Mobile mediation

Integration

10.07.2024

Yandex

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

Adding the Mobile Ads Unity plugin

Warning:

This is an archived version of the documentation.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

Mobile Ads Unity is a plugin for the Unity3d gaming platform. This plugin provides support for the Yandex Mobile Ads SDK.

Note:

1. To run the SDK, you need the Target API Level version 31 or higher.
2. To upload ads of any type, you need iOS 12.0 or higher.

Integrating the plugin

Note: `yandex-ads-unity-plugin` runs only in Android and iOS environments. You can't use it in the Unity editor.

Lite version

1. Download the directory `yandex-ads-unity-plugin` and add the package `yandex-mobileads-lite-2.9.0.unitypackage`. You will be prompted to install the Google resolver with the package. If you already added the Google resolver to your project, clear the checkbox.

   How to add the package

   Select your plugin (**Assets** → **Import Package** → **Custom Package**), then click **Import**.
2. Use the Google resolver to install dependencies: enable **auto-resolve** or select **Assets** → **External Dependency Manager** → **Android Resolver** → **Resolve** in the menu.
3. To test the Mobile Ads Unity plugin, use a sample script from the **samples** directory in the `yandex-ads-unity-plugin` repository. Copy the script to the project directory and add it as a **Component** to the main camera.

Enabling mobile mediation

Enabling all available adapters automatically

You can enable all available adapters automatically using the `yandex-mobileads-mediation` shared mediation package.

1. Download the directory `yandex-ads-unity-plugin` and add the package `Yandex-mobileads-mediation-2.9.0.unitypackage`. You will be prompted to install the Google resolver with the package. If you already added the Google resolver to your project, clear the checkbox.

   How to add the package

   Select your plugin (**Assets** → **Import Package** → **Custom Package**), then click **Import**.
2. Use the Google resolver to install dependencies: enable **auto-resolve** or select **Assets** → **External Dependency Manager** → **Android Resolver** → **Resolve** in the menu.
3. To test the Mobile Ads Unity plugin, use a sample script from the **samples** directory in the `yandex-ads-unity-plugin` repository. Copy the script to the project directory and add it as a **Component** to the main camera.

Enabling a specific adapter using the appropriate library

If you don't need to enable all available adapters automatically, follow the instructions for enabling the necessary adapters only.

Downgrading the Target API Level

To downgrade the Target API Level to version 30, add the explicit downgrade to `mainTemplate.gradle` and `launcherTemplate.gradle` (if you use launcherTemplate in the project):

```
configurations.all {
    resolutionStrategy {
        force 'androidx.core:core:1.6.0'
        force 'androidx.core:core-ktx:1.6.0'
    }
```

