

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.

© 2008—2024 Yandex LLC. All rights reserved.

Copyright Disclaimer

Yandex (and its applicable licensor) has exclusive rights for all results of intellectual activity and equated to them means of individualization, used for development, support, and usage of the service Mobile mediation. It may include, but not limited to, computer programs (software), databases, images, texts, other works and inventions, utility models, trademarks, service marks, and commercial denominations. The copyright is protected under provision of Part 4 of the Russian Civil Code and international laws.

You may use Mobile mediation or its components only within credentials granted by the Terms of Use of Mobile mediation or within an appropriate Agreement.

Any infringements of exclusive rights of the copyright owner are punishable under civil, administrative or criminal Russian laws.

Contact information

Yandex LLC

<https://www.yandex.com>

Ten.: +7 495 739 7000

Email: pr@yandex-team.ru

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

Contents

SKAdNetwork support.....	4
Note.....	4
Integrating the Mobile Ads SDK.....	4
Enabling mobile mediation.....	8
SKAdNetwork support.....	8
.....	18
Ad formats.....	19
Banner ads.....	19
Banner types.....	19
Enabling a banner.....	21
Example of working with banners.....	24
Interstitial ads.....	25
Rewarded ads.....	26
Native ads.....	28
Integration.....	28
Loading ads.....	29
Configuring the ad design.....	32
Debugging.....	39
Mobile mediation adapters.....	40
AdMob.....	41
myTarget.....	42
Start.io.....	43
UnityAds.....	43
AppLovin.....	44
IronSource.....	45
Mintegral.....	46
AdColony.....	46
Chartboost.....	47

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>zq4921623r.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, you need iOS 12.0 or later.
2. To ensure that iOS 14 (or higher) runs correctly, keep in mind the [additional steps](#).

**Attention:**

The SDK requires enabling the resources located in `YandexMobileAdsBundle.bundle` from `YandexMobileAds.framework`. When you integrate the SDK via CocoaPods, these resources are enabled automatically.

When using a customized `YandexMobileAds.framework` integration, make sure that `YandexMobileAds.bundle` is copied to the project resources.

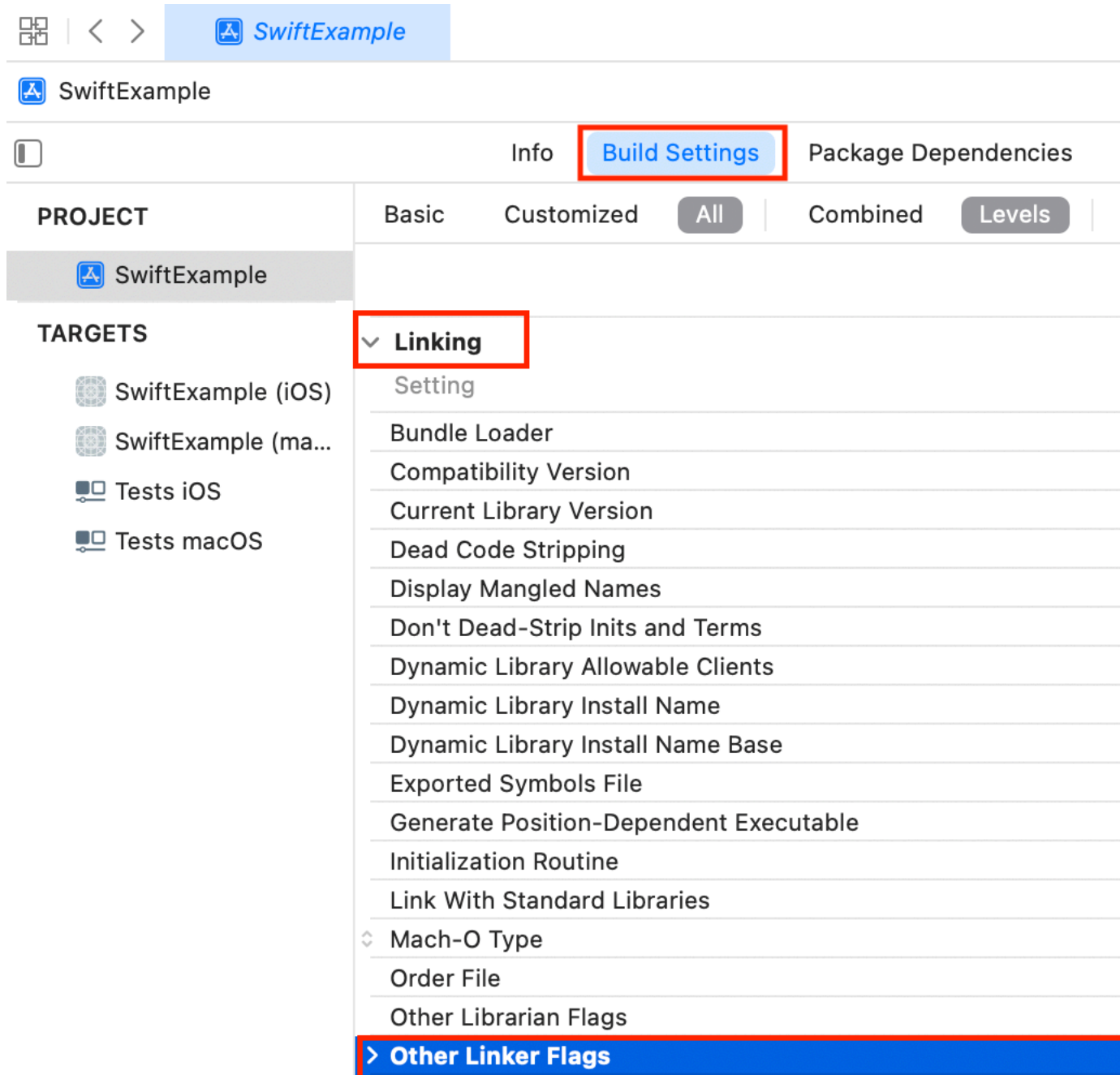
The library is compatible with the following dependency management systems:

Swift Package Manager**Note:**

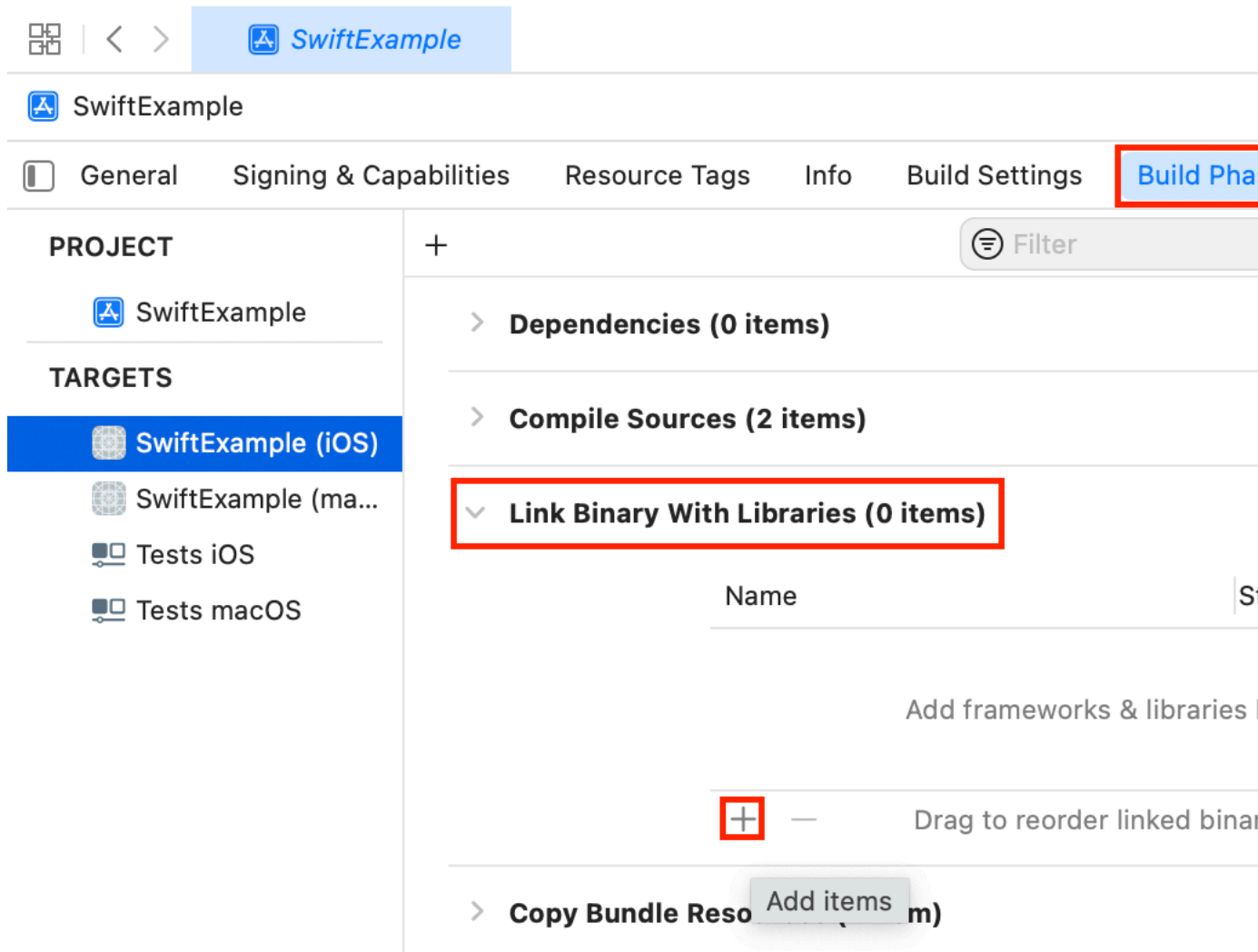
Mediation adapters can't be added via Swift Package Manager. If you use mediation, use integration via CocoaPods.

To attach the library, follow these steps:

1. In Xcode, add a dependency to your project using **File** → **Add Packages**.
2. Specify the repository URL `https://github.com/yandexmobile/yandex-ads-sdk-swift`: it includes the Swift package.
3. In the **Build Settings**, under **Linking**, add the Other Linker Flags = `-ObjC` parameter value.

