

Интеграция

Интеграция

10.07.2024

Яндекс

Интеграция. Версия 1.0

Дата подготовки документа: 10.07.2024

Этот документ является составной частью технической документации Яндекса.

© 2008—2024 ООО «ЯНДЕКС». Все права защищены.

Предупреждение об исключительных правах и конфиденциальной информации

Исключительные права на все результаты интеллектуальной деятельности и приравненные к ним средства индивидуализации юридических лиц, товаров, работ, услуг и предприятий, которым предоставляется правовая охрана (интеллектуальную собственность), используемые при разработке, поддержке и эксплуатации службы, включая, но не ограничиваясь, программы для ЭВМ, базы данных, изображения, тексты, другие произведения, а также изобретения, полезные модели, товарные знаки, знаки обслуживания, коммерческие обозначения и фирменные наименования, принадлежат ООО «ЯНДЕКС» либо его лицензиарам.

Использование результатов интеллектуальной деятельности и приравненных к ним средств индивидуализации в целях, не связанных с разработкой, поддержкой и эксплуатацией службы, не допускается без получения предварительного согласия правообладателя. Настоящий документ содержит конфиденциальную информацию ООО «ЯНДЕКС». Использование конфиденциальной информации в целях, не связанных с разработкой, поддержкой и эксплуатацией службы, а равно как и разглашение таковой, не допускается. При этом под разглашением понимается любое действие или бездействие, в результате которых конфиденциальная информация в любой возможной форме (устной, письменной, иной форме, в том числе с использованием технических средств) становится известной третьим лицам без согласия обладателя такой информации либо вопреки трудовому или гражданско-правовому договору.

Отношения ООО «ЯНДЕКС» с лицами, привлекаемыми для разработки, поддержки и эксплуатации службы, регулируются законодательством Российской Федерации и заключаемыми в соответствии с ним трудовыми и/или гражданско-правовыми договорами (соглашениями). Нарушение требований об охране результатов интеллектуальной деятельности и приравненных к ним средств индивидуализации, а равно как и конфиденциальной информации, влечет за собой дисциплинарную, гражданско-правовую, административную или уголовную ответственность в соответствии с законодательством Российской Федерации.

Контактная информация

ООО «ЯНДЕКС»

<https://www.yandex.ru>

Тел.: +7 495 739 7000

Email: pr@yandex-team.ru

Главный офис: 119021, Россия, г. Москва, ул. Льва Толстого, д. 16

Содержание

Подключение плагина.....	4
Форматы рекламы.....	5
Баннер.....	5
Типы адаптивного баннера.....	7
Добавление Banner в проект.....	9
Очистка рекламы.....	11
Пример работы с адаптивным баннером.....	11
Полноэкранная реклама.....	11
Реклама с вознаграждением.....	13
GDPR.....	15
Общие сведения.....	15
Краткое руководство.....	15

Подключение Mobile Ads Unity плагина



Внимание:

Это архивная версия документации. Актуальная документация по всем платформам находится [здесь](#).

Mobile Ads Unity — плагин для игровой платформы [Unity3d](#), включающий поддержку Yandex Mobile Ads SDK.

Примечание:

1. Для работы SDK требуется Target API Level версии 31 и выше.
2. Для загрузки любого вида рекламы необходима версия iOS 12.0 и выше.

Интеграция плагина

Примечание: `yandex-ads-unity-plugin` работает только в окружениях Android и iOS. Работа в редакторе Unity невозможна.

Lite-версия

1. Скачайте каталог [yandex-ads-unity-plugin](#) и добавьте пакет `yandex-mobileads-lite-2.9.0.unitypackage`. Вместе с ним будет предложено установить Google resolver. Если в ваш проект уже добавлен Google resolver, уберите галочку.

Как добавить пакет

1. Выберите необходимый плагин (**Assets** → **Import Package** → **Custom Package**) и нажмите кнопку **Import**.
2. С помощью Google resolver установите зависимости: включите **auto-resolve** или выберите в меню пункт **Assets** → **External Dependency Manager** → **Android Resolver** → **Resolve**.
3. Чтобы проверить работу Mobile Ads Unity плагина, воспользуйтесь одним из демонстрационных скриптов в каталоге **samples** репозитория [yandex-ads-unity-plugin](#). Скопируйте скрипт в каталог с проектом и добавьте как **Component** в основную камеру.