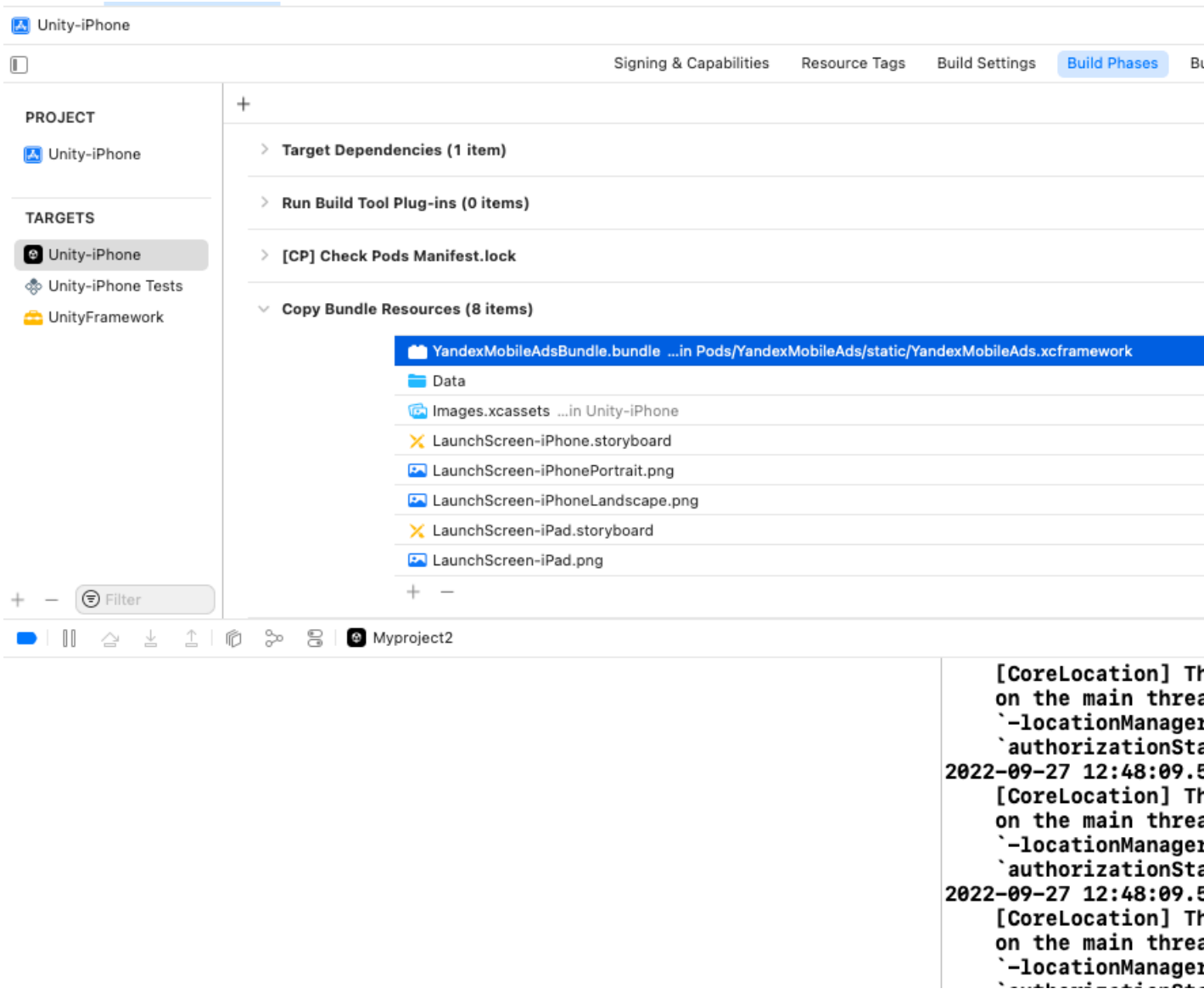
```
}
```

However, we recommend that you upgrade to the Target API Level 31, because Google has restrictions on releasing updates for applications running an outdated version of the Target API Level. Learn more in the article.

Error description

Unity Interstitial Ads aren't displayed, error “Incorrect fullscreen view”

The error “Incorrect fullscreen view” may arise when launching interstitial ads on iOS. If this issue arises, make sure that you added the value **YandexMobileAdsBundle.bundle** under **Copy Bundle Resources** in the **Build Phases** settings. If the value is missing, add it.



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 responds to clicks.

Adding `Banner` to the project

To display a banner in your app, create a `Banner` object in the script (in C#) that is attached to the `GameObject`.

```
...
using YandexMobileAds;
using YandexMobileAds.Base;
...

public class YandexMobileAdsBannerDemoScript : MonoBehaviour
{
    private Banner banner;
    ...
    private void RequestBanner()
    {
        string adUnitId = "YOUR_adUnitId";

        banner = new Banner(adUnitId, AdSize.BANNER_320x50, AdPosition.BottomCenter);
    }
    ...
}
```

The `Banner` constructor contains the following parameters:

- `AdUnitId`: A unique identifier that is issued in the Partner interface and takes the format R-M-XXXXXX-Y.

  Note:

  In the mobile mediation interface, the `AdUnitId` is called the location ID.
- `AdSize`: The size of the banner you want to display.
- `AdPosition`: The position on the screen.