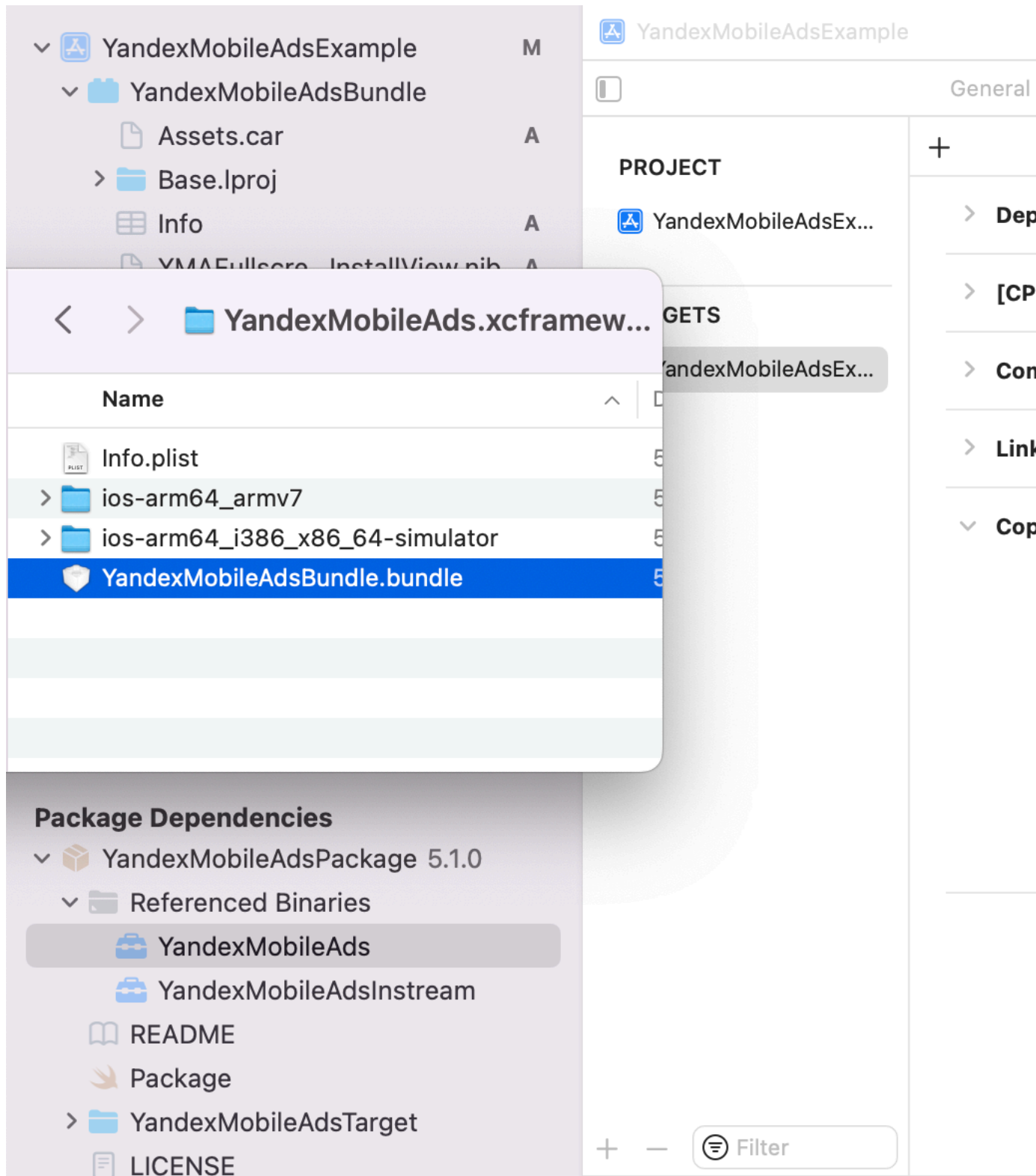


4. Check that the target is linked to YandexMobileAdsPackage. If the target is not linked, add a link to the library under **Link Binary With Libraries** by clicking the + icon.



5. Add YandexMobileAdsBundle.bundle from YandexMobileAds (**Package Dependencies** → **YandexMobileAdsPackage** → **Referenced Binaries** → **YandexMobileAds** → **Right click** → **Show in Finder** →

YandexMobileAds.xcframework) to the Copy Bundle Resource phase (Your target → Build Phases → Copy Bundle Resource).



CocoaPods

The Yandex Mobile Ads SDK library has been adapted to work with the CocoaPods dependency management system and supports a static integration method. To connect the library, add the following dependencies to the project's Podfile (see the [example for a static framework](#)):

```
pod 'YandexMobileAds', '5.9.1'
```

```
pod 'YandexMobileAdsInstream', '0.18.0'
```

Enabling mobile mediation

There are two ways to enable mobile mediation:

Enabling all available adapters automatically

You can enable all available adapters automatically using the `YandexMobileAdsMediation` shared mediation library. The library has been adapted to work with the CocoaPods dependency management system and supports a static integration method. To do this, add the following dependency to the project Podfile:

```
pod 'YandexMobileAdsMediation', '5.9.1'
```

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

Note:

To track app installations in mobile mediation, configure `Info.plist`. For more information, see [SKAdNetwork support](#).

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.

Note:

To track app installations in mobile mediation, configure `Info.plist`. For more information, see [SKAdNetwork support](#).

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 feature, add the IDs of the supported ad networks to the `Info.plist` file of your application.

```
<key>SKAdNetworkItems</key>
<array>
  <dict>
    <key>SKAdNetworkIdentifier</key>
    <string>zq492l623r.skadnetwork</string>
  </dict>
  <dict>
    <key>SKAdNetworkIdentifier</key>
    <string>633vhxsw4.skadnetwork</string>
  </dict>
  <dict>
    <key>SKAdNetworkIdentifier</key>
    <string>tmhh9296z4.skadnetwork</string>
  </dict>
  <dict>
    <key>SKAdNetworkIdentifier</key>
    <string>vcra2ehyfk.skadnetwork</string>
  </dict>
  <dict>
    <key>SKAdNetworkIdentifier</key>
    <string>zh3b7bxvad.skadnetwork</string>
  </dict>
  <dict>
    <key>SKAdNetworkIdentifier</key>
    <string>xmn954pzmp.skadnetwork</string>
  </dict>
  <dict>
    <key>SKAdNetworkIdentifier</key>
    <string>79w64w269u.skadnetwork</string>
  </dict>
  <dict>
    <key>SKAdNetworkIdentifier</key>
    <string>488r3q3dtq.skadnetwork</string>
  </dict>
  <dict>
    <key>SKAdNetworkIdentifier</key>
    <string>d7g9azk84q.skadnetwork</string>
  </dict>
</array>
</dict>
```



```
<key>SKAdNetworkIdentifier</key>
<string>nzq8sh4pbs.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>866k9ut3g3.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>2q884k2j68.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>x8jxxk4ff5.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>gfat3222tu.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>pd25vrrwzn.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>lr83yxwka7.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>cp8zw746q7.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>pwdxu55a5a.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>c6k4g5qq8m.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>s39g8k73mm.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>wg4vff78zm.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>g28c52eehv.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>523jb4fst2.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>294199pt4k.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>3qy4746246.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>a8cz6cu7e5.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>ggvn48r87g.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>y755zyxw56.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>qlbq5gtkt8.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>m7s7yz5dvl.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>67369282zy.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>899vrgt9g8.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>mj797d8u6f.skadnetwork</string>
</dict>
```

```
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>3sh42y64q3.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>f38h382jlk.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>24t9a8vw3c.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>mp6xlyr22a.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>x44k69ngh6.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>88k8774x49.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>hs6bdukanm.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>t3b3f7n3x8.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>prcb7njmu6.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>c7g47wypnu.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>52f12v3hgg.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>9vvzujtq5s.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>m8dbw4sv7c.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>9g2aggbj52.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>m5mvw97r93.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>z5b3gh5ugf.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>dd3a75yxkv.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>9nlqeag3gk.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>cj5566h2ga.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>h5jmj969g5.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>dr774724x4.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>t7ky8fmwkd.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>fz2k2k5tej.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>u679fj5vs4.skadnetwork</string>
</dict>
```

```
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>cs644xg564.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>9b89h5y424.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>w28pnjg2k4.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>2rq3zucswp.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>a7xqa6mtl2.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>g2y4y55b64.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>vc83br9sjg.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>cstr6suwn9.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>eqhxz8m8av.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>7k3cvf297u.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>w9q455wk68.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>nu4557a4je.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>v4nxqhlyqp.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>wzmmz9fp6w.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>7fmhfwg9en.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>su67r6k2v3.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>yclnxr15pm.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>7tnzynbdc7.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>l6nv3x923s.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>h8vml93bkz.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>uzqba5354d.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>8qieqk9qfv.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>v79kvwwj4g.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
```

```
<string>xx9sdjej2w.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>au67k4efj4.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>t38b2kh725.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>7ug5zh24hu.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>rx5hdcabgc.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>5lm9lj6jb7.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>qqp299437r.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>zmvfpc5aq8.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>9rd848q2bz.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>79pbpufp6p.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>dmv22haz9p.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>y5ghdn5j9k.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>n6fk4nfna4.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>7rz58n8ntl.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>v9wttfbk9.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>n38lu8286q.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>feyaarzu9v.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>7fbxrn65az.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>47vhws6wlr.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>ejvt5qm6ak.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>b55w3d8y8z.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>v7896pgt74.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>5ghnmfs3dh.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>275upjj5gd.skadnetwork</string>
</dict>
</dict>
```

```
<key>SKAdNetworkIdentifier</key>
<string>627r9wr2y5.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>kbd757ywx3.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>sczv5946wb.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>8w3np9l82g.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>hb56zgv37p.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>9t245vhmpl.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>nrt9jy4kw9.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>7953jerfzd.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>dn942472g5.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>6v7lgmsu45.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>cad8qz2s3j.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>n9x2a789qt.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>r26jy69rp1.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>eh6m2bh4zr.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>jb7bn6koa5.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>fkak3gfpt6.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>a2p9lx4jpn.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>97r2b46745.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>22mmun2rn5.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>238da6jt44.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>44jx6755aq.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>b9bk5wbcq9.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>k674qkevps.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>t155sbb4fm.skadnetwork</string>
</dict>
```

```
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>24zw6aqk47.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>4468km3ulz.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>2tdux391x8.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>2u9pt9hc89.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>8s468mf13y.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>3cgn6rq224.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>glqzh8vgby.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>av6w8kgt66.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>klf5c315u5.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>nfqy3847ph.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>dticjx1a9i.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>ppxm28t8ap.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>9wsyqb3ku7.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>74b6s63p6l.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>xy9t38ct57.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>424m5254lk.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>qu637u8glc.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>f73kdq92p3.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>44n7hlldy6.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>kbmxgpxpgc.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>5l3tpt7t6e.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>ecpz2srf59.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>x5854y7y24.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>f7s53z58qe.skadnetwork</string>
```

```
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>x8uqf25wch.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>uw77j35x4d.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>6964rsfnh4.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>gvmwg8q7h5.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>6xyyv74ff7.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>84993kbrcf.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>54nzkqm89y.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>pwa73g5rt2.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>mlmmfzh3r3.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>9yg77x724h.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>n66cz3y3bx.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>578prtvx9j.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>4dzt52r2t5.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>bvpn9ufa9b.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>6qx585k4p6.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>mtkv5xtk9e.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>193v5h6a4m.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>rvh3l7un93.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>gta9lk7p23.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>5tjdwbrq8w.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>r45fhh6rf7.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>32z4fx6l9h.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>e5fvkxwrpn.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
```

```
<string>8c4e2ghe7u.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>axh5283zss.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>3rd42ekr43.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>5mv394q32t.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>3qcr597p9d.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>v72qych5uu.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>ydx93a7ass.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>4pfyvq9l8r.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>5a6flpkh64.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>4fzdc2evr5.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>4w7y6s5ca2.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>252b5q8x7y.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>2fnua5tdw4.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>3l6bd9hu43.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>4mn522wn87.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>6g9af3uyq4.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>6p4ks3rnbw.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>6xzipu9s2p8.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>737z793b9f.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>89z7zv988g.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>8m87ys6875.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>8r8llnkz5a.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>b xvub5ada5.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>c3frkrj4fj.skadnetwork</string>
</dict>
</dict>
```



```
<key>SKAdNetworkIdentifier</key>
<string>cg4yq2srnc.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>dbu4b84rxf.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>dkc879ngq3.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>dzg6xy7pwj.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>gta8lk7p23.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>hdw39hrw9y.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>hjevpa356n.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>krvm3zuq6h.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>ln5gz23vtd.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>ludvb6z3bs.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>m297p6643m.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>p78axw29g.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>pu4na253f3.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>s69wq72ugq.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>t6d3zquu66.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>vutu7akeur.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>x2jnk7ly8j.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>x5183yy675.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>y45688jllp.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>yrqqpx2mcb.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>z4gj7hsk7h.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>wzmmZ9fp6w.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>4pfyvq9L8r.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>V72QYCH5UU.skadnetwork</string>
</dict>
```

```
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>2U9PT9HC89.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>3RD42EKR43.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>4FZDC2EVR5.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>7953JERFZD.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>7UG5ZH24HU.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>9RD848Q2BZ.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>9T245VHMPL.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>CJ5566H2GA.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>F38H382JLK.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>KLF5C3L5U5.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>M8DBW4SV7C.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>W9Q455WK68.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>WZMMZ9FP6W.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>XY9T38CT57.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>YCLNXRL5PM.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>4PFYVQ9L8R.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>TL55SBB4FM.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>BD757YWX3.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>33r6p7g8nc.skadnetwork</string>
</dict>
<dict>
  <key>SKAdNetworkIdentifier</key>
  <string>g69uk9uh2b.skadnetwork</string>
</dict>
</array>
```

