Integration

Integration

10.07.2024

Yandex

Integration. Version 1.0

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

Adding the Yandex Mobile Ads Flutter plugin

Warning:

This is an archived version of the documentation.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

About the plugin

Flutter is a popular technology for creating mobile apps. Developers use it to write single code that can be run on different operating systems, including Android and iOS.

Now, it will be easier for you to monetize Android and iOS apps created with Flutter. All our partners in the Yandex Advertising Network can now use the plugin to enable Yandex-based monetization in their apps.

This plugin supports loading and display of the following ad types:

- Banner ads
- Interstitial ads
- Rewarded ads

Requirements

- Flutter 2.5.0 or higher.
- Android 4.1 or higher.

  Video ads are selected for devices with Android 5.0 or higher.
- iOS 12.0 or higher.

  Check the additional steps to ensure that iOS 14 (or higher) works properly.

  Latest Xcode version with enabled command-line tools.
- Registering an account in the Yandex Advertising Network.

Adding the plugin to your project

Install the Yandex Mobile Ads Flutter plugin in your project. From the root of the project, call the command:

```
flutter pub add yandex_mobileads
```

Once the plugin is added, you'll see the following line with a dependency in the `pubspec.yaml` file:

```
dependencies:
    yandex_mobileads: ^X.X.X
```

`X.X.X`: The plugin's current version number.

Configuring for specific platforms

Android

The new permission `com.google.android.gms.permission.AD_ID` is already added in the Yandex Mobile Ads SDK. It enables you to use AD_ID to select relevant ads. You can delete the permission if necessary.

Learn more about the permission and 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 for the user.

How to delete the permission

If some policies (such as Google Play's Families Policy) do not allow the use of ad IDs, you can delete the permission from the `AndroidManifest.xml` file.

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

iOS

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.

Initializing the library

In initState of your app's widget, add the line: `MobileAds.initialize();`.

See also

Enabling the plugin for Mobile mediation
COPPA on page 9
GDPR on page 9

Ad formats

Enabling 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.

Enabling a banner

1. Add the import:

```
import 'package:yandex_mobileads/mobile_ads.dart';
```

2. Create an object of the `BannerAd` class in the Builder or the `initState` of the widget where you want to place the banner:

```
final banner = BannerAd(
  adUnitId: 'R-M-XXXXXX-Y',
  // Flex-size
  adSize: AdSize.flexible(width: screenWidth, height: bannerHeight),
  // Sticky-size
  adSize: AdSize.sticky(width: screenWidth),
  adRequest: AdRequest(),
  onAdLoaded: () {
      /* Do something */
 },
 onAdFailedToLoad: (error) {
      /* Do something */
 },
);
```

3. In the `build` method of your widget, specify the `AdWidget` that will accept the new `BannerAd` object:

```
AdWidget(bannerAd: banner)
```

4. The ad will load automatically. If necessary, you can call the `load` method of the banner object after you call `build` for the first time.

Example of working with banner ads

The code demonstrates how to create and configure `BannerAd` objects:

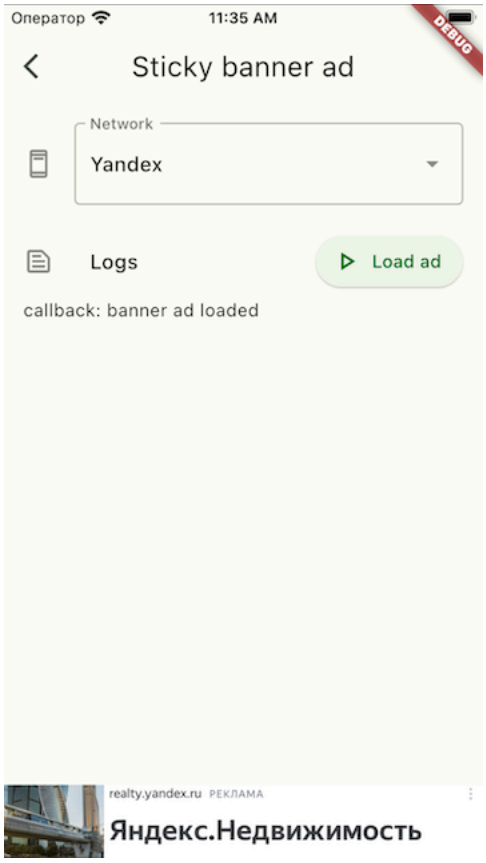
```
import 'package:yandex_mobileads/mobile_ads.dart';

// ...

class BannerAdPage extends StatelessWidget {
  final banner = BannerAd(
    adUnitId: 'demo-banner-yandex',
    adSize: AdSize.sticky(width: screenWidth),
    adRequest: AdRequest(),
    onAdLoaded: () {
        /* Do something */
    },
    onAdFailedToLoad: (error) {
        /* Do something */
    },
  );

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Align(
        alignment: Alignment.bottomCenter,
        child: AdWidget(bannerAd: banner),
      ),
    );
  }
}
```

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



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

- demo-banner-yandex

Enabling 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 full-screen ad that reacts to clicks.