Loading ads

Once the `Banner` object has been created and configured, load the ads. To load your ads, use the `LoadAd` method which takes the `AdRequest` object as an argument.

```
...
private void RequestBanner()
{
    ...
    AdRequest request = new AdRequest.Builder().Build();

    banner.LoadAd(request);
    ...
}
...
```

Banner ad events

To track events that occur in banner ads, register a delegate for the appropriate `EventHandler`, as shown below:

```
...
private void RequestBanner()
{
    ...
        banner.OnAdLoaded += HandleAdLoaded;
        banner.OnAdFailedToLoad += HandleAdFailedToLoad;
        banner.OnReturnedToApplication += HandleReturnedToApplication;
        banner.OnLeftApplication += HandleLeftApplication;
        banner.OnAdClicked += HandleAdClicked;
        banner.OnImpression += HandleImpression;
    ...
}

public void HandleAdLoaded(object sender, EventArgs args)
{
    MonoBehaviour.print("HandleAdLoaded event received");
    banner.Show();
}

public void HandleAdFailedToLoad(object sender, AdFailureEventArgs args)
{
    MonoBehaviour.print("HandleAdFailedToLoad event received with message: " + args.Message);
}

public void HandleLeftApplication(object sender, EventArgs args)
```

```
{
    MonoBehaviour.print("HandleLeftApplication event received");
}

public void HandleReturnedToApplication(object sender, EventArgs args)
{
    MonoBehaviour.print("HandleReturnedToApplication event received");
}

public void HandleAdLeftApplication(object sender, EventArgs args)
{
    MonoBehaviour.print("HandleAdLeftApplication event received");
}

public void HandleAdClicked(object sender, EventArgs args)
{
    MonoBehaviour.print("HandleAdClicked event received");
}

public void HandleImpression(object sender, ImpressionData impressionData)
{
    var data = impressionData == null ? "null" : impressionData.rawData;
    MonoBehaviour.print("HandleImpression event received with data: " + data);
}
```

Clearing ads

When an ad object is no longer needed, you can delete it. To do this, call the `Destroy` method:

```
banner.Destroy();
```

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.

Adding `Interstitial` to the project

To enable advertising, create an `Interstitial` object in the script (in C#) that is attached to the `GameObject`.

```
...
using YandexMobileAds;
using YandexMobileAds.Base;
...

public class YandexMobileAdsInterstitialDemoScript : MonoBehaviour
{
    private Interstitial interstitial;
    ...
    private void RequestInterstitial()
    {
        string adUnitId = "YOUR_adUnitId";

        interstitial = new Interstitial(adUnitId);
    }
    ...
}
```

The `Interstitial` constructor contains the `adUnitId` parameter, which is a unique identifier that is assigned in the Partner interface and takes the format R-M-XXXXXX-Y.

Note:

In the mobile mediation interface, the `AdUnitId` is called the location ID.

Loading ads

After creating and configuring the object of the `Interstitial` class, load the ads. To load your ads, use the `LoadAd` method which takes the `AdRequest` object as an argument.

```
...
private void RequestInterstitial()
```

```
{
    ...
    AdRequest request = new AdRequest.Builder().Build();

    interstitial.LoadAd(request);
    ...
}
...
```

Displaying ads

After the ad has loaded, you can display it:

```
...
private void ShowInterstitial()
{
    if (this.interstitial.IsLoaded())
    {
        interstitial.Show();
    }
    else
    {
        Debug.Log("Interstitial is not ready yet");
    }
}
...
```