For more information, see [Configuring a Source App](#) in the Apple documentation.

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 `+stickySizeWithContainerWidth:` method.
3. The banner height shouldn't exceed 15% of the device height and should be at least 50 dp.

Examples of displaying banners:

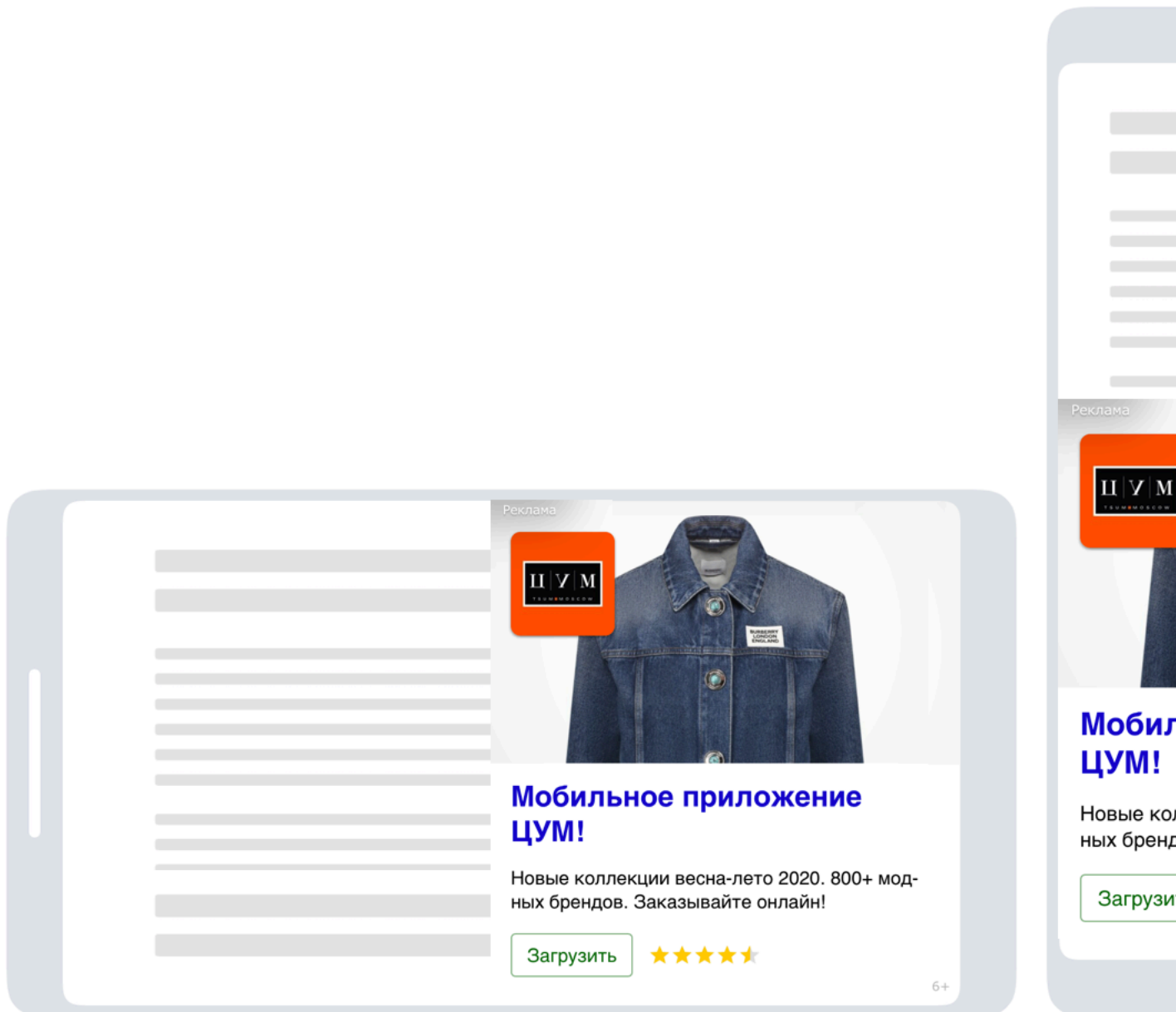


Flex banner

Features:

1. A banner fills up the entire unit using the set maximum sizes.
2. The width and height of banners is set using the `+flexibleSizeWithCGSize:` method.

Examples of displaying banners:



Enabling a banner

To enable the banner:

1. Add the import:

Swift

```
import YandexMobileAds
```

Objective-C

```
#import <YandexMobileAds/YandexMobileAds.h>
```

2. Create `@property`, where the link to the banner ad will be stored:

Swift

```
var adView: YMAAdView!
```

Objective-C

```
@property (nonatomic, strong) YMAAdView *adView;
```

3. Create a banner:

Sticky banner

To set the width of a banner, call the `+stickySizeWithContainerWidth:` method.

Swift

```
let adSize = YMAAdSize.stickySize(withContainerWidth: width)
let adView = YMAAdView(adUnitID: "", adSize: adSize)
adView.delegate = self
```

Objective-C

```
YMAAdSize adSize = [YMAAdSize stickySizeWithContainerWidth:width];
YMAAdView *adView = [[YMAAdView alloc] initWithAdUnitID:<AdUnitID> adSize:adSize];
adView.delegate = self;
```

Flex banner

To set the width and height of a banner, call the `+flexibleSizeWithCGSize:` method.

Swift

```
let adSize = YMAAdSize.flexibleSize(with: size)
let adView = YMAAdView(adUnitID: "", adSize: adSize)
adView.delegate = self
```

Objective-C

```
YMAAdSize adSize = [YMAAdSize flexibleSizeWithCGSize:size];
YMAAdView *adView = [[YMAAdView alloc] initWithAdUnitID:<AdUnitID> adSize:adSize];
adView.delegate = self;
```

Restriction: Banner size requirements when displaying video ads

Minimum size of a banner that supports video playback is 300x160 or 160x300.

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

In addition, `self` must conform to the `YMAAdViewDelegate` protocol. If the delegate implements the `-viewControllerForPresentingModalView` method, links open in the in-app browser. Otherwise, links open in the browser installed on the device.

To find out why ads aren't working correctly, use the `-adViewDidFailLoading:error:` method.

For error descriptions, see [YMAAdErrorCode](#).

4. Display a banner. There are two ways to place a banner:

- Using `autolayout` constraints.

Add the banner to `UIView`. Then add `autolayout` constraints so the banner is displayed in the desired location.

Swift

```
view.addSubview(adView)
adView.translatesAutoresizingMaskIntoConstraints = false
var adViewConstraints = [
    adView.leadingAnchor.constraint(equalTo: adView.superview!.leadingAnchor),
    adView.trailingAnchor.constraint(equalTo: adView.superview!.trailingAnchor)
]
let bottomDistance: CGFloat = 8
if #available(iOS 11.0, *) {
    adViewConstraints.append(
        adView.bottomAnchor.constraint(equalTo: view.safeAreaLayoutGuide.bottomAnchor, constant:
            bottomDistance)
    )
} else {
    adViewConstraints.append(
```

```
        adView.bottomAnchor.constraint(equalTo: view.bottomAnchor, constant: bottomDistance)
    )
}
NSLayoutConstraint.activate(adViewConstraints)
```

Objective-C

```
UIView *adView = self.adView;
[self.view addSubview:adView];
adView.translatesAutoresizingMaskIntoConstraints = NO;

NSMutableArray *adViewConstraints = [NSMutableArray arrayWithArray:@[
    [adView.leadingAnchor constraintEqualToAnchor:adView.superview.leadingAnchor],
    [adView.trailingAnchor constraintEqualToAnchor:adView.superview.trailingAnchor]
]];
int bottomDistance = 8;
if (@available(iOS 11.0, *)) {
    UILayoutGuide *guide = self.view.safeAreaLayoutGuide;
    [adViewConstraints addObject:[adView.bottomAnchor constraintEqualToAnchor:guide.bottomAnchor
                                constant:bottomDistance]];
} else {
    [adViewConstraints addObject:[adView.bottomAnchor
                                constraintEqualToAnchor:adView.superview.bottomAnchor
                                constant:bottomDistance]];
}
[NSLayoutConstraint activateConstraints:adViewConstraints];
```

- Using the following methods:

Swift

```
displayAtTop(in:)
displayAtBottom(in:)
```

Objective-C

```
-displayAtTopInView;;
-displayAtBottomInView;;
```

In both cases, banners are centered horizontally.

