

UTNotifications Manual

Version 1.8

[Introduction](#)

[Getting Started](#)

[Creating Local Notifications](#)

[Using Notification Profiles \(Sounds, Icons and Other Attributes\)](#)

[Custom User Data and Handling Received / Clicked Notifications](#)

[Notification Buttons \(Android\)](#)

[Image Notifications \(Android\)](#)

[Open URL Notifications \(Android\)](#)

[Push Notifications Overview](#)

[What You Need for Push Notifications](#)

[General](#)

[iOS: Apple Push Notification Service \(APNS\)](#)

[Android: Firebase Cloud Messaging \(FCM\)](#)

[Android: Amazon Device Messaging \(ADM\)](#)

[Windows Store: Windows Push Notification Services \(WNS\)](#)

[Push Notifications Payload Format](#)

[FCM Topics](#)

[Handling Push Registration Failures](#)

[Configuring Apple Push Notification Service \(APNS\)](#)

[Configuring the Firebase Cloud Messaging \(FCM\)](#)

[Apply Credentials and Test](#)

[Configuring the Amazon Device Messaging \(ADM\)](#)

[Getting Your OAuth Credentials and API Key](#)

[Apply Credentials and Test](#)

[Configuring the Windows Push Notification Services \(WNS\)](#)

[Register your app with the Dashboard](#)

[Obtain the identity values for your app](#)

[Apply Credentials and Test](#)

[Contacts](#)

Introduction

[API Reference](#) | [Forum](#) | [Support Email](#) | [Issue Tracking](#)

UTNotifications is a professional Unity extension that is yet very convenient and easy to use. It provides a single cross-platform API for posting and handling local, scheduled (including those appearing once and those repeating) and push notifications. It fully supports iOS (7.0 and newer), Android (4.1 and newer, Google Play featured and Amazon Kindle Android devices) and Windows Store/Universal Windows Platform (Windows Phone 8.1, Windows 8.1/10, Universal 8.1, Universal 10).

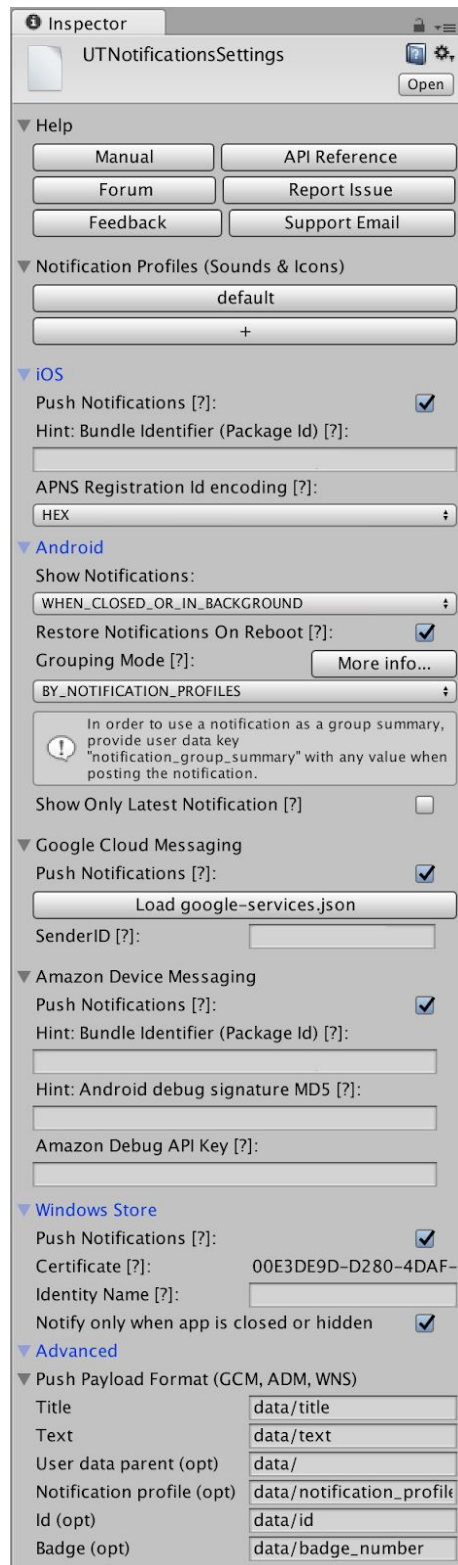
Features:

- Immediate and scheduled (those appearing once and those repeating) local notifications with automated restoring on device reboot.
- Push notifications.
- 2 Android push notifications services: Firebase Cloud Messaging (FCM) & Amazon Device Messaging (ADM) in a single build.
- Completely cross-platform API.
- The full source code is provided as well as the code of the native plugins so one can change and adjust anything one likes.
- A demo push notifications provider web server with the source code is included.
- A sample & test scene.
- A detailed manual and an API Reference docs are included.
- Default or custom notifications sounds and icons.
- Convenient Unity editor extension for configuring.
- Notifications enabling/disabling API for all supported platforms allows one to add notifications toggle to the game options.
- API for handling clicked & received notifications of any type both local and push.
- One can attach custom data to the notification of any type and access it while handling the received notification.
- Hiding or cancelling a specific notification or all of them.
- Application icon badge number management API for iOS and Android.
- Android: Image notifications.
- Android: Complete integration with Android 8+ Notification Channels.
- Android: Custom buttons.
- Android: High Priority/Heads-Up notifications support.
- And more!

UTNotifications consists of two main parts: Unity client extension and a demo server that demonstrates a way to send push (remote) notifications to each of the supported platforms. For the production version of your application your own game server or a dedicated notifications server is required, but you can use the provided demo server source code as you like. There is also a number of third-party solutions for the push notification servers, which are compatible with UTNotifications as it uses plain iOS, Google Android, Amazon Android & Windows push notifications services. You can also leverage the asset for local/scheduled notifications only - in that case you don't need any backend. The asset works well with Unity 2017 or newer.

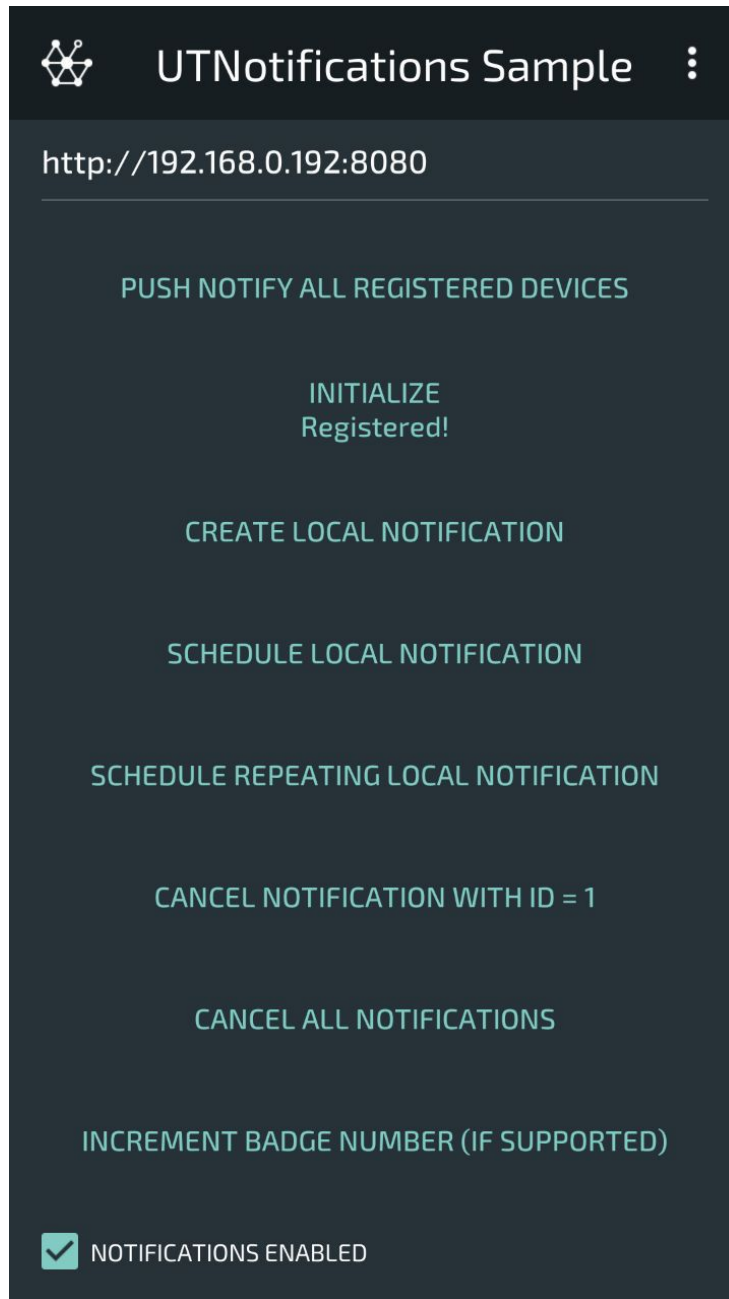
Getting Started

Once you installed the asset into your project, you're able to open its settings from Unity menu: *Edit -> Project Settings -> UTNotifications*.



Please **note**, that moving any of the asset files or folders is not recommended, as it includes a number of editor scripts which rely on the existing directories structure. In case you decide to move anything, please update `Assets/UTNotifications/Src/Settings.cs` accordingly.