Interstitial ad events

To track events that occur in interstitial ads, register a delegate for the appropriate `EventHandler`, as shown below:

```
...
private void RequestInterstitial()
{
    ...
    interstitial.OnInterstitialLoaded += HandleInterstitialLoaded;
    interstitial.OnInterstitialFailedToLoad += HandleInterstitialFailedToLoad;
    interstitial.OnReturnedToApplication += HandleReturnedToApplication;
    interstitial.OnLeftApplication += HandleLeftApplication;
    interstitial.OnAdClicked += HandleAdClicked;
    interstitial.OnInterstitialShown += HandleInterstitialShown;
    interstitial.OnInterstitialDismissed += HandleInterstitialDismissed;
    interstitial.OnImpression += HandleImpression;
    interstitial.OnInterstitialFailedToShow += HandleInterstitialFailedToShow;
    ...
}

public void HandleInterstitialLoaded(object sender, EventArgs args)
{
    MonoBehaviour.print("HandleInterstitialLoaded event received");
}

public void HandleInterstitialFailedToLoad(object sender, AdFailureEventArgs args)
{
    MonoBehaviour.print(
        "HandleInterstitialFailedToLoad event received with message: " + args.Message);
}

public void HandleReturnedToApplication(object sender, EventArgs args)
{
    MonoBehaviour.print("HandleReturnedToApplication event received");
}

public void HandleLeftApplication(object sender, EventArgs args)
{
    MonoBehaviour.print("HandleLeftApplication event received");
}

public void HandleAdClicked(object sender, EventArgs args)
{
    MonoBehaviour.print("HandleAdClicked event received");
}

public void HandleInterstitialShown(object sender, EventArgs args)
{
    MonoBehaviour.print("HandleInterstitialShown event received");
}

public void HandleInterstitialDismissed(object sender, EventArgs args)
{
    MonoBehaviour.print("HandleInterstitialDismissed event received");
}

public void HandleImpression(object sender, ImpressionData impressionData)
{
    var data = impressionData == null ? "null" : impressionData.rawData;
```

```
    MonoBehaviour.print("HandleImpression event received with data: " + data);
}

public void HandleInterstitialFailedToShow(object sender, AdFailureEventArgs args)
{
    MonoBehaviour.print(
        "HandleInterstitialFailedToShow event received with message: " + args.Message);
}
```

Clearing ads

When an ad object is no longer needed, you can delete it. To do this, call the `Destroy` method:

```
interstitial.Destroy();
```

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.

Adding a `rewarded` ad to the project

To enable advertising, create a `RewardedAd` object in the script (in C#) that is attached to the `GameObject`.

```
...
using YandexMobileAds;
using YandexMobileAds.Base;
...

public class YandexMobileAdsRewardedAdDemoScript : MonoBehaviour
{
    private RewardedAd rewardedAd;
    ...
    private void RequestRewardedAd()
    {
        string adUnitId = "YOUR_adUnitId";

        rewardedAd = new RewardedAd(adUnitId);
    }
    ...
}
```

The `RewardedAd` constructor contains the `adUnitId` parameter, a unique identifier that is assigned in the Partner interface that takes the format R-M-XXXXXX-Y.

Note:

In the mobile mediation interface, the `AdUnitId` is called the location ID.

Loading ads

After creating and configuring the `RewardedAd` class object, you need to load ads. To load your ads, use the `LoadAd` method which takes the `AdRequest` object as an argument.

```
...
private void RequestRewardedAd()
{
    ...
    AdRequest request = new AdRequest.Builder().Build();

    rewardedAd.LoadAd(request);
    ...
}
...
```

Displaying ads

After the ad has loaded, you can display it:

```
...
private void ShowRewardedAd()
{
    if (this.rewardedAd.IsLoaded())
    {
        rewardedAd.Show();
    }
    else
    {
        Debug.Log("Rewarded Ad is not ready yet");
    }
}
...
```