5. Load the banner. Optionally, you can use the [YMAAdRequest](#) class to transmit the data for targeting.

Swift

```
func loadAd(with request: YMAAdRequest?)
```

Objective-C

```
- (void)loadAdWithRequest:(YMAAdRequest *)request;
```

When the banner loads, the delegate is notified.

6. You can optionally enable logging by using the [+enableLogging](#) method. If an impression wasn't registered, a message appears in the console.
7. Optionally, you can set up notifications about the end of video playback in banner ads.

Usage example

Swift

```
// Getting an instance of YMAVideoController using VideoController.
let videoController = adView.videoController

// Setting up a delegate that implements the YMAVideoDelegate protocol.
videoController.delegate = self

// Implementing the YMAVideoDelegate protocol method.
// MARK: - YMAVideoDelegate
func videoControllerDidFinishPlayingVideo(_ videoController: YMAVideoController) {
    print("Video complete");
}
```

Objective-C

```
// Getting an instance of YMAVideoController using videoController.
YMAVideoController *videoController = self.adView.videoController;

// Setting up a delegate that implements the YMAVideoDelegate method.
videoController.delegate = self;

// Implementing the YMAVideoDelegate protocol method.
#pragma mark - YMAVideoDelegate
- (void)videoControllerDidFinishPlayingVideo:(YMAVideoController *)videoController
```

```
{
    NSLog(@"%@", @"Video complete");
}
```

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

```
let yandexAdUnitID = "demo-banner-yandex"
let adMobAdUnitID = "demo-banner-admob"
let mintegralAdUnitID = "demo-banner-mintegral"
let myTargetAdUnitID = "demo-banner-mytarget"
let startAppAdUnitID = "demo-banner-startapp"
```

See also

[Classes and protocols for working with banner ads](#)

[Tracking ad activity](#)

Related information

[Ad example](#)

Example of working with banners

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

Swift

```
override func viewDidLoad() {
    super.viewDidLoad()

    // Setting the ad width
    let adSize = YMAAdSize.stickySize(withContainerWidth: containerView())

    // Creating an adView instance
    adView = YMAAdView(adUnitID: "R-M-XXXX-YY", adSize: adSize)
    adView.delegate = self
    addAdView()

    // Loading the ad
    adView.loadAd()
}

private func addAdView() {
    view.addSubview(adView)
    adView.translatesAutoresizingMaskIntoConstraints = false
    var adViewConstraints = [
        adView.leadingAnchor.constraint(equalTo: view.superview!.leadingAnchor),
        adView.trailingAnchor.constraint(equalTo: view.superview!.trailingAnchor)
    ]
    let bottomDistance: CGFloat = 8
    if #available(iOS 11.0, *) {
        adViewConstraints.append(
            adView.bottomAnchor.constraint(equalTo: view.safeAreaLayoutGuide.bottomAnchor, constant:
            bottomDistance)
        )
    } else {
        adViewConstraints.append(
            adView.bottomAnchor.constraint(equalTo: view.bottomAnchor, constant: bottomDistance)
        )
    }
    NSLayoutConstraint.activate(adViewConstraints)
}

private func containerView() -> CGFloat {
    var containerViewWidth = view.frame.width
    if #available(iOS 11, *) {
        containerViewWidth = view.frame.inset(by: view.safeAreaInsets).width
    }
    return containerViewWidth
}
```

Objective-C

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    // Setting the ad width
    YMAAdSize *adSize =
        [YMAAdSize stickySizeWithContainerWidth:[self containerView]];

    // Creating an adView instance
    self.adView = [[YMAAdView alloc] initWithAdUnitID:@"R-M-XXXX-YY"
    adSize:adSize];
    self.adView.delegate = self;
    [self addAdView];

    // Loading the ad
    [self.adView loadAd];
}
```



```

}
- (void)addAdView
{
    UIView *adView = self.adView;
    [self.view addSubview:adView];
    adView.translatesAutoresizingMaskIntoConstraints = NO;

    NSMutableArray *adViewConstraints = [NSMutableArray arrayWithArray:@[
        [adView.leadingAnchor constraintEqualToAnchor:adView.superview.leadingAnchor],
        [adView.trailingAnchor constraintEqualToAnchor:adView.superview.trailingAnchor]
    ]];
    int bottomDistance = 8;
    if (@available(iOS 11.0, *)) {
        UILayoutGuide *guide = self.view.safeAreaLayoutGuide;
        [adViewConstraints addObject:[adView.bottomAnchor constraintEqualToAnchor:guide.bottomAnchor
                                     constant:bottomDistance]];
    } else {
        [adViewConstraints addObject:[adView.bottomAnchor constraintEqualToAnchor:adView.superview.bottomAnchor
                                     constant:bottomDistance]];
    }
    [NSLayoutConstraint activateConstraints:adViewConstraints];
}
- (CGFloat)containerWidth
{
    CGFloat containerWidth = self.view.frame.size.width;
    if (@available(iOS 11.0, *)) {
        containerWidth =
            UIEdgeInsetsInsetRect(self.view.frame, self.view.safeAreaInsets).size.width;
    }
    return containerWidth;
}

```

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:

1. Add the import:

Swift

```
import YandexMobileAds
```

Objective-C

```
#import <YandexMobileAds/YandexMobileAds.h>
```

2. Create @property where the link to the banner ad will be stored:

Swift

```
var interstitialAd: YMAInterstitialAd!
```

Objective-C

```
@property (nonatomic, strong) YMAInterstitialAd *interstitialAd;
```

3. Perform initialization and pre-loading of the ad. This must be performed after displaying the app interface.

Optionally, you can use the [YMAAdRequest](#) class to transmit the data for targeting. For an example of using the class, see [Interstitial ads](#).

Swift

```
interstitialAd = YMAInterstitialAd(adUnitID: "<AdUnitID>")
interstitialAd.delegate = self
interstitialAd.load()
```

Objective-C

```
self.interstitialAd = [[YMAInterstitialAd alloc] initWithAdUnitID:<your unique AdUnitId>];
```

```
self.interstitialAd.delegate = self;
[self.interstitialAd load];
```

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

In addition, self must conform to the [YMAInterstitialAdDelegate](#) protocol. The ad must be pre-loaded in the same orientation as it will be shown (otherwise, the ad won't be shown because the banner size doesn't match the screen size).

4. Start displaying ads by using this method:

Swift

```
func interstitialAdDidLoad(_ interstitialAd: YMAInterstitialAd) {
    interstitialAd.present(from: self)
}
```

Objective-C

```
- (void)interstitialAdDidLoad:(YMAInterstitialAd *)interstitialAd
{
    [interstitialAd presentFromViewController:self];
}
```

5. You can optionally enable logging by using the [+enableLogging](#) method. If an impression wasn't registered, a message appears in the console.

To find out why ads aren't working correctly, use the methods

Swift

```
func interstitialAdDidFail(toLoad interstitialAd: YMAInterstitialAd, error: Error)
func interstitialAdDidFail(toPresent interstitialAd: YMAInterstitialAd, error: Error)
```

Objective-C

```
- (void)interstitialAdDidFailToLoad:(YMAInterstitialAd *)interstitialAd error:(NSError *)error;
- (void)interstitialAdDidFailToPresent:(YMAInterstitialAd *)interstitialAd error:(NSError *)error;
```

For error descriptions, see [YMAAdErrorCode](#).

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

```
let yandexAdUnitID = "demo-interstitial-yandex"
let adMobAdUnitID = "demo-interstitial-admob"
let appLovinAdUnitID = "demo-interstitial-applovin"
let ironSourceAdUnitID = "demo-interstitial-ironsource"
let mintegralAdUnitID = "demo-interstitial-mintegral"
let myTargetAdUnitID = "demo-interstitial-mytarg"
let startAppAdUnitID = "demo-interstitial-startapp"
let unityAdsAdUnitID = "demo-interstitial-unityads"
```

See also

[Classes and protocols for working with interstitial ads](#)

[Tracking ad activity](#)

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:

1. Add the import:

Swift

```
import YandexMobileAds
```

Objective-C

```
#import <YandexMobileAds/YandexMobileAds.h>
```

2. Create @property for storing the link to the ad:

Swift

```
var rewardedAd: YMARewardedAd!
```

Objective-C

```
@property (nonatomic, strong) YMARewardedAd *rewardedAd;
```

3. Initialize the ad:

Swift

```
rewardedAd = YMARewardedAd(adUnitID: "<AdUnitID>")
rewardedAd.delegate = self
```

Objective-C

```
self.rewardedAd = [[YMARewardedAd alloc] initWithAdUnitID:<your unique AdUnitID>];
self.rewardedAd.delegate = self;
```

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

In addition, self must conform to the [YMARewardedAdDelegate](#) protocol.

4. Load an ad.

Swift

```
rewardedAd.load()
```

Objective-C

```
[self.rewardedAd load];
```

Optionally, you can use the [YMAAdRequest](#) class to transmit the data for targeting. For an example of using the class, see [Interstitial ads](#).

The ad must be pre-loaded in the same orientation as it will be shown (otherwise, the ad will not be shown because the banner size won't match the screen size).

5. Start displaying ads by using this method:

Swift

```
func rewardedAdDidLoad(_ rewardedAd: YMARewardedAd) {
    rewardedAd.present(from: self)
}
```

Objective-C

```
- (void)rewardedAdDidLoad:(YMARewardedAd *)rewardedAd
{
    [rewardedAd presentFromViewController:self];
}
```

6. If you are using the “client-side reward” mechanism, implement the `- rewardedAd:didReward:` delegate 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.

7. You can optionally enable logging by using the [+enableLogging](#) method. If an impression wasn't registered, a message appears in the console.

To find out why ads aren't working correctly, use the methods

Swift