Local notifications don't require any additional setup, though a number of configuration options is available for them. Configuring push notifications services is [described below](#). There is an example scene: `Assets/UTNotifications/Sample/UTNotificationsSample.unity` which can be used to get familiar with many of the UTNotifications features and how to work with them. It also helps you checking the asset configuration.



Please **note** that notifications are not available in the Unity editor and some device emulators, so please deploy to a device in order to test or debug notifications related functionality. You can find the API Reference in UTNotifications Unity Settings and [here](#).

Creating Local Notifications

Local notifications are notifications shown by a request of the client application itself. With UTNotifications you can create immediate, scheduled and repeated scheduled local notifications.

The entire UTNotifications API is stored in [namespace](#) `UTNotifications`. So you might like adding a `using` statement to access the contents of that namespace easily:

```
using UTNotifications;
```

Now let's Initialize `UTNotifications.Manager`. It must be done prior to accessing any `UTNotifications` methods. `Awake()` or `Start()` method of some `MonoBehaviour` is a good place for it:

```
public void Start()
{
    UTNotifications.Manager.Instance.Initialize(false);
}
```

`UTNotifications.Manager.Instance` (or just `Manager.Instance` if you added `using UTNotifications`) is the main access point to all the methods of `UTNotifications`. It returns a [singleton](#) instance of the `UTNotifications.Manager` class. We specified `false` in the example above as the value of argument `willHandleReceivedNotifications` of `UTNotifications.Manager.Initialize` as we don't have any intention to handle shown notifications at the moment. For more info on handling notifications, please refer to the [appropriate section](#) of the manual.

It's recommended that you also check the returned value of

`UTNotifications.Manager.Instance.Initialize` call, as returning `false` signals about a failure with the notification system initialization, which makes it impossible to use any of the asset functionality (check the application logs to find more details on the issue). It's always easy to check whether the Manager is in the successfully initialized state by accessing the boolean property `UTNotifications.Manager.Instance.Initialized`.

As soon as the Manager has been initialized, you can start creating local notifications. F.e.:

```
UTNotifications.Manager.Instance.PostLocalNotification("Title", "Text", 1);
```

It creates an immediate local notification with title = "Title", text = "Text" and id = 1. Notification ids are used to identify each notification. A new notification with the same id as an old one replaces that old notification instead of creating a second separate notification. id is also used to hide or cancel a specific notification (see the [API Reference](#) for the details).

Note that with the default settings you will not be able to see or hear any immediate notifications on any of the supported platforms, because notifications are not shown while the application is running by default. You can modify this behaviour in UTNotifications Settings: **Android -> **Show Notifications** and **Windows Store** -> **Notify only when app is closed or hidden**. Unfortunately, iOS doesn't allow controlling it: you can never see any notifications while the app is running on iOS.**

Let's now try to schedule a local notification:

```
UTNotifications.Manager.Instance.ScheduleNotification(15, "Title", "Text", 2);
```

A local notification with title = "Title", text = "Text" and id = 2 will be triggered in 15 seconds after that code is executed. You can also specify a `System.DateTime` value as the first argument as a date and time to trigger the notification.

Similarly you can create a repeated scheduled notification:

```
UTNotifications.Manager.Instance.ScheduleNotificationRepeating(5, 25, "Title",  
"Text", 3);
```

This notification with title = "Title", text = "Text" and id = 3 will be shown first time in 5 seconds after that code is executed and then will be repeated every 25 seconds. There is also a `System.DateTime` version of this method.

Note that the repeating times are approximate and may differ, especially on iOS where only fixed options like every minute, every day, every week and so on are available. So the provided interval value will be approximated by one of the available options.

On Android there is a way to show a notification containing an image:

```
Manager.Instance.ScheduleNotification(10, "Image Notification",  
"Image notification text", 4, new Dictionary<string, string>  
{  
    {  
        "image_url",  
        "https://thecatapi.com/api/images/get?format=src&type=png&size=med"  
    }  
});
```

For more details on image notifications see [Image Notifications \(Android\)](#). See also [Open URL Notifications \(Android\)](#) and [Notification Buttons \(Android\)](#) for more information on some more options.

You can also configure notifications icons, sounds and other notifications attributes. For more details see [Using Notification Profiles \(Sounds, Icons and Other Attributes\)](#).

Using Notification Profiles (Sounds, Icons and Other Attributes)

By default any notification will be posted with a default system notification sound and the application icon. UTNotifications allows defining custom sounds and icons for notifications (custom notification icons are not supported by iOS, no customization is currently supported on Windows Store). What sound and icon is to be used for a specific notification is defined by a **notification profile** - a named set of the notification options. Notification profiles also define [Android 8+ Notification Channels](#) - named user-configurable groups of notifications. It's important to configure at least default profile for Android.

For example, a game might have two kinds of notifications - when a player receives a gift and when some in-game research is complete. One can define two notification profiles: "gift" & "research_complete". The first one will use a gift box icon when shown and some specific sound, while the other will have a bulb icon and another sound.

You can create and edit notification profiles in UTNotifications editor: *Edit -> Project Settings -> UTNotifications -> Notification Profiles (sounds, icons and more)*.

Each of functions `UTNotifications.Manager.Instance.PostLocalNotification`, `UTNotifications.Manager.Instance.ScheduleNotification` and `UTNotifications.Manager.Instance.ScheduleNotificationRepeating` has an optional argument **string notificationProfile** which defines a name of a notification profile used for this notification. Also, all implementations of `UTNotifications.Notification` can optionally have a value of **notificationProfile**.

You can also specify a notification profile for push notifications.

- **iOS (APNS).**

Notification profile name is specified as a sound in the APNS json payload:

```
{
  "aps":
  {
    <...>
    "sound" : "Data/Raw/<NOTIFICATION PROFILE NAME>"
  }
}
```

Note that *<NOTIFICATION PROFILE NAME>* should not contain any file extension.

- **Android.**

Notification profile name is stored in node "data" of the notification json.

FCM:

```
{
  "registration_ids":<...>,
  "data":
  {
    <...>,
    "notification_profile": "<NOTIFICATION PROFILE NAME>"
  }
}
```

ADM:

```
{
  "data":
  {
    <...>,
    "notification_profile": "<NOTIFICATION PROFILE NAME>"
  }
}
```

- **Windows Store (WNS):**

Notification profile name is stored in the payload json root node:

```
{
  <...>,
  "notification_profile": "<NOTIFICATION PROFILE NAME>"
}
```

Push notifications sent from the `UTNotificationsSample` (Notify all registered devices) use notification profile "demo_notification_profile". You can try configuring a profile with that name to see how the feature works.

There is a predefined profile "default", which is used on Android when no notification profile is specified for a notification. It's **important** to configure at least its Small Icon (Android 5.0+): Android, starting with version 5.0, ignores any color information of small notification icons: the icons are considered to be completely white and only alpha channel of the icons is applied (so icons can be only white & transparent). So any non-transparent icons turn into just white squares when using as small notification icons.

Custom User Data and Handling Received / Clicked Notifications

UTNotifications provides a way to handle a list of all notifications shown before or when an app was running, and also a notification which was clicked by a user. Besides, each notification (local and push) can contain some custom data which can be read when handling a clicked or received notification.

In order to do so, please subscribe to UTNotifications.[Manager](#) events `OnNotificationClicked` / `OnNotificationsReceived` **before** initializing UTNotifications. F.e.:

```
UTNotifications.Manager notificationsManager = UTNotifications.Manager.Instance;

notificationsManager.OnNotificationClicked += (notification) =>
{
    Debug.Log(notification.text + " clicked");
};

notificationsManager.OnNotificationsReceived += (receivedNotifications) =>
{
    foreach (var notification in receivedNotifications)
    {
        Debug.Log(notification.text + " received/triggered");
    }
};

notificationsManager.Initialize(true);
```

Here we specified `true` as the value of argument `willHandleReceivedNotifications` of `UTNotifications.Manager.Initialize`, as we'd like to handle received notifications with `OnNotificationsReceived`. Please never set it to `true` if you don't intend to handle received notifications as it can affect the app performance and memory consumption: the received notifications will be stored and never cleaned up then. Handling only clicked notifications doesn't require turning on `willHandleReceivedNotifications`.

Note that iOS doesn't provide the list of all notifications shown when the app wasn't running in foreground. Received notifications list will contain only the notification which was clicked and all the notifications shown while the app is running in foreground. On the rest platforms you'll receive a list of all the shown notifications, even ones displayed while the app was closed.

You can specify a `Dictionary<string, string>` containing any custom data which can then be accessed when handling clicked or received notifications as `ReceivedNotification.userData`. Each of the methods for creating local notifications can accept an optional value of `userData`. Push notifications payload is used to get the value of `userData` when handling them.

Local notifications example:

```
Dictionary<string, string> userData = new Dictionary<string, string>();
userData.Add("event_type", "DAILY_GIFT_RECEIVED");
```



```

Manager.Instance.ScheduleNotificationRepeating(DateTime.Now.AddDays(1),
    TimeUtils.DaysToSeconds(1), "A gift for you!",
    "Start the game to receive your gift", 5, userData);

```

Push notifications example (ADM payload format):

```

{
  "data":
  {
    <...>,
    "event_type": "DAILY_GIFT_RECEIVED"
  }
}

```

Now, let's handle the user data:

```

// Should be subscribed before initializing UTNotifications.Manager
UTNotifications.Manager.Instance.OnNotificationClicked += (notification) =>
{
    if (notification.userData != null &&
        notification.userData.ContainsKey("event_type"))
    {
        string eventType = notification.userData["event_type"];
        switch (eventType)
        {
            case "DAILY_GIFT_RECEIVED":
                ShowDailyGiftDialog();
                break;

            default:
                Debug.LogWarning("Unexpected event_type: " + eventType);
                break;
        }
    }
};

```

Notification Buttons (Android)

Any Android notification can contain an arbitrary number of custom buttons. Each of the buttons has a title and optionally custom user data as Dictionary<string, string>.