Понижение Target API Level

Чтобы понизить Target API Level до версии 30, добавьте в `mainTemplate.gradle` и `launcherTemplate.gradle` (если `launcherTemplate` используется в проекте) явное понижение версии:

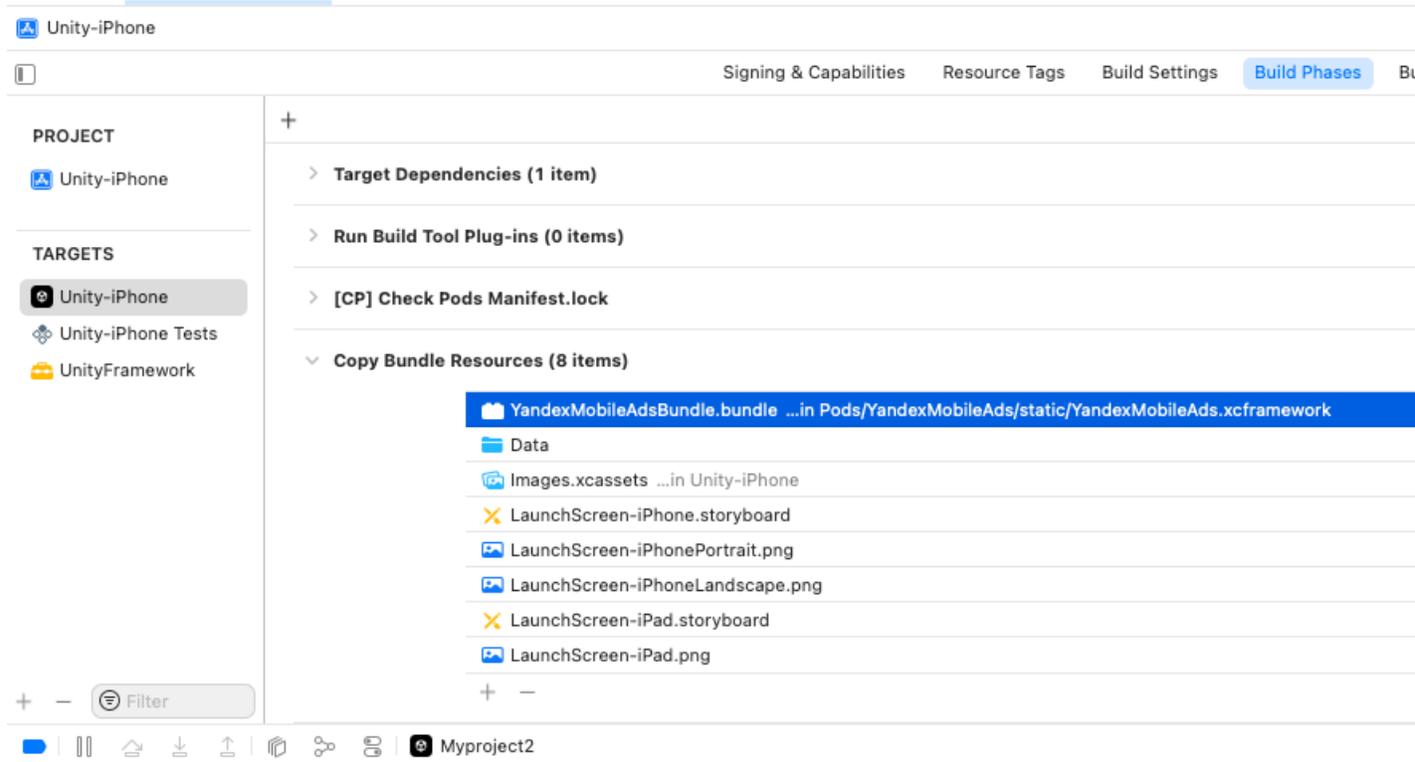
```
configurations.all {
    resolutionStrategy {
        force 'androidx.core:core:1.6.0'
        force 'androidx.core:core-ktx:1.6.0'
    }
}
```

Однако, рекомендуется обновление до Target API Level версии 31, так как у Google есть ограничения на выпуск обновлений для приложений с устаревшей версией Target API Level. Подробнее в [статье](#).

