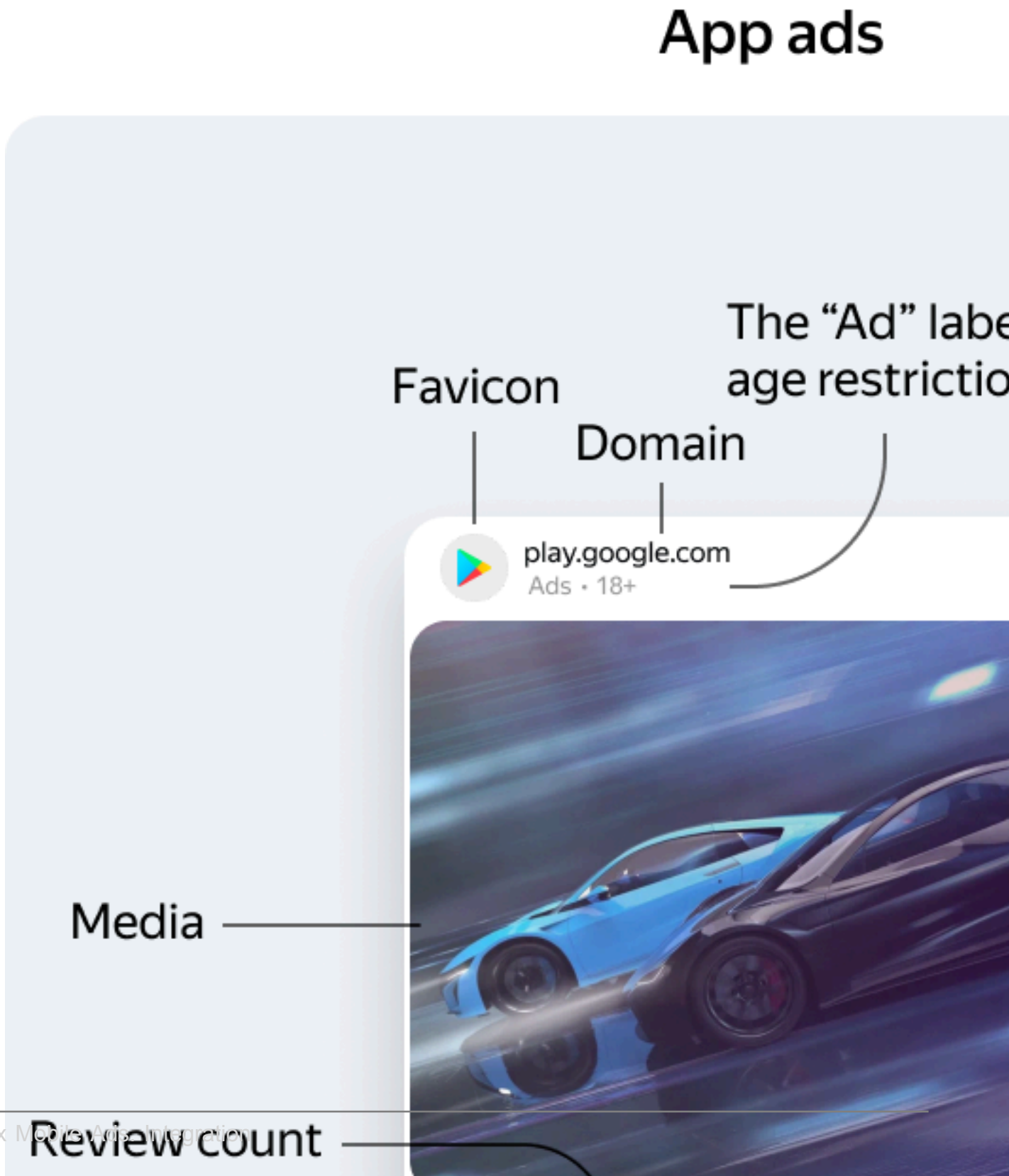
When the template settings aren't enough to get the desired effect, you can configure native ads manually.

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.

Ad slider

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 a slider containing related ads. For more information about the slider, see this post.

The slider implements the native advertising paradigm. Publishers can adjust the ad layout based on its features and the design of their apps that render the slider.

Getting started

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

Loading the slider

Note:

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

1. Create an instance of the SliderAdLoader class to fetch ads into the slider.
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 (slider loaded successfully or failed with an error), create a SliderAdLoadListener instance and set it as an event listener for the ad loader.
4. Start the slider loading process.

Code example:

```
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);
```

Tip:

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

Displaying the slider

When the slider is loaded, you must render all its assets.