Each of functions `UTNotifications.Manager.Instance.PostLocalNotification`, `UTNotifications.Manager.Instance.ScheduleNotification` and `UTNotifications.Manager.Instance.ScheduleNotificationRepeating` has optional argument `ICollection<Button>` buttons to specify the notification buttons, f.e.:

```
using UTNotifications;
using System.Collections.Generic;

List<Button> buttons = new List<Button>();
// (Android only) Just a simple button with some custom user data assigned
```

```

buttons.Add(new Button("Open App", new Dictionary<String, String> {{ "button",
"first" }}}));
// (Android only) "open_url" in userData opens an URL on a notification click
instead of the application. Can be used for the whole notification or a specific
button, like here.
buttons.Add(new Button("Open URL", new Dictionary<String, String>{{ "open_url",
"https://assetstore.unity.com/packages/tools/utnotifications-professional-local-pu
sh-notification-plugin-37767"}, {"button", "second" }}}));
// Repeating scheduled notification
Manager.Instance.ScheduleNotificationRepeating(DateTime.Now.AddSeconds(10), 25,
"Scheduled Repeating Notification", "Click to open the app",
RepeatingNotificationId, userData, "demo_notification_profile", 1, buttons);

```

Push notifications (FCM & ADM) can contain custom buttons too:

```

"data":
{
    <...>,
    "buttons":
    "[
        {
            \"title\": \"<Button title>\",
            \"<Button user data key 1>\": \"<Button user data value 1>\", ...
        },
        <...>
    ]"
}

```

Note that JSON value of node **“buttons”** is actually a JSON array converted to **string**.

Buttons are not supported on the rest platforms at the moment and will be ignored on them.

Image Notifications (Android)

With UTNotifications you can create image notifications, i.e. notifications containing large images. It's supported with both local and push notifications. In order to create an image notification [add a user data argument](#) `“image_url”` with a string value, containing an URL of a picture to use. `“image_url”` value may be a normal `https://` URL, or an Android file system URL: `file:///<full path to a picture file>`.

Open URL Notifications (Android)

You can also make clicking on a notification open a specified URL in a browser instead of activating your application. It's supported by both local and push notifications. To achieve that behavior [add a user data argument](#) `“open_url”` which string value should contain an URL to open on a click.

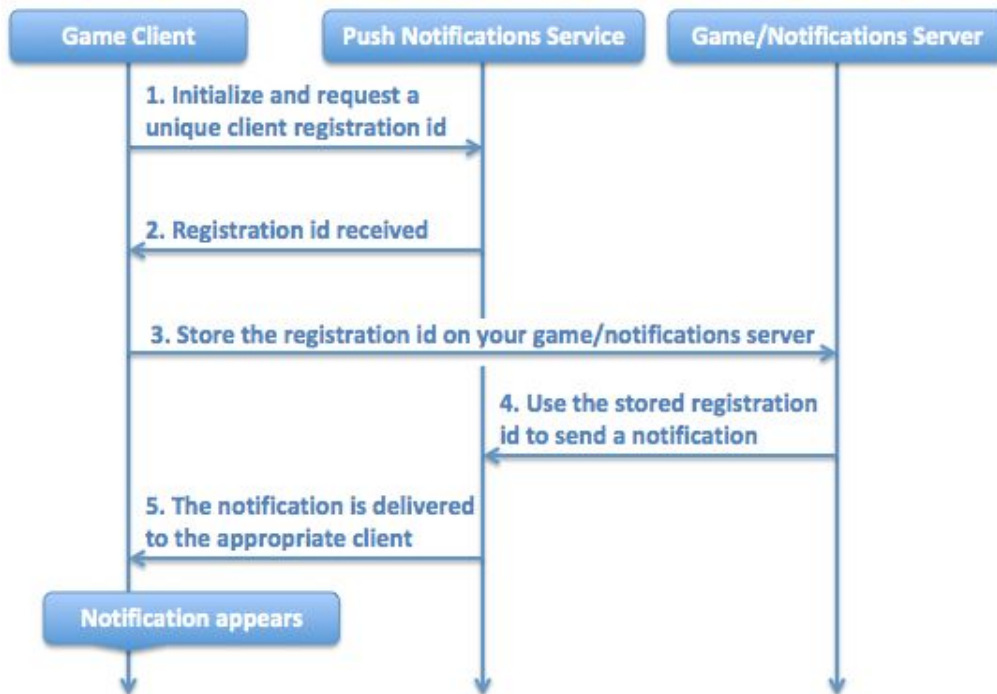
Note that on the rest platforms it activates the app instead of following the URL.

Push Notifications Overview

Push notifications, also known as server notifications or remote notifications, are the notifications sent to a device without a specific request from the client. Unlike local notifications, which don't include any server part, push notifications always originate from a server.

Different devices rely on different methods to deliver push notifications. Apple, for example, uses the Apple Push Notification Service. Different Android devices provide different push notifications services.

Google Play featured ones (i.e. most of Android devices) use Firebase Cloud Messaging (FCM, ex Google Cloud Messaging - GCM). Amazon Android devices (entire **Kindle Fire** series) don't support FCM and have their own Amazon Device Messaging (ADM) API. Windows 8.1+ and Windows Phones use Windows Push Notification Services (WNS). UTNotifications rely on OS-specific push notifications systems internally, but externally provides the common client side API for all the supported services. No matter what OS and service is used, the general scheme is the same:



- 1. Initialize and request a unique registration id.** The client application using a push notifications service ("PNS": one of APNS, FCM, ADM and WNS) API requests a unique identifier for that specific PNS of that specific application on that specific device. *Please **note** that in general it should be done on every start of the app because this identifier can get out of date and the application will receive a new one.* With UTNotifications it's done by calling `UTNotifications.Manager.Instance.Initialize(...)` function.
- 2. Registration id received.** The application (game client) receives the id from PNS API asynchronously or synchronously. In order to receive it you will subscribe to `UTNotifications.Manager.Instance.OnSendRegistrationId` event (please subscribe before calling the `Initialize` function because in some cases receiving the registration id may be done synchronously).
- 3. Store the registration id on your game/notifications server.** You send the received id to your own server which will later send push notifications. You do it in the delegate subscribed to the `OnSendRegistrationId` event.
- 4. Use the stored registration id to send a notification.** Your server requests the server side of PNS API to send (i.e. "push") custom notification to one or more clients using their registration ids which were previously stored. Please see `DemoServer.PushNotificator` class source code (`Assets/UTNotifications/Editor/DemoServer/src/DemoServer/PushNotificator.java`) for an example.

Note that FCM also allows subscribing a device to arbitrary “topics” and then sending push notifications to all the devices that are subscribed to a particular topic, avoiding the need to specify any registration ids. See [the appropriate section](#) for more details.

5. **The notification is delivered to the appropriate client.** PNS delivers the notification to the client with the specified registration id. You don’t have to do anything on this stage with `UNotifications` (because it takes care of everything with both Android PNSes and Windows Store/UWP and there is nothing to be done for iOS). A click on the notification will open your application: it’s being started if has’t been and goes foreground if it was in the background. If you would like to handle incoming notifications please see API Reference for `UNotifications.Manager.OnNotificationsReceived` event and `UNotifications.Manager.Initialize(...)` function.

Please **note** that every push notification service requires some configuring. This is described in the sections below.

What You Need for Push Notifications

General

- A server that is connected to the internet. Push notifications are always sent by a server. For development purposes you can use your computer as the server but for production use you will typically need at something like a VPS (Virtual Private Server).

iOS: Apple Push Notification Service (APNS)

- An iPhone or iPad. Notifications do not work in the simulator, so you will need to test on the device.
- An iOS Developer Program membership. You need to make a new App ID and provisioning profile for each app that uses push, as well as certificates for the server. You do this at the iOS Provisioning Portal (this is described below).
- An OS X computer.

Android: Firebase Cloud Messaging (FCM)

- Any Google Play featured device with Android 4.4+.
- Please **note** that you can’t use Firebase Console for sending push notifications to FCM-enabled devices due to its limitations. For more details see [Push Notifications Payload Format/FCM](#).

Android: Amazon Device Messaging (ADM)

- Any Amazon Kindle Fire device (tablet or phone) except the 1st generation of Kindle Fire tablets which don’t support push notifications.

Windows Store: Windows Push Notification Services (WNS)

- Any Windows Phone 8.1 or Windows 8.1/10 device.

Push Notifications Payload Format

APNS requires any push notifications sent by your server to have a specific format. It is described [in this document](#).