```
func interstitialAdDidFail(toLoad interstitialAd: YMAInterstitialAd, error: Error)
func interstitialAdDidFail(toPresent interstitialAd: YMAInterstitialAd, error: Error)
```

Objective-C

```
- (void)interstitialAdDidFailToLoad:(YMAInterstitialAd *)interstitialAd error:(NSError *)error;
- (void)interstitialAdDidFailToPresent:(YMAInterstitialAd *)interstitialAd error:(NSError *)error;
```

Swift

```
func rewardedAdDidFail(toLoad rewardedAd: YMARewardedAd, error: Error)
func rewardedAdDidFail(toPresent rewardedAd: YMARewardedAd, error: Error)
```

Objective-C

```
- (void)rewardedAdDidFailToLoad:(YMARewardedAd *)rewardedAd error:(NSError *)error;
- (void)rewardedAdDidFailToPresent:(YMARewardedAd *)rewardedAd error:(NSError *)error;
```

For error descriptions, see [YMAAdErrorCode](#).

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

```
let yandexAdUnitID = "demo-rewarded-yandex"
let adMobAdUnitID = "demo-rewarded-admob"
let appLovinAdUnitID = "demo-rewarded-applovin"
let ironSourceAdUnitID = "demo-rewarded-ironsource"
let mintegralAdUnitID = "demo-rewarded-mintegral"
let myTargetAdUnitID = "demo-rewarded-mytarget"
let startAppAdUnitID = "demo-rewarded-startapp"
let unityAdsAdUnitID = "demo-rewarded-unityads"
```

See also

[Classes and protocols for working with rewarded video ads](#)

Related information

[Ad example](#)

Native ads

Integration



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.

The Yandex Mobile Ads SDK supports several types of advertising: App Install, Content, Image.

To enable advertising:

1. Read the [advertising requirements](#).
2. [Load an ad](#).
3. [Configure the ad's design](#).

See also[Classes and protocols for working with native ads](#)[Tracking ad activity](#)**Related information**[YMANativeErrorCode](#)[Ad example](#)**Loading ads****Warning:**

This is an archived version of the documentation. Actual documentation for all platforms can be found [here](#).

Loading ads

1. Create an instance of the [YMANativeAdLoader](#) class to get native ads.
2. Create a configuration for the nativeAdRequestConfiguration request using the [YMANativeAdRequestConfiguration](#) 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. Set a delegate for retrieving an ad that implements the [YMANativeAdLoaderDelegate](#) protocol:
4. To track the ad loading process, implement the [YMANativeAdLoaderDelegate](#) protocol methods: `-nativeAdLoader:didFailLoadingWithError:` and `-nativeAdLoader:didLoadAd:`.
5. To load the ad, send the `loadAdWithRequestConfiguration:` message to the loader.

Swift

```
adLoader.loadAd(with: requestConfiguration)
```

Objective-C

```
[self.adLoader loadAdWithRequestConfiguration:requestConfiguration];
```

6. If the ad loaded, the following method is called:

Swift

```
func nativeAdLoader(_ loader: YMANativeAdLoader, didLoad ad: YMANativeAd)
```

Objective-C

```
- (void)nativeAdLoader:(YMANativeAdLoader *)loader didLoadAd:(id<YMANativeAd>)ad
```

7. If the ad didn't load, the following method is called:

Swift

```
func nativeAdLoader(_ loader: YMANativeAdLoader, didFailLoadingWithError error: Error)
```

Objective-C

```
- (void)nativeAdLoader:(YMANativeAdLoader *)loader didFailLoadingWithError:(NSError *)error
```

For more information about possible errors, see [YMANativeErrorCode](#).

Examples with a demo AdUnitId

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

```
let yandexAdUnitID = "demo-native-content-yandex"
let adMobAdUnitID = "demo-native-admob"
let myTargetAdUnitID = "demo-native-mytarget"
```

Loading multiple ads

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

1. Create an instance of the [YMANativeBulkAdLoader](#) class to get native ads.
2. Create a configuration for the nativeAdRequestConfiguration request using the [YMANativeAdRequestConfiguration](#) class. As the request parameters, you can use the ad unit ID, method for loading images, age, gender characteristics, and other data that might improve the quality of ad selection.
3. Set a delegate for retrieving an ad that implements the [YMANativeBulkAdLoaderDelegate](#) protocol.
4. To track the ad loading process, implement the [YMANativeBulkAdLoaderDelegate](#) protocol methods: - [nativeBulkAdLoader:didLoadAds:](#) and -[nativeBulkAdLoader:didFailLoadingWithError:](#).
5. Send to the loader the request configuration and count of ads requested (the adsCount parameter).

Swift

```
adLoader.loadAds(with: requestConfiguration, adsCount: adsCount)
```

Objective-C

```
[self.adLoader loadAdsWithRequestConfiguration:requestConfiguration adsCount:adsCount];
```

Swift

```
// Creating a request configuration
let requestConfiguration = YMAMutableNativeAdRequestConfiguration(adUnitID: AdUnitID)

// Creating a loader
adLoader = YMANativeBulkAdLoader()
adLoader.delegate = self

// Passing the request configuration and the number of requested ads to the loader
adLoader.loadAds(with: requestConfiguration, adsCount: adsCount)

// Implementing delegate methods

func nativeBulkAdLoader(_ nativeBulkAdLoader: YMANativeBulkAdLoader, didLoad ads: [YMANativeAd]) {
    ...
    // Working with each id<YMANativeAd> object separately. For more information, see "Customizing the ad
    design".
}
```

Objective-C

```
// Creating a request configuration
YMANativeAdRequestConfiguration *requestConfiguration =
    [[YMANativeAdRequestConfiguration alloc] initWithAdUnitID:AdUnitID];

// Creating a loader self.adLoader = [[YMANativeBulkAdLoader alloc] init];
self.adLoader.delegate = self;

// Passing the request configuration and count of requested ads to the loader
[self.adLoader loadAdsWithRequestConfiguration:requestConfiguration adsCount:adsCount];

// Implementing delegate methods - (void)nativeBulkAdLoader:(YMANativeBulkAdLoader *)nativeBulkAdLoader
didLoadAds:(NSArray<idYMANativeAd> *)ads
{
    ...
    // Working with each id<YMANativeAd> object separately. For more information, see "Customizing the ad
    design".
}
```

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 adsCount NativeAd objects. You can render all the received ad objects separately from each other using the above methods for laying out native ads.

Ways to load images**Automatic loading**

If the app simultaneously stores links to just one or a small number of ads, we recommend using automatic loading.

When creating the configuration, set [shouldLoadImagesAutomatically](#) to YES:

Swift

```
let requestConfiguration = YMAMutableNativeAdRequestConfiguration(adUnitID: AdUnitID)
requestConfiguration.shouldLoadImagesAutomatically = true
```

Objective-C

```
YMAMutableNativeAdRequestConfiguration *requestConfiguration =
    [[YMAMutableNativeAdRequestConfiguration alloc] initWithAdUnitID:AdUnitID];
requestConfiguration.shouldLoadImagesAutomatically = YES;
```

The resulting ad will have the images ready. They are stored in the device memory until the ad is destroyed.

Manual loading

If the app simultaneously stores links to a large number of ads, we recommend using manual image loading.

When creating the configuration, set [shouldLoadImagesAutomatically](#) to NO:

Swift

```
let requestConfiguration = YMAMutableNativeAdRequestConfiguration(adUnitID: AdUnitID)
requestConfiguration.shouldLoadImagesAutomatically = false
```

Objective-C

```
YMAMutableNativeAdRequestConfiguration *requestConfiguration =
    [[YMAMutableNativeAdRequestConfiguration alloc] initWithAdUnitID:AdUnitID];
requestConfiguration.shouldLoadImagesAutomatically = NO;
```

The resulting ad will only have the image sizes. To load the images themselves, call the [loadImages](#) method on the received ad.

**Attention:**

All images are cached, but they can be deleted at any time, so you need to call the [loadImages\(\)](#) method before every ad impression.

Swift

```
func showAd() {
    // Show ad: custom native view or template
    view.addSubview(adView)
    adView.ad?.loadImages()
}
```

Objective-C

```
- (void)showAd
{
    // Show ad: custom native view or template
    [self.view addSubview:self.adView];
    [self.adView.ad loadImages];
}
```

Notifications about image loading

Restriction: You can only get notifications when loading images manually.

Enabling notifications

To enable notifications that are sent when an image is loaded, call the [-addImageLoadingObserver:](#) method.

Swift

```
ad?.add(self)

func nativeAdDidFinishLoadingImages(_ ad: YMANativeAd) {
    print("Finished loading images")
}
```

Objective-C

```
[ad addImageLoadingObserver:self];

- (void)nativeAdDidFinishLoadingImages:(id<YMANativeAd>)ad
{
    NSLog(@"Finished loading images");
}
```

Disabling notifications

To disable notifications that are sent when an image is loaded, call the [-removeImageLoadingObserver:](#) method.

Swift

```
ad?.remove(self)
```

Objective-C

```
[ad removeImageLoadingObserver:self];
```

Configuring the ad design

**Warning:**

This is an archived version of the documentation. Actual documentation for all platforms can be found [here](#).

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

1. [Layout using a template.](#)

The easiest way to work with native ads is to use a standard template for layout. The template already has the set of required assets, and defines their arrangement relative to each other. The template works with any supported type of ad.

2. [Layout without a template.](#)

Create a layout without using a template when the template settings don't give you enough configuration options to make the ad look organic and match the design of the app.

Layout using a template

**Warning:**

This is an archived version of the documentation. Actual documentation for all platforms can be found [here](#).

To customize the layout and design, you can use a standard template or create your own design based on a standard template.

Using the standard layout template

Note:

After setting up the layout, set the [position and ad size](#) relative to the device screen.

1. Create an instance of the [YMANativeBannerView](#) class and set the loaded ad for it:

Swift

```
let bannerView = YMANativeBannerView()  
bannerView.ad = ad  
view.addSubview(bannerView)
```

Objective-C