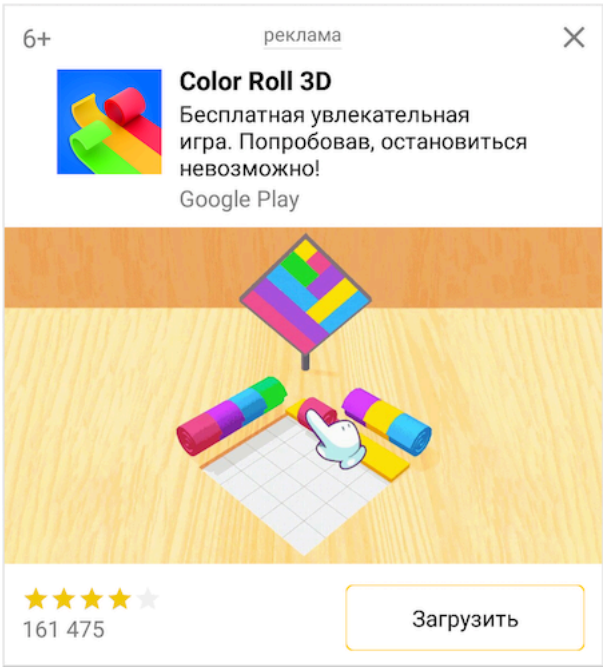
A successfully loaded slider contains one or more native ads with the same context. Ads in the slider must be displayed within the same shared container: otherwise, ad impressions won't count.

There are two ways to configure the slider layout:

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

Layout using a template

Example of a native ad template



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.

Making a slider using a template is easy. Link the SliderAd object to the root NativeAdView container and link all the ads in the slider to the template:

```
@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
    }
}
```

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

Layout without a template

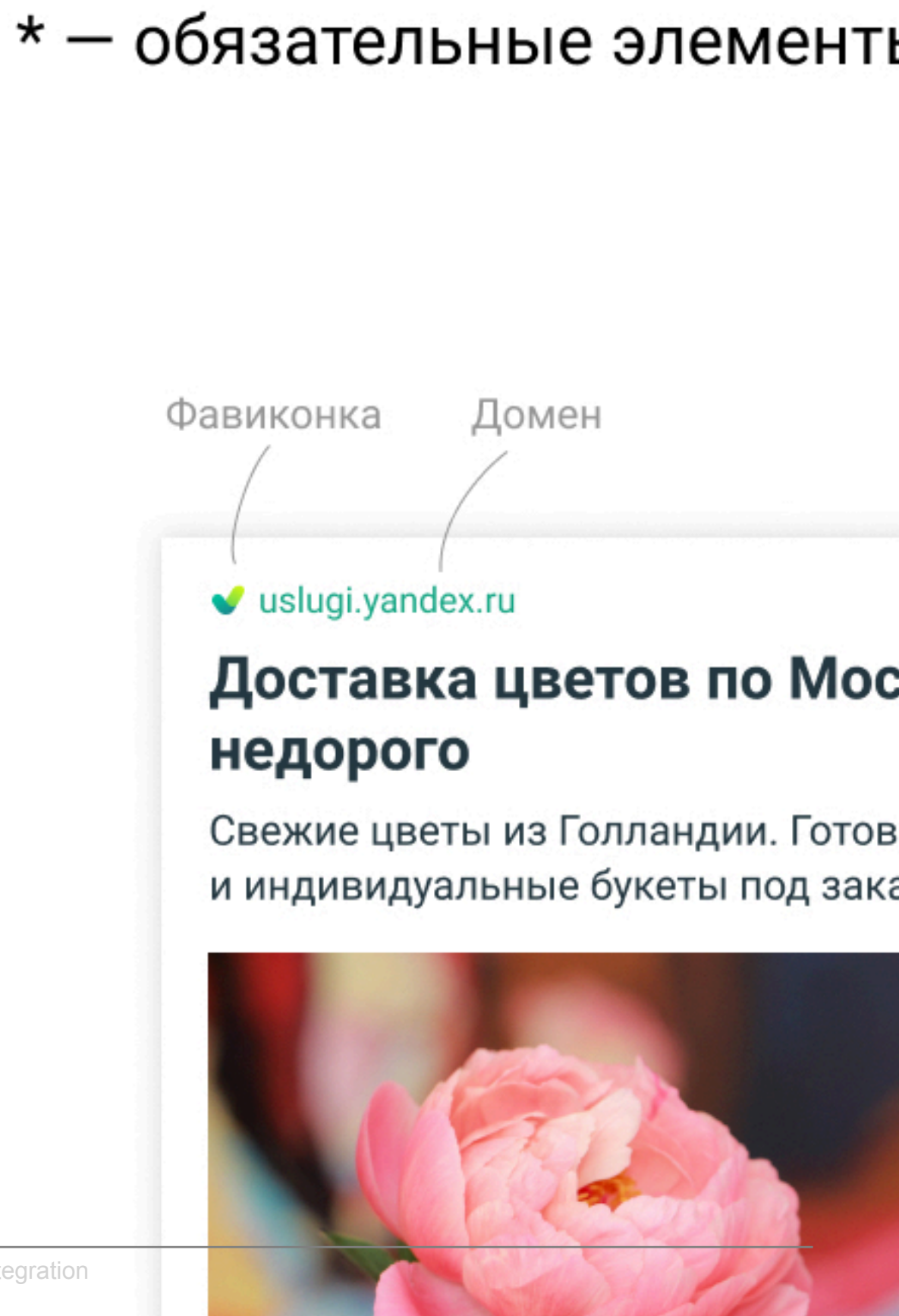
When the template settings aren't enough to get the desired effect, you can configure native ads manually.

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



Link the SliderAd object to the root NativeAdView object and link each ad in the slider to the child NativeAdView object.

Each ad in the slider is laid out using a standard native advertising layout method.

```
@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
    }
}
```

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.