FCM, ADM & WNS don’t have a single fixed structure of message payloads. Each of them accepts a JSON data payload, which then is interpreted by the client application. The client application itself is

responsible for creating notifications based on the payload received from the appropriate service. Fortunately, UTNotifications does this job for you. This is why it requires the JSON payload to be in a specific format, which though you can configure in the asset options in Unity. The default format looks like:

FCM:

```
{
  "registration_ids":["<id1>", ...], <or "to":"id1", or "to":"/topics/topic1",>
  "data":
  {
    "title":"<Title>",
    "text":"<Text>",
    "id":<int id>,
    "badge_number":<int badge>,
    "buttons":
    "[
      {
        \"title\": \"<Button title>\",
        \"<Button user data key 1>\": \"<Button user data value 1>\", ...
      }, ...
    ]\",
    "<User data key 1>\":\"<User data value 1>\", ...
  }
}
```

Please **note** that UTNotifications supports “data”-only messages, and Firebase Console can send only “notification” and “notification” + “data” messages: see https://firebase.google.com/docs/cloud-messaging/concept-options#notifications_and_data_messages. It's important as messages containing “notification” node in their payload are handled by Android itself when the app is not running. It restricts the app drastically: notification profiles, image notifications and handling of received notifications gets impossible; it also creates many other issues. This is why you'll have to send FCM messages either from your own server or using a 3rd party service supporting “data”-only messages.

FCM Topics

UTNotifications supports [FCM Topics Messaging](#), i.e. sending push notifications to all the devices that has been subscribed to a specific topic, without a need to specify the target devices' registration ids. Use `UTNotifications.Manager.Instance.SubscribeToTopic("<any topic name>")` to subscribe to a topic and `UTNotifications.Manager.Instance.UnsubscribeFromTopic("<any topic name>")` to unsubscribe from it (both calls are ignored on all platforms except Android/FCM). When sending a Json payload to FCM, specify `"to":"/topics/<topic name>"`.

ADM:

```
{
  "data":
  {
    "title":"<Title>",
```

```

    "text": "<Text>",
    "id": <int id>,
    "badge_number": <int badge>,
    "buttons":
    [
        {
            \"title\": \"<Button title>\",
            \"<Button user data key 1>\": \"<Button user data value 1>\", ...
        }, ...
    ],
    \"<User data key 1>\": \"<User data value 1>\", ...
}
}

```

WNS:

```

{
    "title": "<Title>",
    "text": "<Text>",
    "id": <int id>,
    "badge_number": <int badge>,
    "<User data key 1>": "<User data value 1>\", ...
}

```

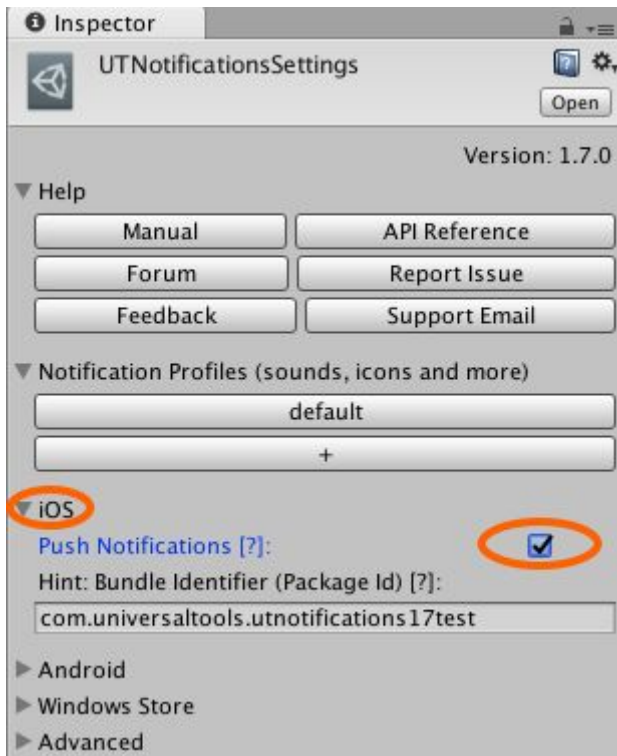
If push server you're going to use sends push messages in a different format, you can configure it in the **UTNotifications Unity settings**: Edit -> Project Settings -> UTNotifications -> Advanced -> Push Payload Format (FCM, ADM, WNS). “data/” prefix is always added to each of the field names (but it's ignored on WNS).

Handling Push Registration Failures

There is a number of reasons that can cause registering for push notifications to fail, including misconfiguration, missing Internet connection and other. In case of a failue, event `UTNotifications.Manager.Instance.OnSendRegistrationId` is not invoked, but another one is invoked instead: `UTNotifications.Manager.Instance.OnPushRegistrationFailed`. Please subscribe to it before initializing the manager if you would like to handle push registration failures.

Configuring Apple Push Notification Service (APNS)

1. In UTNotifications Settings (*Edit -> Project Settings -> UTNotifications*) enable iOS -> Push Notifications toggle:



2. Follow the official Apple instructions on generating and downloading a .p8 APNS enabled encryption key:
https://developer.apple.com/documentation/usernotifications/setting_up_a_remote_notification_server/establishing_a_token-based_connection_to_apns
3. Specify the credentials in *Assets/UTNotifications/Editor/DemoServer/src/main/java/com/universal_tools/demoserver/PushNotificator.java*:

```
22  /// <summary>
23  /// The sample class showing how you can send push notifications for different "providers", such as APNS, FCM, ADM and WNS.
24  /// </summary>
25  public class PushNotificator {
26  // private
27  // Please provide the required values. Find more details in the manual: Assets/UTNotifications/Documentation/Manual.pdf
28  private static final String FIREBASE_SERVER_KEY = null;
29  private static final String AMAZON_CLIENT_ID = null;
30  private static final String AMAZON_CLIENT_SECRET = null;
31  private static final String APNS_AUTH_KEY = "-----BEGIN PRIVATE KEY-----";
32  private static final String APNS_TEAM_ID = "-----";
33  private static final String APNS_KEY_ID = "-----";
34  private static final String APNS_BUNDLE_ID = "com.universaltools.utnotifications17test";
35  private static final boolean APNS_DEVELOPMENT = true;
36  private static final String WINDOWS_PACKAGE_SID = null;
37  private static final String WINDOWS_CLIENT_SECRET = null;
38  }
```

- **APNS_AUTH_KEY**: the contents of the .p8 authentication key, without -----BEGIN PRIVATE KEY-----, -----END PRIVATE KEY----- and any line breaks.
- **APNS_TEAM_ID**: can be found at <https://developer.apple.com/account/#/membership/>
- **APNS_KEY_ID**: is included in the .p8 key file name and also be found at <https://developer.apple.com/account/resources/authkeys/list>:

Certificates, Identifiers & Profiles

[← All Keys](#)

View Key Details

Download

Revoke

Edit

Name

[Redacted]

Key ID

[Redacted]

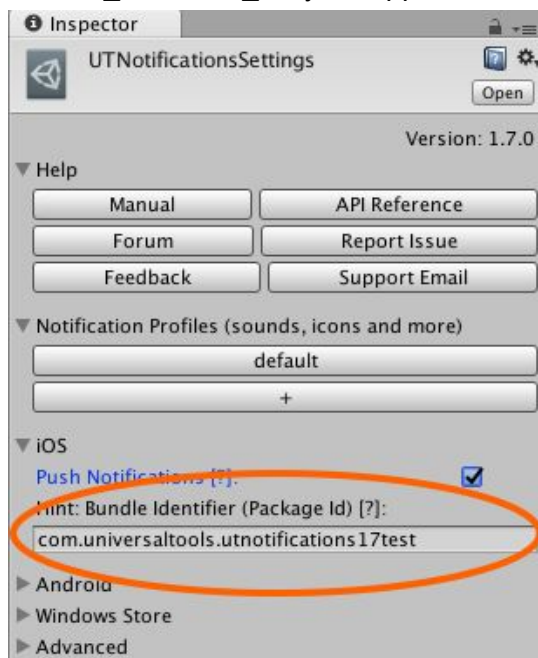
Enabled Services

NAME

CONFIGURATION

APNs

- **APNS_BUNDLE_ID**: your application bundle id, can be found in UTNotifications Settings