```
YMANativeBannerView *bannerView = [[YMANativeBannerView alloc] init];  
bannerView.ad = ad;  
[self.view addSubview:bannerView];
```


- To receive notifications about user interactions with the ad (opening the ad or exiting the app), assign it the `YMANativeAdDelegate`, which implements the methods:

- `-nativeAd:didDismissScreen:`
- `-nativeAd:willPresentScreen:`
- `-nativeAdWillLeaveApplication:`
- `-viewControllerForPresentingModalView`

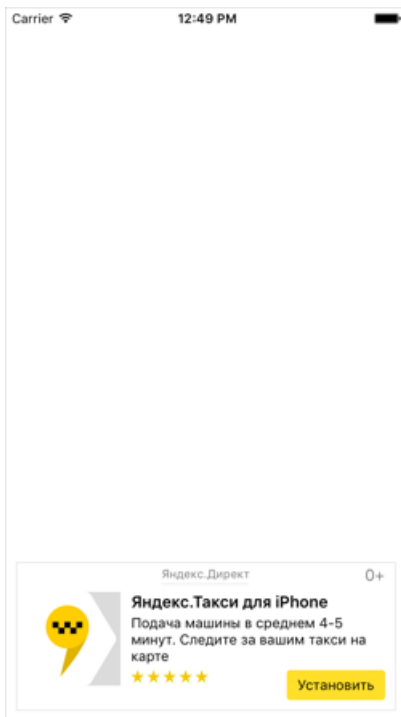
Swift

```
ad.delegate = self
```

Objective-C

```
ad.delegate = self;
```

- Example of using the standard layout template:



Note: If the standard layout doesn't fit your app, you can change it. For more information, see [Creating your own template-based layout](#).

Creating your own template-based layout

Note:

After setting up the layout, set the [position and ad size](#) relative to the device screen.

- Create an instance of the `YMANativeBannerView` class and set the loaded ad for it:

Swift

```
let bannerView = YMANativeBannerView()
bannerView.ad = ad
view.addSubview(bannerView)
```

Objective-C

```
YMANativeBannerView *bannerView = [[YMANativeBannerView alloc] init];
bannerView.ad = ad;
[self.view addSubview:bannerView];
```

- To receive notifications about user interactions with the ad (opening the ad or exiting the app), assign it the [YMANativeAdDelegate](#), which implements the methods:

- [-nativeAd:didDismissScreen:](#)
- [-nativeAd:willPresentScreen:](#)
- [-nativeAdWillLeaveApplication:](#)
- [-viewControllerForPresentingModalView](#)

Swift

```
ad.delegate = self
```

Objective-C

```
ad.delegate = self;
```

- Request the settings for the standard layout template:

Swift

```
let appearance = YMAMutableNativeTemplateAppearance.default()
```

Objective-C

```
YMAMutableNativeTemplateAppearance *appearance = [[YMANativeTemplateAppearance defaultAppearance] mutableCopy];
```

- Set the preferred settings.
- To apply the settings to the template, call the [-applyAppearance:](#) method:

Swift

```
bannerView.apply(appearance)
```

Objective-C

```
[bannerView applyAppearance:appearance];
```

Example of layout configuration

Swift

```
// Defining a custom color scheme.
let orangeColor = UIColor(red: 1, green: 176.0/255, blue: 32.0/255, alpha: 1)
let blueColor = UIColor(red: 0, green: 170.0/255, blue: 1, alpha: 1)

// Creating a copy with the settings of the native design template.
let appearance = YMAMutableNativeTemplateAppearance.default()

// Starting to change the native template settings.

// Setting the color for the ad frame.
appearance.borderColor = orangeColor

// Creating a copy with rating settings.
let ratingAppearance = appearance.ratingAppearance?.mutableCopy() as? YMAMutableRatingAppearance

// Setting the color for filled stars in the rating.
ratingAppearance?.filledStarColor = orangeColor
appearance.ratingAppearance = ratingAppearance

// Setting the font color and size for the action button label.
let callToActionTextAppearance = YMALabelAppearance(font: .systemFont(ofSize: 14), textColor: blueColor)

// Setting the button color for the normal state and clicked state, along with the color and thickness of the button border.
let callToActionAppearance = YMAButtonAppearance(
    textAppearance: callToActionTextAppearance,
    normalColor: .clear,
    highlightedColor: .gray,
    borderColor: blueColor,
    borderWidth: 1
)
appearance.callToActionAppearance = callToActionAppearance

// Setting the font size and color for the age restriction label.
appearance.ageAppearance = YMALabelAppearance(font: .systemFont(ofSize: 12), textColor: .gray)

// Setting the font size and color for the ad title.
appearance.titleAppearance = YMALabelAppearance(font: .systemFont(ofSize: 14), textColor: .black)
```

```
// Setting the font size and color for the main ad text.
appearance.bodyAppearance = YMALabelAppearance(font: .systemFont(ofSize: 12), textColor: .gray)

// Setting the image width and the sizing rule.
let imageConstraint = YMASizeConstraint(type: .fixed, value: 60)

// Applying the settings to the image.
appearance.imageAppearance = YMAImageAppearance(widthConstraint: imageConstraint)
```

Objective-C

```
// Define custom colors.
UIColor *orangeColor =
    [UIColor colorWithRed:255.f / 255.f green:176.f / 255.f blue:32.f / 255.f alpha:1.f];
UIColor *blueColor =
    [UIColor colorWithRed:0.f / 255.f green:170.f / 255.f blue:255.f / 255.f alpha:1.f];

// Creating a copy with the settings of the standard layout template.
YMAMutableNativeTemplateAppearance *appearance = [[YMANativeTemplateAppearance defaultAppearance] mutableCopy];

// Starting to change the standard template settings.

// Setting the color for the ad frame.
appearance.borderColor = orangeColor;

// Creating a copy with rating settings.
YMAMutableRatingAppearance *ratingAppearance = [appearance.ratingAppearance mutableCopy];

// Setting the color for filled stars in the rating.
ratingAppearance.filledStarColor = orangeColor;
appearance.ratingAppearance = ratingAppearance;

// Setting the font color and size for the action button label.
YMALabelAppearance *callToActionTextAppearance =
    [YMALabelAppearance appearanceWithFont:[UIFont systemFontOfSize:14.f]
                                     textColor:blueColor];

// Setting the button color for the normal state and the clicked state, along with the color and thickness of
// the button border.
YMAButtonAppearance *callToActionAppearance =
    [YMAButtonAppearance appearanceWithTextAppearance:callToActionTextAppearance
                                normalColor:[UIColor clearColor]
                                highlightedColor:[UIColor grayColor]
                                borderColor:blueColor
                                borderWidth:1.f];
appearance.callToActionAppearance = callToActionAppearance;

// Setting the font size and color for the age restriction label.
appearance.ageAppearance =
    [YMALabelAppearance appearanceWithFont:[UIFont systemFontOfSize:12.f]
                                     textColor:[UIColor grayColor]];

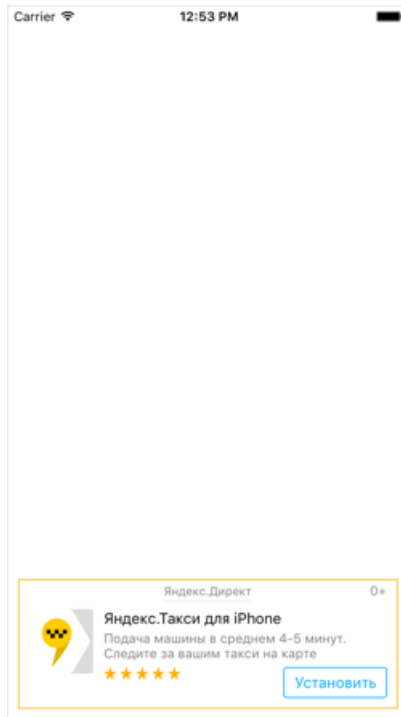
// Setting the font size and color for the ad title.
appearance.titleAppearance =
    [YMALabelAppearance appearanceWithFont:[UIFont systemFontOfSize:14.f]
                                     textColor:[UIColor blackColor]];

// Setting the font size and color for the main text of the ad.
appearance.bodyAppearance =
    [YMALabelAppearance appearanceWithFont:[UIFont systemFontOfSize:12.f]
                                     textColor:[UIColor grayColor]];

// Setting the image width and sizing constraint.
YMASizeConstraint *imageConstraint =
    [YMASizeConstraint constraintWithType:YMASizeConstraintTypeFixed value:60.f];

// Applying the settings to the image.
appearance.imageAppearance =
    [YMAImageAppearance appearanceWithWidthConstraint:imageConstraint];
```

We get our custom design based on the template:



Setting the position and the ad size

Restriction: mediaView size requirements when displaying video ads

Minimum size of an instance of the [YMANativeMediaView](#) class, which supports video playback: 300x160 or 160x300.

To support video playback in native ad templates, we recommend setting the width for [YMANativeBannerView](#) to at least 300. The correct height for mediaView will be calculated automatically based on the width to height ratio.

There are two ways to control size and location:

1. Using the system's AutoLayout mechanism.

Note:

When using AutoLayout, set constraint for the UIView width. The height is determined automatically based on the specified width.

2. By manually setting all the sizes.

Setting sizes manually

Note:

We don't recommend setting an object width greater than 420 logical pixels.

Set the width and height for the [YMANativeBannerView](#) object. The height is determined using the [+heightWithAd:width:appearance:](#) method which must be passed the ad, the width, and the [YMANativeTemplateAppearance](#) object (used for setting the ad's appearance).

Swift

```
// Setting the margins on the left and right relative to the screen.
let inset: CGFloat = 50

// Setting the width.
let width = view.frame.width - 2 * inset

// Setting the height.
let height = YMANativeBannerView.height(with: ad, width: width, appearance: nil)

// Setting the vertical position.
let y = view.frame.maxY - height - inset

// Setting the coordinates and sizes for the frame.
bannerView.frame = CGRect(x: inset, y: y, width: width, height: height)
```

Objective-C

```
// Set the margins on the left and right relative to the screen.
CGFloat inset = 50.f;

// Setting the width.
CGFloat width = CGRectGetWidth(self.view.frame) - 2 * inset;

// Setting the height.
CGFloat height = [YMANativeBannerView heightForAd:ad width:width appearance:nil];

// Setting the vertical position.
CGFloat y = CGRectGetMaxY(self.view.frame) - height - inset;

// Setting the coordinates and sizes for the frame.
bannerView.frame = CGRectMake(inset, y, width, height);
```

Layout without a template**Warning:**

This is an archived version of the documentation. Actual documentation for all platforms can be found [here](#).

To display native ads, `YMANativeAd` must be bound to a specific `View`. This `View` must be defined as the root element for the subview set that the data contained in the native ad will be bound to.

Restriction: mediaView size requirements when displaying video ads

Minimum size of an instance of the `YMANativeMediaView` class, which supports video playback: 300x160 or 160x300.

To support video playback in native ads, we recommend setting the width for `mediaView` to at least 300. To calculate the appropriate `mediaView` height value, use the `aspectRatio` property value.

Layout of native ads

1. Review the [list](#) of required and optional subviews for each type of `View`.
2. Set the values for all required subviews. You can do this using `nib`, or directly in the code. Create an instance of `YMANativeAdView` and define the subview values in `initWithFrame`:

Example of registering a set of subviews:**Swift**