Rewarded ad events

To track events that occur in a rewarded ad, register a delegate for the appropriate `EventHandler`, as shown below:

```
...
private void RequestRewardedAd()
{
    ...
    rewardedAd.OnRewardedAdLoaded += HandleRewardedAdLoaded;
    rewardedAd.OnRewardedAdFailedToLoad += HandleRewardedAdFailedToLoad;
    rewardedAd.OnReturnedToApplication += HandleReturnedToApplication;
    rewardedAd.OnLeftApplication += HandleLeftApplication;
    rewardedAd.OnAdClicked += HandleAdClicked;
    rewardedAd.OnRewardedAdShown += HandleRewardedAdShown;
    rewardedAd.OnRewardedAdDismissed += HandleRewardedAdDismissed;
    rewardedAd.OnImpression += HandleImpression;
    rewardedAd.OnRewarded += HandleRewarded;
    rewardedAd.OnRewardedAdFailedToShow += HandleRewardedAdFailedToShow;
    ...
}

public void HandleRewardedAdLoaded(object sender, EventArgs args)
{
    MonoBehaviour.print("HandleRewardedAdLoaded event received");
}

public void HandleRewardedAdFailedToLoad(object sender, AdFailureEventArgs args)
{
    MonoBehaviour.print(
        "HandleRewardedAdFailedToLoad event received with message: " + args.Message);
}

public void HandleReturnedToApplication(object sender, EventArgs args)
{
    MonoBehaviour.print("HandleReturnedToApplication event received");
}

public void HandleLeftApplication(object sender, EventArgs args)
{
    MonoBehaviour.print("HandleLeftApplication event received");
}

public void HandleAdClicked(object sender, EventArgs args)
{
    MonoBehaviour.print("HandleAdClicked event received");
}

public void HandleRewardedAdShown(object sender, EventArgs args)
{
    MonoBehaviour.print("HandleRewardedAdShown event received");
}

public void HandleRewardedAdDismissed(object sender, EventArgs args)
{
    MonoBehaviour.print("HandleRewardedAdDismissed event received");
}

public void HandleImpression(object sender, ImpressionData impressionData)
{
    var data = impressionData == null ? "null" : impressionData.rawData;
    MonoBehaviour.print("HandleImpression event received with data: " + data);
}

public void HandleRewarded(object sender, Reward args)
{
    MonoBehaviour.print("HandleRewarded event received: amout = " + args.amount + ", type = " + args.type);
}

public void HandleRewardedAdFailedToShow(object sender, AdFailureEventArgs args)
{
```

```
    MonoBehaviour.print(
        "HandleRewardedAdFailedToShow event received with message: " + args.Message);
}
```

Clearing ads

When an ad object is no longer needed, you can delete it. To do this, call the `Destroy` method:

```
rewardedAd.Destroy();
```

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

Ad network	Banner Ad	Interstitial Ad	Rewarded Ad
AdMob	✓	✓	✓
myTarget	✓	✓	✓
Start.io (Android)	✓	✓	✓
UnityAds	✗	✓	✓
AppLovin	✓ (Android)	✓	✓
IronSource	✗	✓	✓
AdColony (Android)	✓	✓	✓
ChartBoost (Android)	✓	✓	✓
Pangle (Android)	✗	✓	✓
Tapjoy (Android)	✓	✓	✓
Vungle (Android)	✓	✓	✓
Mintegral	✓	✓	✓

Enabling AdMob

Warning:

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

Supported ad formats

- Banner ads
- Interstitials
- Rewarded ads

Integration

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Import the package `yandex-mobileads-lite-2.9.0.unitypackage` to the project.
3. Import `mobileads-admob-mediation-2.9.0.unitypackage` from the **mobileads-mediation** directory.
4. Create the main AndroidManifest.xml file by clicking **File** → **Build Settings** → **Android** → **Player Settings** → **Publishing settings** → **Custom Main Manifest** (select the checkbox).

   Add your `AdMob ID` to the created AndroidManifest.xml file of the app using the `<meta-data>` tag named com.google.android.gms.ads.APPLICATION_ID (how to find AdMob ID).

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

Enabling myTarget

Warning:

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

Supported ad formats

- Banner ads
- Interstitials
- Rewarded ads

Integration

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Import the package `yandex-mobileads-lite-2.9.0.unitypackage` to the project.
3. Import `mobileads-mytarget-mediation-2.9.0.unitypackage` from the **mobileads-mediation** directory.

Enabling Start.io

Warning:

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

Supported ad formats

- Banner ads
- Interstitials
- Rewarded ads

Integration

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Import the package `yandex-mobileads-lite-2.9.0.unitypackage` to the project.
3. Import `mobileads-startapp-mediation-2.9.0.unitypackage` from the **mobileads-mediation** directory.

Enabling UnityAds

Warning:

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

Supported ad formats

- Interstitials
- Rewarded ads

Integration

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Import the package `yandex-mobileads-lite-2.9.0.unitypackage` to the project.
3. Import `mobileads-unityads-mediation-2.9.0.unitypackage` from the **mobileads-mediation** directory.

Error description

Duplicate classes “Duplicate class com.unity3d.ads.BuildConfig found in modules...”