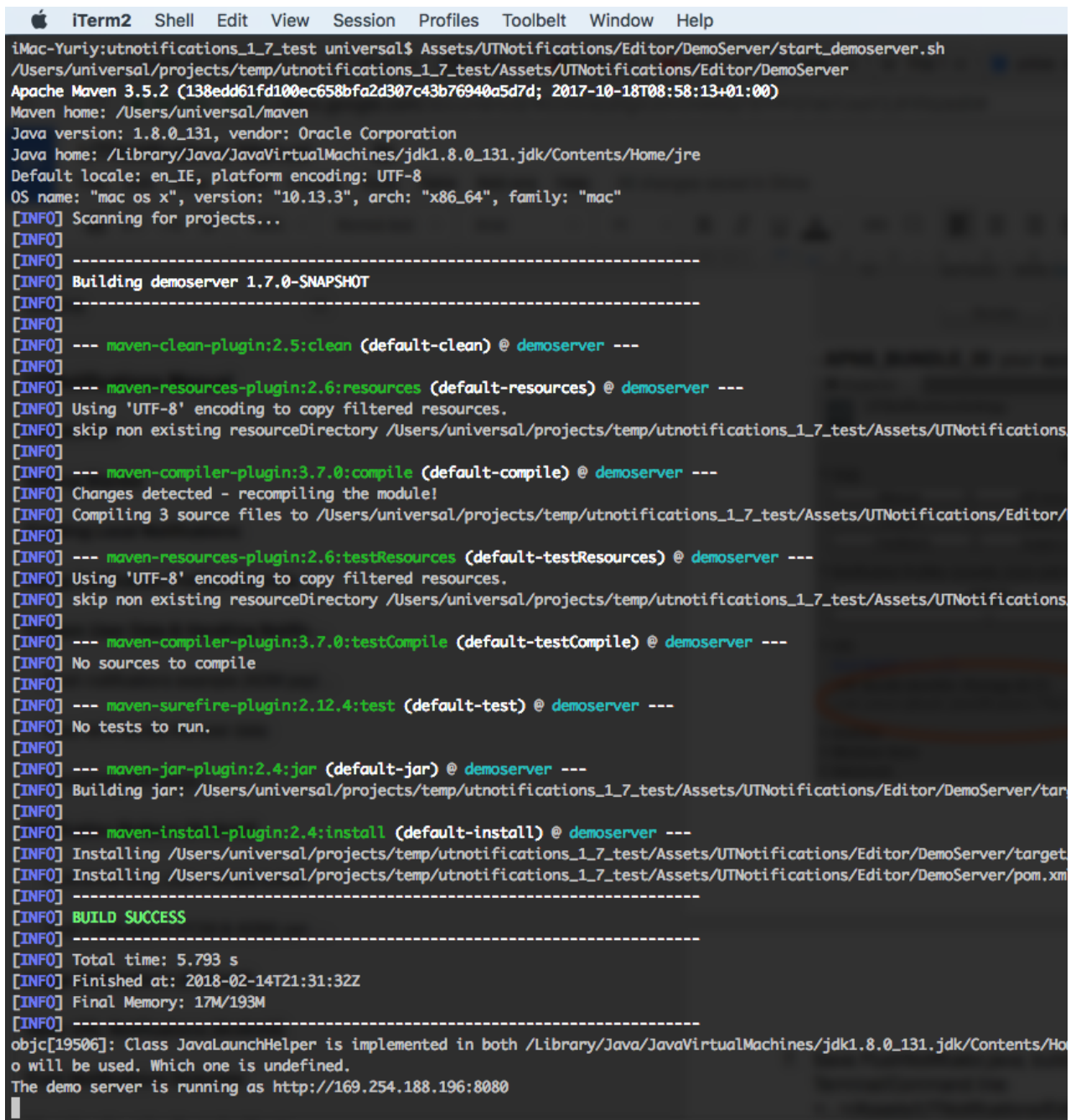
4. Save *PushNotificator.java*, build and start DemoServer, by executing the following script in Terminal/Command line:

`<...>/Assets/UTNotifications/Editor/DemoServer/start_demoSERVER.sh` (macOS / Linux)

or

`<...>\Assets\UTNotifications\Editor\DemoServer\start_demoSERVER.bat` (Windows)

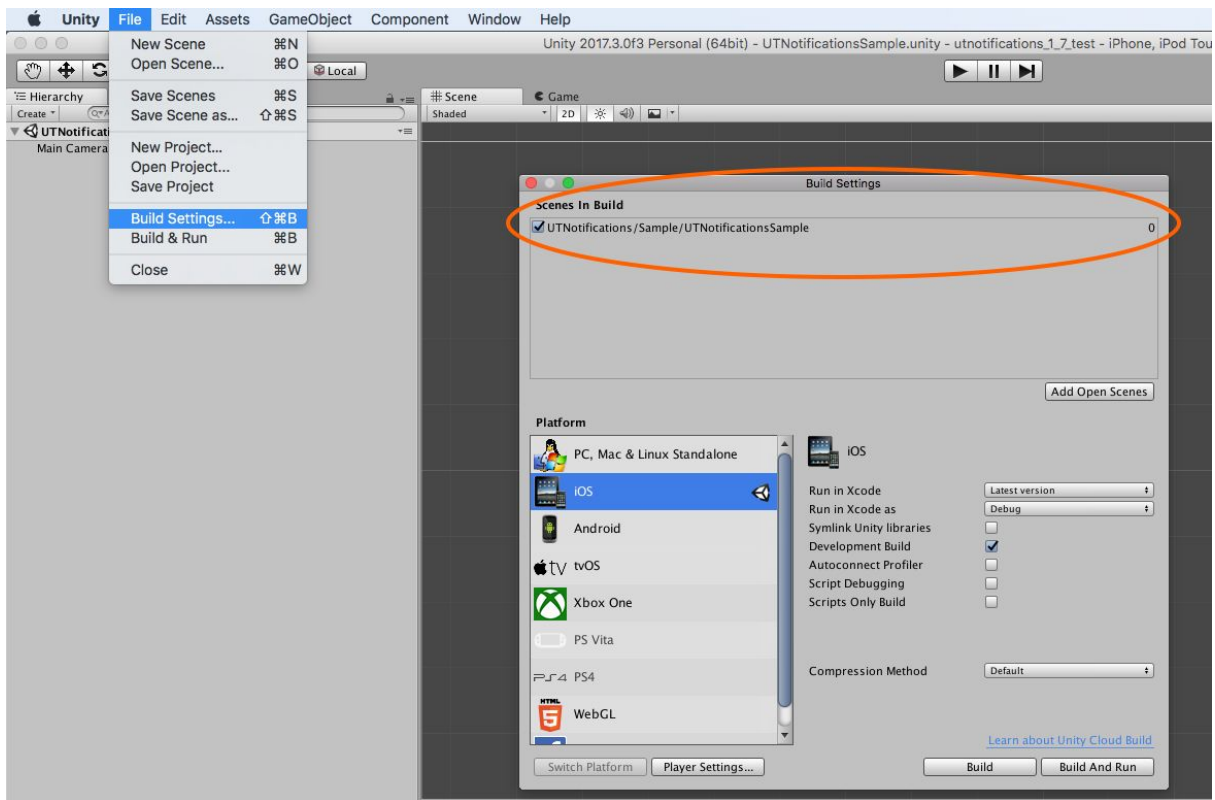
Note that you'll need [JDK](#) and [Maven](#).



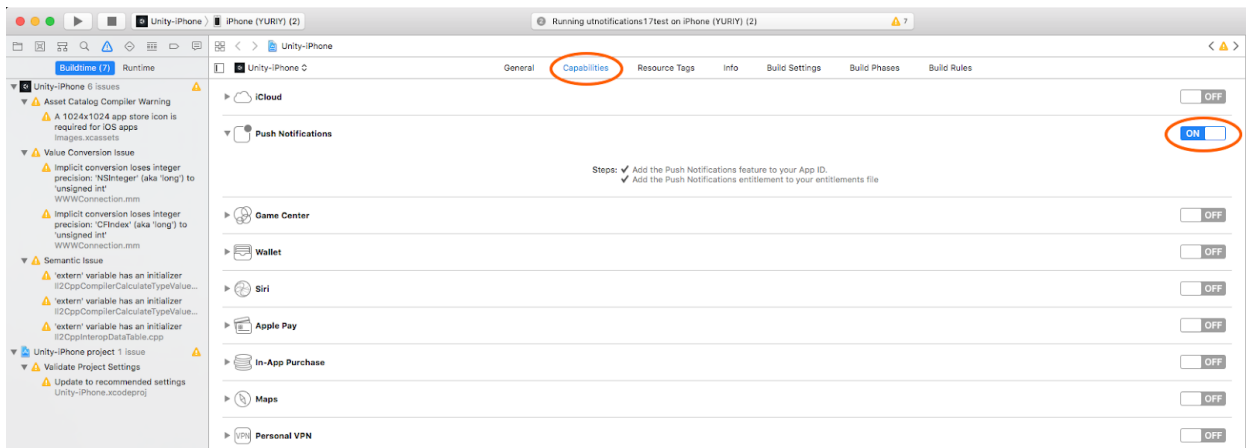
```
iMac-Yuriy:utnotifications_1_7_test universal$ Assets/UTNotifications/Editor/DemoServer/start_demoServer.sh
/Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications/Editor/DemoServer
Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T08:58:13+01:00)
Maven home: /Users/universal/maven
Java version: 1.8.0_131, vendor: Oracle Corporation
Java home: /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/jre
Default locale: en_IE, platform encoding: UTF-8
OS name: "mac os x", version: "10.13.3", arch: "x86_64", family: "mac"
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building demoServer 1.7.0-SNAPSHOT
[INFO] -----
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ demoServer ---
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ demoServer ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications
[INFO]
[INFO] --- maven-compiler-plugin:3.7.0:compile (default-compile) @ demoServer ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 3 source files to /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications/Editor/
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ demoServer ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications
[INFO]
[INFO] --- maven-compiler-plugin:3.7.0:testCompile (default-testCompile) @ demoServer ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ demoServer ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ demoServer ---
[INFO] Building jar: /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications/Editor/DemoServer/target
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ demoServer ---
[INFO] Installing /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications/Editor/DemoServer/target
[INFO] Installing /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications/Editor/DemoServer/pom.xml
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] -----
[INFO] Total time: 5.793 s
[INFO] Finished at: 2018-02-14T21:31:32Z
[INFO] Final Memory: 17M/193M
[INFO] -----
objc[19506]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Ho
o will be used. Which one is undefined.
The demo server is running as http://169.254.188.196:8080
```

5. The running DemoServer will print its hostname (ip address) and port. Please note, that the ip address it prints can be either a local network address (usually 192.168.*.*) or an external (like on the screenshot above). In later case, please find your internal IP address in the OS network settings.