Creating and displaying InterstitialAd

1. Add the import:

```
import 'package:yandex_mobileads/mobile_ads.dart';
```

2. Create an object of the `InterstitialAd` class. You can only use the asynchronous method to create the object:

```
final ad = await InterstitialAd.create(
  adUnitId: 'R-M-XXXXXX-Y',
  onAdLoaded: () {
     /* Do something */
  },
  onAdFailedToLoad: (error) {
     /* Do something */
 },
);
```

3. After you create and configure the `InterstitialAd` class object, load the ads. To load an ad, use the `load` method that accepts an optional `AdRequest` object.

```
await ad.load(adRequest: AdRequest());
```

   You can wait for your ad to load asynchronously using the `await` keyword. If there's an error, the `onAdFailedToLoad` method is called to terminate waiting.

4. Interstitial ads are loaded in the background immediately after the `load` method is called. To display an interstitial ad, call the `show` method:

```
await ad.show();
```

   You can wait for the ad to start displaying asynchronously.

5. Alternatively, you can use the `waitForDismiss` asynchronous method to wait until the end of ad serving:

```
await ad.waitForDismiss();
```

Example of working with interstitial ads

The code demonstrates how to create and configure an `InterstitialAd` object, and how to load and display an interstitial ad:

```
Future<void> showInterstitialAd() async {
  final ad = await InterstitialAd.create(
    adUnitId: 'demo-interstitial-yandex',
    onAdLoaded: () {
        /* Do something */
    },
    onAdFailedToLoad: (error) {
        /* Do something */
    },
  );
  await ad.load(adRequest: AdRequest());
  await ad.show();
  await ad.waitForDismiss();
}
```

If an ad is integrated this way, the ad unit appears when `showInterstitialAd` is called.

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

- demo-interstitial-yandex

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.

Creating and displaying RewardedAd

1. Add the import:

```
import 'package:yandex_mobileads/mobile_ads.dart';
```

2. Create a `RewardedAd` class object. You can only use the asynchronous method to create the object:

```
final ad = await RewardedAd.create(
 adUnitId: 'R-M-XXXXXX-Y',
 onAdLoaded: () {
     /* Do something */
 },
 onAdFailedToLoad: (error) {
     /* Do something */
 },
);
```

3. After you create the `RewardedAd` class object, load the ads. To load an ad, use the `load` method that accepts an optional `AdRequest` object.

```
await ad.load(adRequest: AdRequest());
```

   You can wait for your ad to load asynchronously using the `await` keyword.

4. Rewarded ads are loaded in the background immediately after the `load` method is called. To display an ad, call the `show` method.

```
await ad.show();
```

   You can wait for the ad to start displaying asynchronously.

5. Alternatively, you can use the `waitForDismiss` asynchronous method to wait until the end of the ad serving and/or reward:

```
Reward? reward = await ad.waitForDismiss();
```

Example of working with rewarded ads

The code demonstrates how to create and configure a `RewardedAd` object, and how to load and display a rewarded ad:

```
Future<void> showRewardedAd() async {
  final ad = await RewardedAd.create(
    adUnitId: 'demo-rewarded-yandex',
    onAdLoaded: () {
        /* Do something */
    },
    onAdFailedToLoad: (error) {
        /* Do something */
    },
  );
  await ad.load(adRequest: AdRequest());
  await ad.show();
  final reward = await ad.waitForDismiss();
  if (reward != null) debugPrint('got ${reward.amount} of ${reward.type}');
}
```

If an ad is integrated this way, the ad unit appears when `showRewardedAd` is called.

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

- `demo-rewarded-yandex`

GDPR

Warning:

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

General information

The General Data Protection Regulation (GDPR) came into effect in the spring of 2018. GDPR regulates how information about citizens of the European Economic Area and Switzerland can be collected and processed. Its intention is to protect the privacy of confidential data and to ensure the transparency of all processes related to the collection, storage and processing of information on the internet.

The GDPR has an extraterritorial scope that applies to all companies that process the personal data of citizens of the European Economic Area and Switzerland, regardless of where the company is located.

The Yandex Mobile Ads SDK Flutter can limit the collection of data from users located in the European Economic Area and Switzerland who have not consented to share their confidential information.

Quick guide

The user's consent for processing personal data must be sent to the SDK each time the application is launched.

1. Follow the instructions to integrate the Mobile Ads SDK Flutter.
2. Show a window where the user can accept the user agreement for personal data processing.
3. Use the `setUserConsent` method to pass the received value to the Mobile Ads SDK.

```
MobileAds.setUserConsent(true);
```

4. For users from GDPR regions, the data will be processed only if they give their consent to it.

COPPA

Warning:

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

General information

On April 21, 2000, the Children's Online Privacy Protection Act (COPPA) came into effect. The act governs the collection of personal information from children under the age of 13 by individuals or entities within US jurisdiction. Pursuant to the Act, the application operator shall include in their privacy policy the methods for obtaining parental or caretaker's consent, as well as the operator's commitment to protecting the privacy and safety of children on the internet, including marketing restrictions.

Data about the user's age must be transmitted to the SDK every time the application starts.

Quick guide

Data about the user's age must be transmitted to the SDK every time the application starts.

1. Follow the instructions to integrate the Mobile Ads SDK Flutter.
2. Show a window where the user can accept the user agreement for personal data processing.

3. Use the `setAgeRestrictedUser` method to pass the received value to the Mobile Ads SDK. By default, it is assumed that the user isn't a child under the age of 13.

```
MobileAds.setAgeRestrictedUser(true);
```