If you have a module from Unity Ads installed, the build will fail, and you'll see the class duplication error:

```
CommandInvokationFailure: Gradle build failed.
/Applications/Unity/Hub/Editor/2021.3.6f1/PlaybackEngines/AndroidPlayer/OpenJDK/bin/java -classpath
 "/Applications/Unity/Hub/Editor/2021.3.6f1/PlaybackEngines/AndroidPlayer/Tools/gradle/lib/gradle-
launcher-6.1.1.jar" org.gradle.launcher.GradleMain "-Dorg.gradle.jvmargs=-Xmx4096m" "assembleRelease"

stderr[
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8

FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':launcher:checkReleaseDuplicateClasses'.
> 1 exception was raised by workers:
  java.lang.RuntimeException: Duplicate class com.unity3d.ads.BuildConfig found in modules jetified-UnityAds-
runtime.jar (:UnityAds:) and jetified-com.unity3d.ads.unity-ads-4.2.1-runtime.jar (:com.unity3d.ads.unity-
ads-4.2.1:)
  Duplicate class com.unity3d.ads.IUnityAdsInitializationListener found in modules jetified-UnityAds-runtime.jar
 (:UnityAds:) and jetified-com.unity3d.ads.unity-ads-4.2.1-runtime.jar (:com.unity3d.ads.unity-ads-4.2.1:)
  Duplicate class com.unity3d.ads.IUnityAdsLoadListener found in modules jetified-UnityAds-runtime.jar
 (:UnityAds:) and jetified-com.unity3d.ads.unity-ads-4.2.1-runtime.jar (:com.unity3d.ads.unity-ads-4.2.1:)
  Duplicate class com.unity3d.ads.IUnityAdsShowListener found in modules jetified-UnityAds-runtime.jar
 (:UnityAds:) and jetified-com.unity3d.ads.unity-ads-4.2.1-runtime.jar (:com.unity3d.ads.unity-ads-4.2.1:)
  Duplicate class com.unity3d.ads.IUnityAdsTokenListener found in modules jetified-UnityAds-runtime.jar
 (:UnityAds:) and jetified-com.unity3d.ads.unity-ads-4.2.1-runtime.jar (:com.unity3d.ads.unity-ads-4.2.1:)
  Duplicate class com.unity3d.ads.UnityAds found in modules jetified-UnityAds-runtime.jar (:UnityAds:) and
 jetified-com.unity3d.ads.unity-ads-4.2.1-runtime.jar (:com.unity3d.ads.unity-ads-4.2.1:)
  Duplicate class com.unity3d.ads.UnityAds$UnityAdsInitializationError found in modules jetified-UnityAds-
runtime.jar (:UnityAds:) and jetified-com.unity3d.ads.unity-ads-4.2.1-runtime.jar (:com.unity3d.ads.unity-
ads-4.2.1:)
  Duplicate class com.unity3d.ads.UnityAds$UnityAdsLoadError found in modules jetified-UnityAds-runtime.jar
 (:UnityAds:) and jetified-com.unity3d.ads.unity-ads-4.2.1-runtime.jar (:com.unity3d.ads.unity-ads-4.2.1:)
  Duplicate class com.unity3d.ads.UnityAds$UnityAdsShowCompletionState found in modules jetified-UnityAds-
runtime.jar (:UnityAds:) and jetified-com.unity3d.ads.unity-ads-4.2.1-runtime.jar (:com.unity3d.ads.unity-
ads-4.2.1:)
  Duplicate class com.unity3d.ads.UnityAds$UnityAdsShowError found in modules jetified-UnityAds-runtime.jar
 (:UnityAds:) and jetified-com.unity3d.ads.unity-ads-4.2.1-runtime.jar (:com.unity3d.ads.unity-ads-4.2.1:)
  Duplicate class com.unity3d.ads.UnityAdsBaseOptions found in modules jetified-UnityAds-runtime.jar
 (:UnityAds:) and jetified-com.unity3d.ads.unity-ads-4.2.1-runtime.jar (:com.unity3d.ads.unity-ads-4.2.1:)
  Duplicate class com.unity3d.ads.UnityAdsLoadOptions found in modules jetified-UnityAds-runtime.jar
 (:UnityAds:) and jetified-com.unity3d.ads.unity-ads-4.2.1-runtime.jar (:com.unity3d.ads.unity-ads-4.2.1:)
  Duplicate class com.unity3d.ads.UnityAdsShowOptions found in modules jetified-UnityAds-runtime.jar
 (:UnityAds:) and jetified-com.unity3d.ads.unity-ads-4.2.1-runtime.jar (:com.unity3d.ads.unity-ads-4.2.1:)
  Duplicate class com.unity3d.ads.metadata.InAppPurchaseMetaData found in modules jetified-UnityAds-runtime.jar
 (:UnityAds:) and jetified-com.unity3d.ads.unity-ads-4.2.1-runtime.jar (:com.unity3d.ads.unity-ads-4.2.1:)
  Duplicate class com.unity3d.ads.metadata.MediationMetaData found in modules jetified-UnityAds-runtime.jar
 (:UnityAds:) and jetified-com.unity3d.ads.unity-ads-4.2.1-runtime.jar (:com.unity3d.ads.unity-ads-4.2.1:)
  Duplicate class com.unity3d.ads.metadata.MetaData found in modules jetified-UnityAds-runtime.jar (:UnityAds:)
 and jetified-com.unity3d.ads.unity-ads-4.2.1-runtime.jar (:com.unity3d.ads.unity-ads-4.2.1:)
...
```