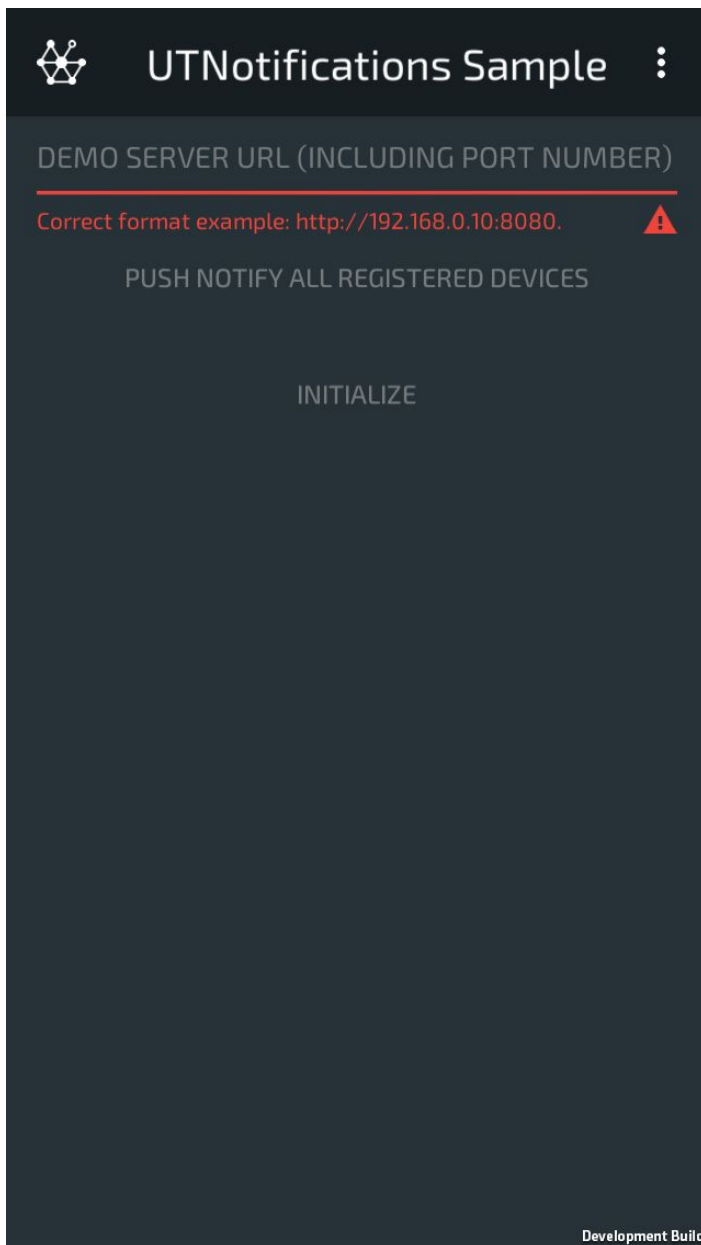
6. Temporarily make UTNotifications/Sample/UTNotificationsSample scene default for your Unity project:



7. Build and run an iOS build. Make sure Push Notifications got enabled in the project Capacities in Xcode (should be done by UTNotifications post build event automatically):

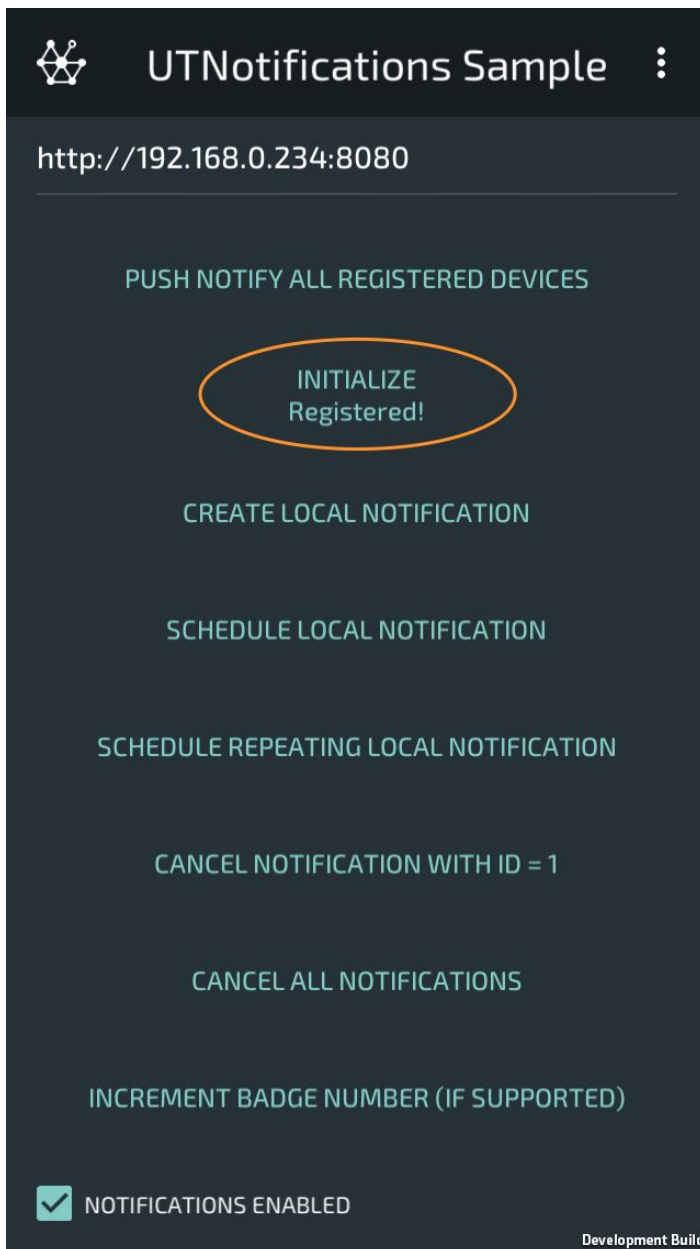


8. The sample scene should start requesting an URL of DemoServer in order to continue:



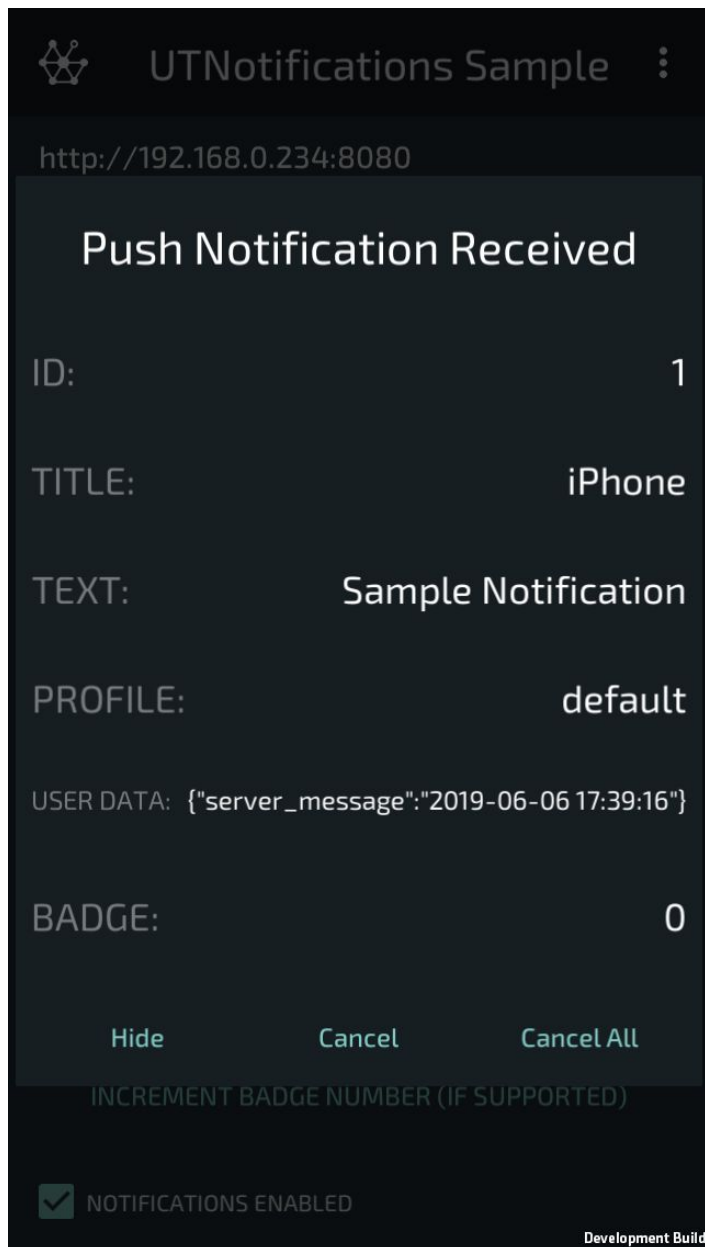
9. Make sure your test device and the computer running DemoServer belong to the same local network (f.e. connected to the same Wi-Fi router). Specify the full URL of the running DemoServer (as **http://<ip address>:8080**) and press **INITIALIZE**:

```
<< /register uid=DB058E75-958D-4707-9E9D-DD86C867B219&provider=APNS&id=c0cbc62f080d528c54cafa7cd
>> HTTP/1.1 200 OK
>> Server: UTNotificationsDemoServer
>> Content-Type: text/html
>> Content-Length: 11
>> Connection: close
>>
>> Registered!
```



10. Press **PUSH NOTIFY ALL REGISTERED DEVICES** to send a push notification to all the DemoServer-registered devices. If everything was configured correctly, you should see how