```
override init(frame: CGRect) {
    super.init(frame: frame)
    let titleLabel = createLabel()
    let bodyLabel = createLabel()
    let ageLabel = createLabel()
    let warningLabel = createLabel()
    let sponsoredByLabel = createLabel()
    let priceLabel = createLabel()
    let starRatingView = createStarRatingView()
    let button = createButton()
    let iconImageView = createIconAssetImageView()
    let mediaView = createMediaAssetView()
    addSubview(titleLabel)
    addSubview(bodyLabel)
    addSubview(ageLabel)
    addSubview(warningLabel)
    addSubview(sponsoredByLabel)
    addSubview(priceLabel)
    addSubview(starRatingView)
    addSubview(button)
    addSubview(iconImageView)
    addSubview(mediaView)
    self.titleLabel = titleLabel
    self.bodyLabel = bodyLabel
    self.ageLabel = ageLabel
    self.warningLabel = warningLabel
    self.sponsoredLabel = sponsoredByLabel
    self.callToActionButton = callToActionButton
    self.priceLabel = priceLabel
    self.ratingView = ratingView
    self.iconImageView = iconImageView
    self.mediaView = mediaView
}
```

Objective-C

```
- (instancetype)initWithFrame:(CGRect)frame
{
```

```

self = [super initWithFrame:frame];
if (self != nil) {
    UILabel *titleLabel = [self label];
    UILabel *bodyLabel = [self label];
    UILabel *ageLabel = [self secondaryLabel];
    UILabel *warningLabel = [self label];
    UILabel *sponsoredByLabel = [self label];
    UILabel *priceLabel = [self label];
    StarRatingView *ratingView = [self starRatingView];
    UIButton *callToActionButton = [self button];
    UIImageView *iconImageView = [self iconAssetImageView];
    YMANativeMediaView *mediaView = [self mediaAssetView];
    [self addSubview:titleLabel];
    [self addSubview:bodyLabel];
    [self addSubview:ageLabel];
    [self addSubview:warningLabel];
    [self addSubview:sponsoredByLabel];
    [self addSubview:callToActionButton];
    [self addSubview:priceLabel];
    [self addSubview:ratingView];
    [self addSubview:iconImageView];
    [self addSubview:mediaView];
    self.titleLabel = titleLabel;
    self.bodyLabel = bodyLabel;
    self.ageLabel = ageLabel;
    self.warningLabel = warningLabel;
    self.sponsoredLabel = sponsoredByLabel;
    self.callToActionButton = callToActionButton;
    self.priceLabel = priceLabel;
    self.ratingView = ratingView;
    self.iconImageView = iconImageView;
    self.mediaView = mediaView;
}
return self;
}

```

- To receive notifications about user interactions with the ad (opening or closing the ad or exiting the app), assign it the `YMANativeAdDelegate`, which implements the methods:
 - `-closeNativeAd;`
 - `-nativeAd:didDismissScreen:`
 - `-nativeAd:willPresentScreen:`
 - `-nativeAdWillLeaveApplication:`
 - `-viewControllerForPresentingModalView`
- Request the values for native ad assets using the `-adAssets` method. This will help you calculate the position and sizes of these assets in advance.

Swift

```
let assets = ad.adAssets()
```

Objective-C

```
YMANativeAdAssets *assets = [ad adAssets];
```

Example of getting the image size and aspect ratio of the ad title text and media

Swift

```

let image = assets.image
let title = assets.title
let media = assets.media
print("Image size: \(image?.size ?? .zero)")
print("Title: \(title ?? "")")
print(String(format: "Media aspect ratio: %.2f", media?.aspectRatio ?? 0))

```

Objective-C

```

YMANativeAdImage *image = assets.image;
NSString *title = assets.title;
YMANativeAdMedia *media = assets.media;
NSLog(@"Image size: %@", NSStringFromCGSize(image.size));
NSLog(@"Title: %@", title);
NSLog(@"Media aspect ratio: %.2f", media.aspectRatio);

```

- Call the `-bindWithAdView:error:` method to bind the content to the native ad object.

Swift

```

// ...
adView = YMANativeAdView(frame: frame)
//configure content ad view
// ...

```

```
func nativeAdLoader(_ loader: YMANativeAdLoader, didLoad ad: YMANativeAd) {
    ad.delegate = self
    do {
        try ad.bind(with: contentAdView)
    } catch {
        print("Error: \(error)")
    }
}
```

Objective-C

```
// ...
self.adView = [[YMANativeAdView alloc] initWithFrame:frame];
//configure content ad view
// ...
- (void)nativeAdLoader:(YMANativeAdLoader *)loader didLoadAd:(id<YMANativeAd>)ad
{
    ad.delegate = self;
    NSError *error = nil;
    BOOL result = [ad bindWithAdView:self.contentAdView error:error];
    if (error != nil) {
        NSLog(@"Error: %@", error);
    }
}
```

Note:

If a required element of a native ad has the corresponding `YMANativeAdView` property set to `nil`, binding doesn't occur and the ad isn't displayed. See details in the error.

Debugging



Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found [here](#).

Logging

You can optionally enable logging by using the `+enableLogging` method. If an impression wasn't registered, a message appears in the console.

Invalid integration indicator for native ads

If an error was made when integrating native ads, an indicator appears on top of the ad in simulator mode. Click on the indicator to see a message with debugging information to help you understand the reason for the error. Click the indicator again to hide the message.



Attention:

By default, the indicator is only shown in simulator mode (device type `YMADeviceTypeSimulator`). You can view device types in [YMADeviceType](#).

To enable the indicator for real devices as well, pass the value `YMADeviceTypeHardware | YMADeviceTypeSimulator` in the [enableVisibilityErrorIndicatorForDeviceType](#) method:

Swift

```
YMAMobileAds.enableVisibilityErrorIndicator(for: [.hardware, .simulator])
```

Objective-C

```
[YMAMobileAds enableVisibilityErrorIndicatorForDeviceType:YMADeviceTypeHardware | YMADeviceTypeSimulator]
```

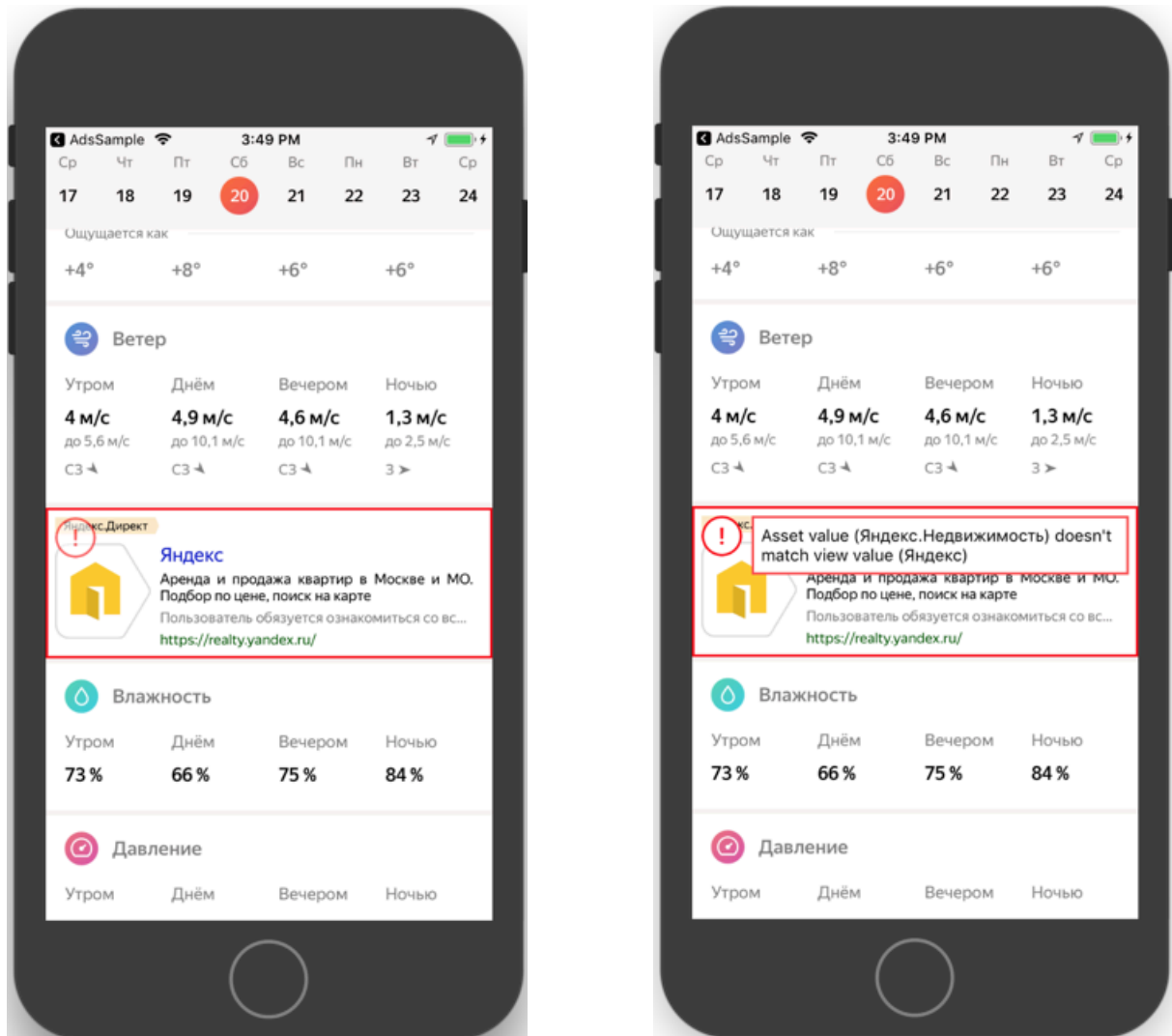
To turn off the indicator, pass the value `YMADeviceTypeNone` in the [enableVisibilityErrorIndicatorForDeviceType](#) method:

Swift