Описание ошибок

Полноэкранная реклама Unity Ads не отображается, ошибка «Incorrect fullscreen view»

Во время запуска полноэкранной рекламы на iOS возможна ошибка «Incorrect fullscreen view». При возникновении данной проблемы проверьте, что в настройках **Build Phases**, секции **Copy Bundle Resources** добавлено значение **YandexMobileAdsBundle.bundle**. Если значение отсутствует, добавьте его.



```
[CoreLocation] TH
on the main threa
`-locationManager
`authorizationSta
2022-09-27 12:48:09.5
[CoreLocation] TH
on the main threa
`-locationManager
`authorizationSta
2022-09-27 12:48:09.5
[CoreLocation] TH
on the main threa
`-locationManager
```

Форматы рекламы

Баннерная реклама



Внимание:

Это архивная версия документации. Актуальная документация по всем платформам находится [здесь](#).

Баннер — это настраиваемое объявление, которое занимает часть экрана и реагирует на нажатие.

Добавление Banner в проект

Чтобы отобразить баннер в вашем приложении, создайте объект Banner в скрипте (на C#), который прикреплен к GameObject.

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

public class YandexMobileAdsBannerDemoScript : MonoBehaviour
```

```

{
    private Banner banner;
    ...
    private void RequestBanner()
    {
        string adUnitId = "demo-banner-yandex";

        // Set flexible banner maximum width and height
        AdSize bannerMaxSize = AdSize.FlexibleSize(GetScreenWidthDp(), 100);
        // Or set sticky banner maximum width
        //AdSize bannerMaxSize = AdSize.StickySize(GetScreenWidthDp());

        banner = new Banner(adUnitId, bannerMaxSize, AdPosition.BottomCenter);
    }

    // Example how to get screen width for request
    private int GetScreenWidthDp()
    {
        int screenWidth = (int)Screen.safeArea.width;
        return ScreenUtils.ConvertPixelsToDp(screenWidth);
    }
    ...
}

```

Конструктор Banner содержит следующие параметры:

- AdUnitId — уникальный идентификатор, который выдается в Партнерском интерфейсе и имеет следующий вид: R-M-XXXXXX-Y;
- AdSize — размер баннера, который необходимо показать;
- AdPosition — позиция на экране.

Поддерживаются следующие размеры баннера:

- FlexibleSize(int width, int height) — адаптивный баннер с заданной максимальной шириной и высотой.
- StickySize(int width) — адаптивный баннер с заданной максимальной шириной.
- Фиксированные размеры. Устаревший вариант запрашиваемого размера, рекомендовано использовать Flexible или Sticky размеры.

Размер	Устройство	Константа AdSize
320 x 50	Телефоны и планшеты	BANNER_320x50
320 x 100	Телефоны и планшеты	BANNER_320x100
300 x 250	Телефоны и планшеты	BANNER_300x250
300 x 300	Телефоны и планшеты	BANNER_300x300
240 x 400	Телефоны и планшеты	BANNER_240x400
400 x 240	Телефоны и планшеты	BANNER_400x240
728 x 90	Планшеты	BANNER_728x90

Загрузка рекламы

После создания и настройки объекта класса Banner необходимо загрузить рекламу. Для загрузки рекламы используйте метод LoadAd, принимающий в качестве параметра объект AdRequest.

```

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

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

```

События баннерной рекламы

Чтобы отслеживать события, происходящие в баннерной рекламе, зарегистрируйте делегата для соответствующего EventHandler, как показано ниже:

```
...
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);
}
}
```

Очистка рекламы

Когда объект рекламы больше не нужен, его можно удалить. Для этого вызовите метод Destroy:

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

Адаптивный баннер

Адаптивный баннер — это баннер, органично вписывающийся в размеры блока, определенные пользователем. В зависимости от способа интеграции адаптивного баннера, для него подбирается оптимальная высота с заданной шириной, либо используются указанные размеры рекламного места.

Примечание:

Про создание рекламного блока для адаптивного баннера можно прочитать в [справке Рекламной сети](#).

Типы адаптивного баннера



Внимание:

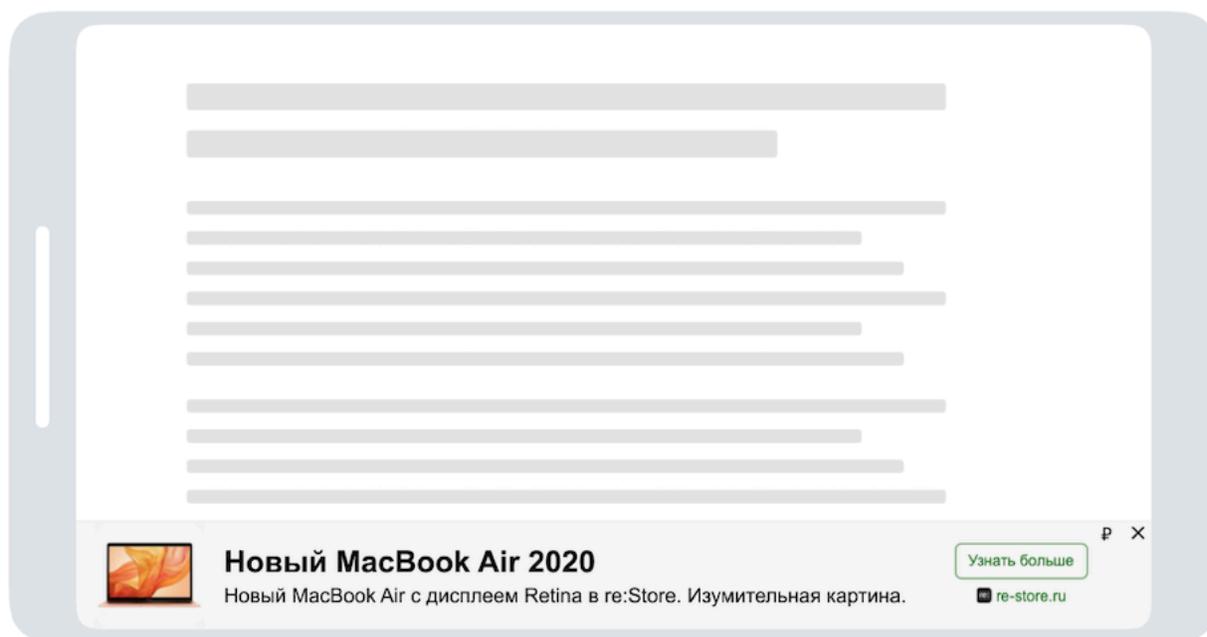
Это архивная версия документации. Актуальная документация по всем платформам находится [здесь](#).

Баннер с заданной шириной

Особенности:

1. Альтернатива баннерам с размерами 320x50 (при определении высоты баннера сохраняется соотношение сторон 320x50).
2. Фиксация баннера в верхней или нижней части экрана (настраивается в приложении).
3. Использование заданной ширины баннера, а не ширины экрана устройства. Это позволяет учитывать особенности дисплея.
4. Ширина адаптивного баннера задается с помощью метода `AdSize.StickySize(int width)`.

Примеры отображения адаптивного баннера:

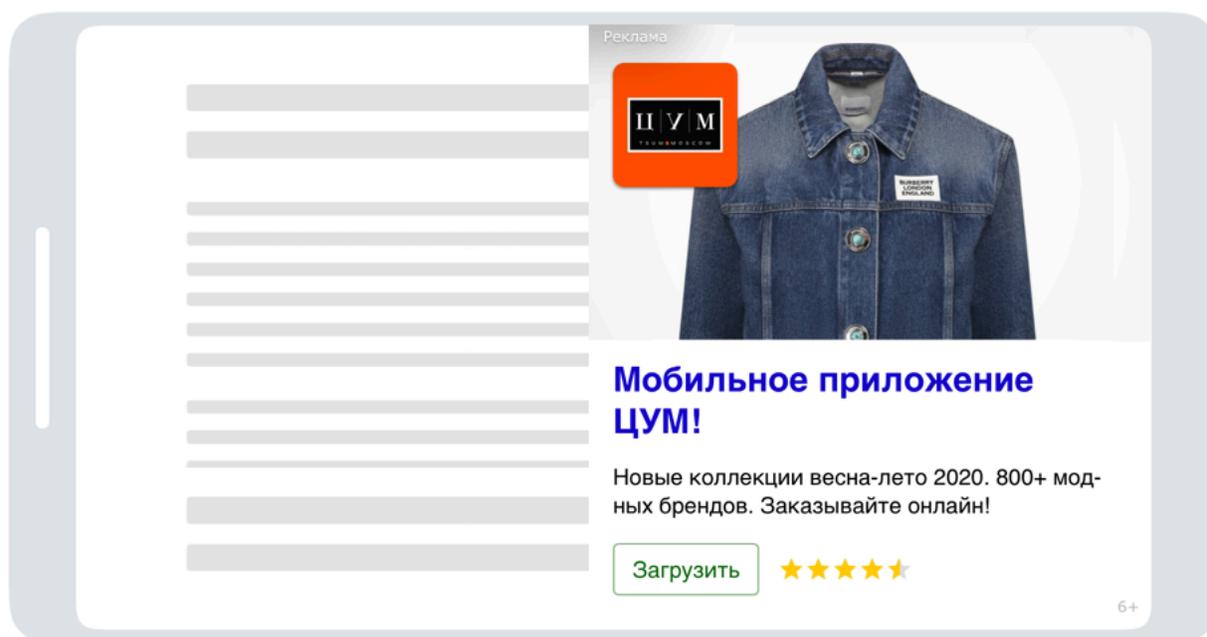


Баннер с заданной шириной и высотой

Особенности:

1. Адаптивный баннер заполняет весь блок, используя заданную ширину и высоту.
2. Ширина и высота адаптивного баннера задается с помощью метода `AdSize.FlexibleSize(int width, int height)`.

Примеры отображения адаптивного баннера:



Добавление Banner в проект

1. Чтобы отобразить баннер в вашем приложении, создайте объект Banner в скрипте (на C#), который прикреплен к GameObject.