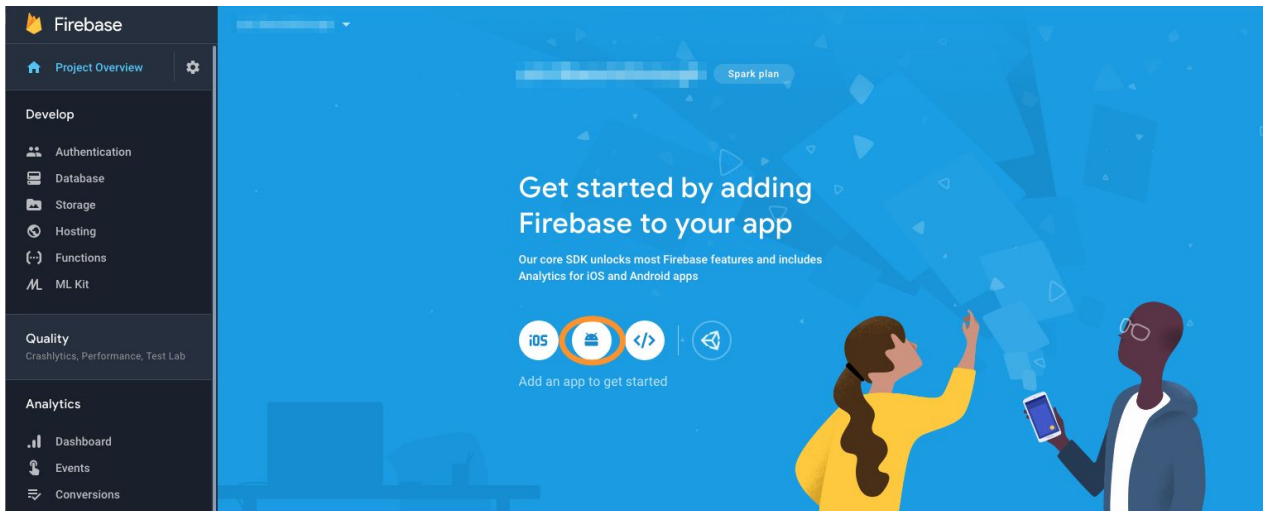
UTNotifications SampleScene handled the push message:



Configuring the Firebase Cloud Messaging (FCM)

Based on FCM official documentation: <https://firebase.google.com/docs/cloud-messaging/>.

1. Open [Firebase Developer Console](#).
2. Create a new Firebase project or import existing Google Project.
3. Press **Add Firebase to your Android app**.



4. Enter your app's bundle id as **Android package name** and press **REGISTER APP** button (no need to specify any other details).

×

Add Firebase to your Android app

1

Register app

Android package name ⓘ

App nickname (optional) ⓘ

My Android App

Debug signing certificate SHA-1 (optional) ⓘ

00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00

Required for Dynamic Links, Invites and Google Sign-In or phone-number support in Auth. Edit SHA-1s in Settings.

Register app

2

Download config file

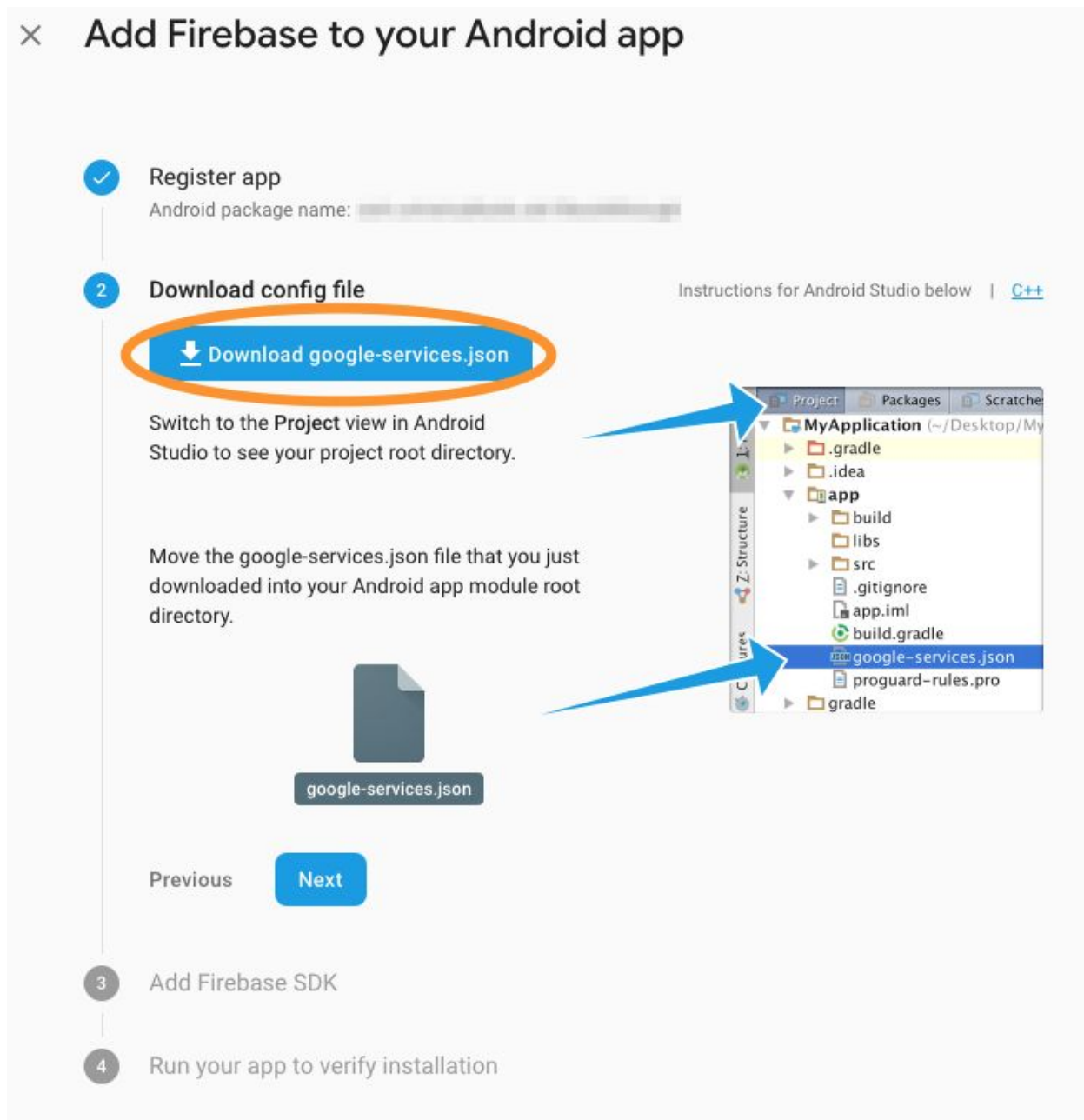
3

Add Firebase SDK

4

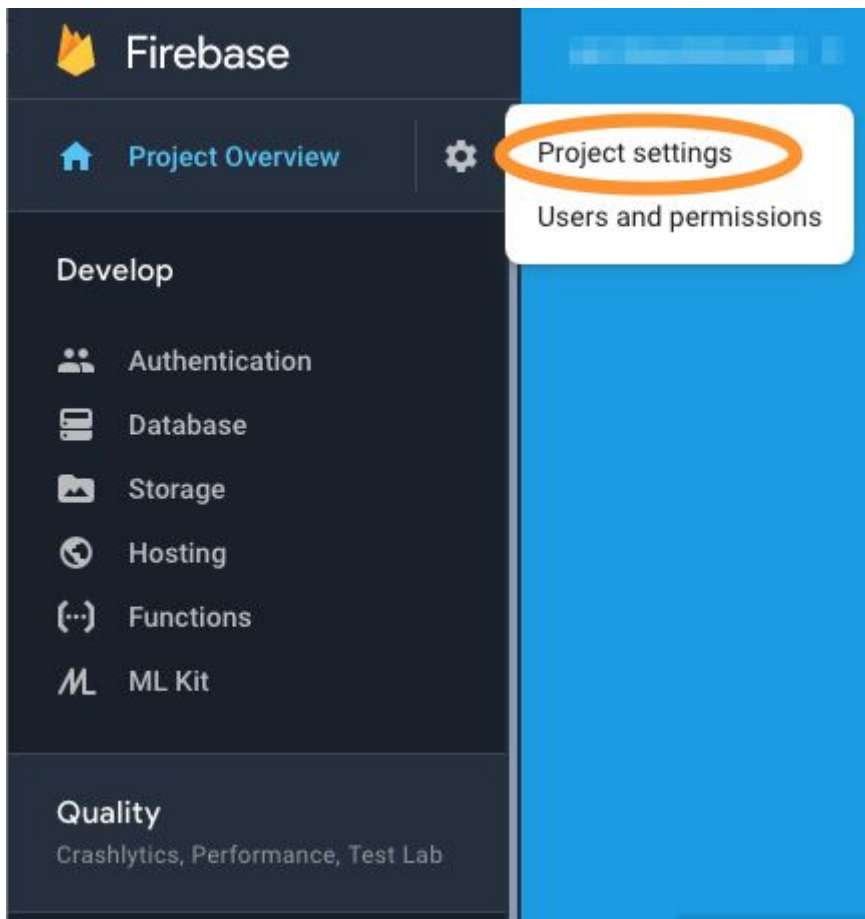
Run your app to verify installation

5. Download **google-services.json**. Store it somewhere. Close the configuration dialog (you don't have to press **Next** button).