To fix the error, delete the Unity Ads module (**Window** → **Package manager**).

Enabling AppLovin

Warning:

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

Supported ad formats

- Banner ads (for Android only)
- Interstitials
- Rewarded ads

Integration

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Import the package `yandex-mobileads-lite-2.9.0.unitypackage` to the project.
3. Import `mobileads-applovin-mediation-2.9.0.unitypackage` from the **mobileads-mediation** directory.

Enabling IronSource

Warning:

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

Supported ad formats

- Interstitials
- Rewarded ads

Integration

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Import the package `yandex-mobileads-lite-2.9.0.unitypackage` to the project.
3. Import `mobileads-ironsource-mediation-2.9.0.unitypackage` from the **mobileads-mediation** directory.

Enabling AdColony

Warning:

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

Supported ad formats

- Banner ads
- Interstitials
- Rewarded ads

Integration

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Import the package `yandex-mobileads-lite-2.9.0.unitypackage` to the project.

3. Import `mobileads-adcolony-mediation-2.9.0.unitypackage` from the **mobileads-mediation** directory.

Enabling ChartBoost

Warning:

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

Supported ad formats

- Banner ads
- Interstitials
- Rewarded ads

Integration

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Import the package `yandex-mobileads-lite-2.9.0.unitypackage` to the project.
3. Import `mobileads-chartboost-mediation-2.9.0.unitypackage` from the **mobileads-mediation** directory.

Enabling Pangle

Warning:

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

Supported ad formats

- Banner ads
- Interstitials
- Rewarded ads

Integration

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Import the package `yandex-mobileads-lite-2.9.0.unitypackage` to the project.
3. Import `mobileads-pangle-mediation-2.9.0.unitypackage` from the **mobileads-mediation** directory.

Enabling Tapjoy

Warning:

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

Supported ad formats

- Banner ads
- Interstitials
- Rewarded ads

Integration

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Import the package `yandex-mobileads-lite-2.9.0.unitypackage` to the project.
3. Import `mobileads-tapjoy-mediation-2.9.0.unitypackage` from the **mobileads-mediation** directory.

Enabling Vungle

Warning:

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

Supported ad formats

- Banner ads
- Interstitials
- Rewarded ads

Integration

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Import the package `yandex-mobileads-lite-2.9.0.unitypackage` to the project.
3. Import `mobileads-vungle-mediation-2.9.0.unitypackage` from the **mobileads-mediation** directory.

Enabling Mintegral

Warning:

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

Supported ad formats

- Banner ads
- Interstitials
- Rewarded ads

Integration

1. Set up mediation in the Yandex Partner interface and YAN interface.
2. Import the package `yandex-mobileads-lite-2.9.0.unitypackage` to the project.
3. Import `mobileads-mintegral-mediation-2.9.0.unitypackage` from the **mobileads-mediation** directory.