```

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

public class YandexMobileAdsBannerDemoScript : MonoBehaviour
{
    private Banner banner;
    ...
    private void RequestBanner()
    {
        string adUnitId = "demo-banner-yandex";
        banner = ...
    }
    ...
}
    
```

- Укажите размер баннера и инициализируйте баннер в приложении. Используйте метод [ScreenUtils.ConvertPixelsToDp](#) чтобы преобразовать ширину в пикселях в density independent пиксели.

Баннер с заданной шириной

```
AdSize adSize = AdSize.StickySize(width);
banner = new Banner(adUnitId, adSize, AdPosition.BottomCenter);
```

Баннер с заданной шириной и высотой

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

Загрузка рекламы

После создания и настройки объекта класса Banner необходимо загрузить рекламу. Для загрузки рекламы используйте метод LoadAd, принимающий в качестве параметра объект AdRequest.

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

События баннерной рекламы

Чтобы отслеживать события, происходящие в баннерной рекламе, зарегистрируйте делегата для соответствующего EventHandler, как показано ниже:

```
...
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);
}
```

```
}
```

Очистка рекламы

Когда объект рекламы больше не нужен, его можно удалить. Для этого вызовите метод `Destroy`:

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

Пример работы с адаптивным баннером

Следующий код демонстрирует создание, настройку и загрузку объекта `Banner`:

```
...
public class YandexMobileAdsBannerDemoScript : MonoBehaviour
{
    private Banner banner;
    ...
    private void RequestBanner()
    {
        string adUnitId = "demo-banner-yandex";
        int screenWidth = (int)Screen.safeArea.width;
        int width = ScreenUtils.ConvertPixelsToDp(screenWidth);
        AdSize adSize = AdSize.StickySize(width);
        banner = new Banner(adUnitId, adSize, AdPosition.BottomCenter);
        AdRequest request = new AdRequest.Builder().Build();
        banner.LoadAd(request);
    }
    ...
}
```

Узнайте больше

[Пример рекламы](#)

Полноэкранная реклама



Внимание:

Это архивная версия документации. Актуальная документация по всем платформам находится [здесь](#).

Полноэкранная реклама (Interstitial) — это настраиваемое объявление, отображаемое на весь экран и реагирующее на нажатие.

Добавление `Interstitial` в проект

Чтобы подключить рекламу, создайте объект `Interstitial` в скрипте (на C#), который прикреплен к `GameObject`.