```
YMAMobileAds.enableVisibilityErrorIndicator(for: [])
```

Objective-C

```
[YMAMobileAds enableVisibilityErrorIndicatorForDeviceType:YMADeviceTypeNone]
```



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	Adapter version	Ad network SDK version	Banner Ad	Interstitial Ad	Rewarded Ad	Native Ad
AdMob	10.7.0.0	10.7.0				
myTarget	5.17.5.1	5.17.5				

Ad network	Adapter version	Ad network SDK version	Banner Ad	Interstitial Ad	Rewarded Ad	Native Ad
Start.io	4.7.3.0	4.7.3	✓	✓	✓	✗
UnityAds	4.8.0.0	4.8.0	✓	✓	✓	✗
AppLovin	11.10.1.1	11.10.1	✗	✓	✓	✗
IronSource	7.3.0.3	7.3.0	✗	✓	✓	✗
Mintegral	7.3.9.0	7.3.9	✓	✓	✓	✗
AdColony	4.9.0.4	4.9.0	✓	✓	✓	✗
Chartboost	9.3.1.0	9.3.1	✓	✓	✓	✗

Enabling AdMob



Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found [here](#).

Note:

To track app installations in mobile mediation, you need to configure `Info.plist` for the adapter. Follow the ad network [documentation](#).

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.9.0.

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

GoogleMobileAdsSDK

Minimum supported version: 10.7.0.

Maximum supported version: 11.0.0 (up to but not including).

iOS

Minimum supported version: 12.0.

Supported ad formats

- [Banner ads](#)
- [Interstitial ads](#)
- [Rewarded ads](#)
- [Native ads](#)

Integration

To work with [AdMob](#) you must connect the `AdMobYandexMobileAdsAdapters` library.

The `AdMobYandexMobileAdsAdapters` library has been adapted to work with the CocoaPods dependency management system and supports a static integration method.

1. [Set up mediation](#) in the Yandex Partner interface and YAN interface.

2. To connect the library, add the dependency to the project's Podfile:

```
pod 'AdMobYandexMobileAdsAdapters', '10.7.0.0'
```

3. To your app's Info.plist file, add the GADApplicationIdentifier key with your AdMob ID (learn more about how to [find out the AdMob ID](#)).

```
<key>GADApplicationIdentifier</key>  
<string>ca-app-pub-xxxxxxxxxxxxxxxx~yyyyyyyyyy</string>
```

Integration examples

- [Ad examples](#)

Enabling myTarget



Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found [here](#).

Note:

To track app installations in mobile mediation, you need to configure Info.plist for the adapter. Follow the ad network [documentation](#).

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.9.0.

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

myTargetSDK

Minimum supported version: 5.17.5.

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

iOS

Minimum supported version: 12.0.

Supported ad formats

- [Banner ads](#)
- [Interstitial ads](#)
- [Rewarded ads](#)
- [Native ads](#)

Integration

To work with [myTarget](#), you need to connect the MyTargetYandexMobileAdsAdapters library.

The MyTargetYandexMobileAdsAdapters library has been adapted to work with the CocoaPods dependency management system and supports a static integration method.

1. [Set up mediation](#) in the Yandex Partner interface and YAN interface.
2. To connect the library, add the dependency to the project's Podfile:

```
pod 'MyTargetYandexMobileAdsAdapters', '5.17.5.1'
```

Integration examples

- [Ad example](#)

Enabling Start.io

**Warning:**

This is an archived version of the documentation. Actual documentation for all platforms can be found [here](#).

Note:

To track app installations in mobile mediation, you need to configure `Info.plist` for the adapter. Follow the ad network [documentation](#).

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.2.0.

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

Start.io SDK

Minimum supported version: 4.7.3.

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

iOS

Minimum supported version: 12.0.

Supported ad formats

- [Banner ads](#)
- [Interstitial ads](#)
- [Rewarded ads](#)

Integration

To work with [Start.io](#), you need to connect the `StartAppYandexMobileAdsAdapters` library.

The `StartAppYandexMobileAdsAdapter` library has been adapted to work with the CocoaPods dependency management system and supports a static integration method.

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.

1. [Set up mediation](#) in the Yandex Partner interface and YAN interface.
2. To connect the library, add the dependency to the project's Podfile:

```
pod 'StartAppYandexMobileAdsAdapters', '4.7.3.0'
```

Integration examples

- [Ad example](#)

Enabling UnityAds

**Warning:**

This is an archived version of the documentation. Actual documentation for all platforms can be found [here](#).

Note:

To track app installations in mobile mediation, you need to configure `Info.plist` for the adapter. Follow the ad network [documentation](#).

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.9.0.

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

UnityAdsSDK

Minimum supported version: 4.8.0.

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

iOS

Minimum supported version: 12.0.

Supported ad formats

- [Interstitial ads](#)
- [Rewarded ads](#)
- [Banner ads](#)

Integration

To work with [UnityAds](#), you need to connect the `UnityYandexMobileAdsAdapters` library.

The `UnityYandexMobileAdsAdapters` library has been adapted to work with the CocoaPods dependency management system and supports a static integration method.

1. [Set up mediation](#) in the Yandex Partner interface and YAN interface.
2. To connect the library, add the dependency to the project's Podfile:

```
pod 'UnityAdsYandexMobileAdsAdapters', '4.8.0.0'
```

Integration examples

- [Ad examples](#)

Enabling AppLovin



Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found [here](#).

Note:

To track app installations in mobile mediation, you need to configure `Info.plist` for the adapter. Follow the ad network [documentation](#).

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.9.0.

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).

iOS

Minimum supported version: 12.0.

Supported ad formats

- [Interstitial ads](#)
- [Rewarded ads](#)

Integration

To work with [AppLovin](#), you need to connect the AppLovinYandexMobileAdsAdapters library.

The AppLovinYandexMobileAdsAdapters library has been adapted to work with the CocoaPods dependency management system and supports a static integration method.

1. [Set up mediation](#) in the Yandex Partner interface and YAN interface.
2. To connect the library, add the dependency to the project's Podfile:

```
pod 'AppLovinYandexMobileAdsAdapters', '11.10.1.1'
```

Integration examples

- [Ad example](#)

Enabling IronSource



Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found [here](#).

Note:

To track app installations in mobile mediation, you need to configure Info.plist for the adapter. Follow the ad network [documentation](#).

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.9.0.

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

IronSourceSDK

Minimum supported version: 7.3.0.

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

iOS

Minimum supported version: 12.0.

Supported ad formats

- [Interstitial ads](#)
- [Rewarded ads](#)

Integration

To work with [IronSource](#), you need to connect the IronSourceYandexMobileAdsAdapters library.

The IronSourceYandexMobileAdsAdapters library has been adapted to work with the CocoaPods dependency management system and supports a static integration method.

1. [Set up mediation](#) in the Yandex Partner interface and YAN interface.
2. To connect the library, add the dependency to the project's Podfile:

```
pod 'IronSourceYandexMobileAdsAdapters', '7.3.0.3'
```

Integration examples

- [Ad example](#)

Enabling Mintegral



Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found [here](#).

Note:

To track app installations in mobile mediation, you need to configure `Info.plist` for the adapter. Follow the ad network [documentation](#).

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.9.0.

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

MintegralSDK

Minimum supported version: 7.3.9.

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

iOS

Minimum supported version: 12.0.

Supported ad formats

- [Banner ads](#)
- [Interstitial ads](#)
- [Rewarded ads](#)

Integration

To work with [Mintegral](#), you need to connect the `MintegralYandexMobileAdsAdapters` library.

The `MintegralYandexMobileAdsAdapters` library has been adapted to work with the CocoaPods dependency management system and supports a static integration method.

1. [Set up mediation](#) in the Yandex Partner interface and YAN interface.
2. To connect the library, add the dependency to the project's `Podfile`:

```
pod 'MintegralYandexMobileAdsAdapters', '7.3.9.0'
```

Integration examples

- [Ad example](#)

Enabling AdColony



Warning:

This is an archived version of the documentation. Actual documentation for all platforms can be found [here](#).

Note:

To track app installations in mobile mediation, you need to configure `Info.plist` for the adapter. Follow the ad network [documentation](#).

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.9.0.

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

AdColonySDK

Minimum supported version: 4.9.0.

Maximum supported version: 4.10.0 (up to but not including).

iOS

Minimum supported version: 12.0.

Supported ad formats

- [Banner ads](#)
- [Interstitial ads](#)
- [Rewarded ads](#)

Integration

To work with [AdColony](#), you need to connect the AdColonyYandexMobileAdsAdapters library.

The AdColonyYandexMobileAdsAdapters library has been adapted to work with the CocoaPods dependency management system and supports a static integration method.

1. [Set up mediation](#) in the Yandex Partner interface and YAN interface.
2. To connect the library, add the dependency to the project's Podfile:

```
pod 'AdColonyYandexMobileAdsAdapters', '4.9.0.4'
```

Integration examples

- [Ad example](#)

Enabling Chartboost

**Warning:**

This is an archived version of the documentation. Actual documentation for all platforms can be found [here](#).

Note:

To track app installations in mobile mediation, you need to configure Info.plist for the adapter. Follow the ad network [documentation](#).

Supported versions of libraries and platforms:

YandexMobileAdsSDK

Minimum supported version: 5.9.0.

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

ChartboostSDK

Minimum supported version: 9.3.1.

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

iOS

Minimum supported version: 12.0.

Supported ad formats

- [Banner ads](#)

- [Interstitial ads](#)
- [Rewarded ads](#)

Integration

To work with [Chartboost](#), you need to connect the ChartboostYandexMobileAdsAdapters library.

The ChartboostYandexMobileAdsAdapters library has been adapted to work with the CocoaPods dependency management system and supports a static integration method.

1. [Set up mediation](#) in the Yandex Partner interface and YAN interface.
2. To connect the library, add the dependency to the project's Podfile:

```
pod 'ChartboostYandexMobileAdsAdapters', '9.3.1.0'
```

Integration examples

- [Ad example](#)

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: 5.9.0.

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

BigoADSSDK

Minimum supported version: 2.2.0.

Maximum supported version: 2.3.2 (up to but not including).

iOS

Minimum supported version: 12.0.

Supported ad formats

- [Banner ads](#)
- [Interstitial ads](#)
- [Rewarded ads](#)

Integration

To work with [BigoADS](#), you need to connect the BigoADSYandexMobileAdsAdapters library.

The BigoADSYandexMobileAdsAdapters library has been adapted to work with the CocoaPods dependency management system and supports a static integration method.

1. [Set up mediation](#) in the Yandex Partner interface and YAN interface.
2. To connect the library, add the dependency to the project's Podfile:

```
pod 'BigoADSYandexMobileAdsAdapters', '2.3.2.0'
```

Integration examples

- [Ad example](#)