```
...
using YandexMobileAds;
using YandexMobileAds.Base;
...
public class YandexMobileAdsInterstitialDemoScript : MonoBehaviour
{
    private Interstitial interstitial;
    ...
    private void RequestInterstitial()
    {
        string adUnitId = "demo-interstitial-yandex";
        interstitial = new Interstitial(adUnitId);
    }
    ...
}
```

Конструктор `Interstitial` содержит параметр `adUnitId` — уникальный идентификатор, который выдается в Партнерском интерфейсе и имеет следующий вид: R-M-XXXXXX-Y.

Загрузка рекламы

После создания и настройки объекта класса `Interstitial` необходимо загрузить рекламу. Для загрузки рекламы используйте метод `LoadAd`, принимающий в качестве параметра объект `AdRequest`.

```
...
private void RequestInterstitial()
{
    ...
    AdRequest request = new AdRequest.Builder().Build();
    interstitial.LoadAd(request);
    ...
}
...
```

Показ рекламы

После того, как реклама загружена, ее можно показать:

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

События полноэкранной рекламы

Чтобы отслеживать события, происходящие в полноэкранной рекламе, зарегистрируйте делегата для соответствующего `EventHandler`, как показано ниже:

```
...
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);
}
```

Очистка рекламы

Когда объект рекламы больше не нужен, его можно удалить. Для этого вызовите метод `Destroy`:

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

Реклама с вознаграждением



Внимание:

Это архивная версия документации. Актуальная документация по всем платформам находится [здесь](#).

Реклама с вознаграждением (Rewarded Ad) — настраиваемое объявление, отображаемое на весь экран. За просмотр такой рекламы пользователь получает вознаграждение.

Добавление Rewarded Ad в проект

Чтобы подключить рекламу, создайте объект `RewardedAd` в скрипте (на C#), который прикреплен к `GameObject`.

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

public class YandexMobileAdsRewardedAdDemoScript : MonoBehaviour
{
    private RewardedAd rewardedAd;
    ...
    private void RequestRewardedAd()
    {
        string adUnitId = "demo-rewarded-yandex";
        rewardedAd = new RewardedAd(adUnitId);
    }
    ...
}
```

Конструктор `RewardedAd` содержит параметр `adUnitId` — уникальный идентификатор, который выдается в Партнерском интерфейсе и имеет следующий вид: R-M-XXXXXX-Y.

Загрузка рекламы

После создания и настройки объекта класса `RewardedAd` необходимо загрузить рекламу. Для загрузки рекламы используйте метод `LoadAd`, принимающий в качестве параметра объект `AdRequest`.

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

Показ рекламы

После того, как реклама загружена, ее можно показать:

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

События рекламы с вознаграждением

Чтобы отслеживать события, происходящие в рекламе с вознаграждением, зарегистрируйте делегата для соответствующего EventHandler, как показано ниже:

```
...
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: amount = " + args.amount + ", type = " + args.type);
}

public void HandleRewardedAdFailedToShow(object sender, AdFailureEventArgs args)
{

```

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

Очистка рекламы

Когда объект рекламы больше не нужен, его можно удалить. Для этого вызовите метод `Destroy`:

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

GDPR



Внимание:

Это архивная версия документации. Актуальная документация по всем платформам находится [здесь](#).

Общие сведения

Весной 2018 года в силу вступил общий регламент по защите данных (General Data Protection Regulation, сокращенно GDPR). Регламент регулирует сбор и обработку информации о физических лицах — гражданах Европейской экономической зоны и Швейцарии. Он призван усилить защиту конфиденциальных данных и сделать прозрачными все элементы сбора, хранения и обработки информации в интернете.

GDPR имеет экстерриториальное действие и применяется ко всем компаниям, которые обрабатывают персональные данные граждан Европейской экономической зоны и Швейцарии, независимо от местонахождения такой компании.

Начиная с версии 0.3.0, Yandex Mobile Ads Unity позволит ограничить сбор данных пользователей, расположенных в Европейской экономической зоне и Швейцарии, при отсутствии их согласия на это.

Краткое руководство

Согласие пользователя на обработку персональных данных необходимо передавать в SDK при каждом запуске приложения.

1. Подключите Yandex Mobile Ads Unity по [инструкции](#).
2. Отобразите для пользователя диалог, предлагающий принять пользовательское соглашение на обработку персональных данных.
3. Передайте полученное значение в Yandex Mobile Ads SDK с помощью метода [SetUserConsent](#). Данные пользователей, находящихся в GDPR-регионе, будут обрабатываться только при наличии согласия пользователя на обработку данных.

```
MobileAds.SetUserConsent(true);
```