6. Open Firebase Console: <https://console.firebase.google.com> and choose the project you've just created/configured.

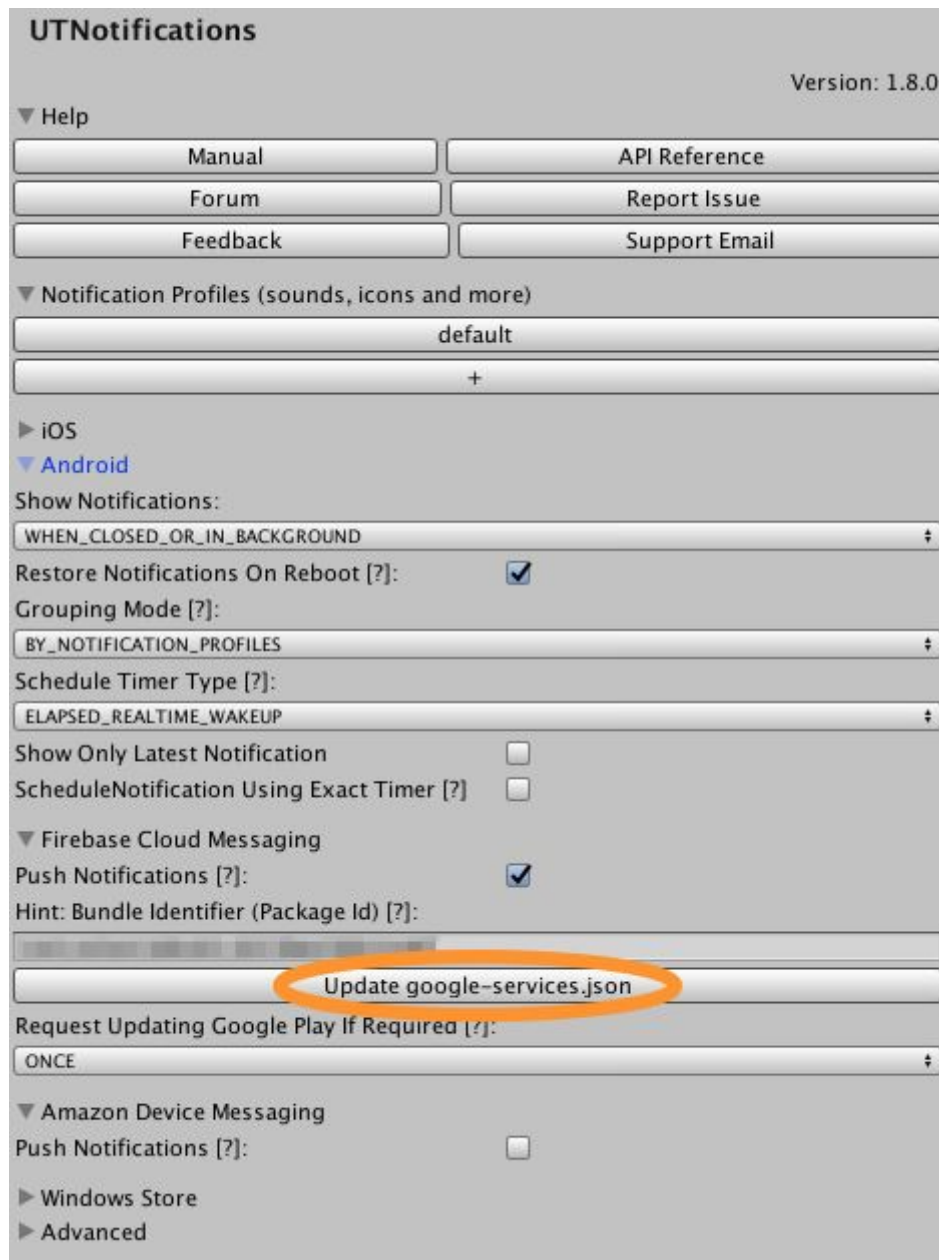
7. Open the project settings and switch to tab **CLOUD MESSAGING**.



8. Copy and store somewhere the value of Server key.

Apply Credentials and Test

1. In Unity open the UINotifications Settings in menu: Edit -> Project Settings -> UINotifications (Unity restart may be required to see this menu item first time) and enable **Push Notifications** toggle in the **Firebase Cloud Messaging**.
2. In **Firebase Play Settings** press **Load google-services.json** button to load and apply the configuration file you obtained previously.



3. Specify the credentials in *Assets/UTNotifications/Editor/DemoServer/src/main/java/com/universal_tools/demoserver/PushNotificator.java*:

```

/// <summary>
/// The sample class showing how you can send push notifications for different "providers", such as APNS, FCM, ADM and WNS.
/// </summary>
public class PushNotificator {
    // private
    // Please provide the required values. Find more details in the manual: Assets/UTNotifications/Documentation/Manual.pdf
    private static final String FIREBASE_SERVER_KEY = "";
    private static final String AMAZON_CLIENT_ID = null;
    private static final String AMAZON_CLIENT_SECRET = null;
    private static final String APNS_AUTH_KEY = null;
    private static final String APNS_TEAM_ID = null;
    private static final String APNS_KEY_ID = null;
    private static final String APNS_BUNDLE_ID = null;
    private static final boolean APNS_DEVELOPMENT = true;
    private static final String WINDOWS_PACKAGE_SID = null;
    private static final String WINDOWS_CLIENT_SECRET = null;

```

- **FIREBASE_SERVER_KEY**: The value of server key you've copied from Firebase Console previously.

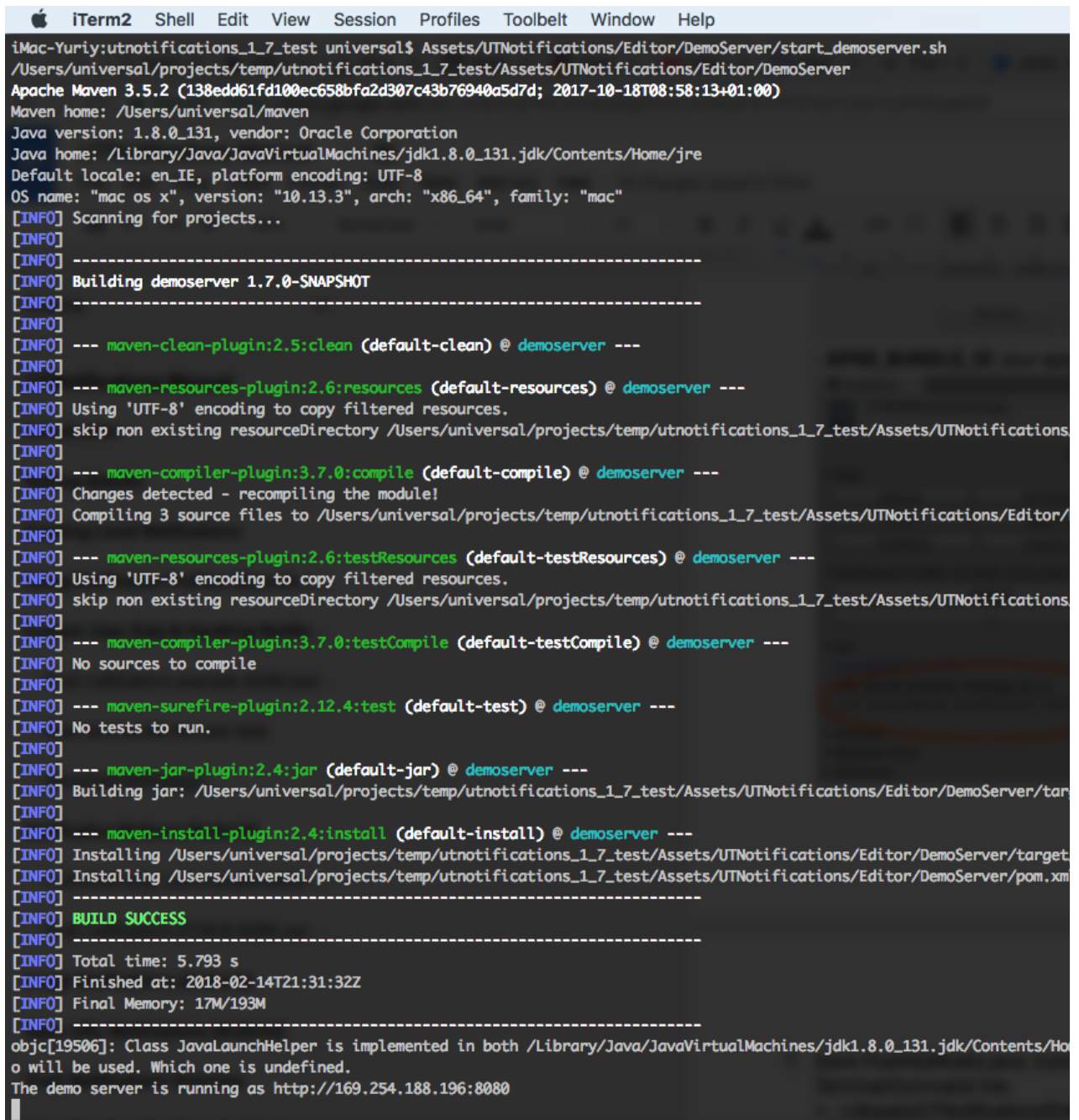
4. Save *PushNotificator.java*, build and start DemoServer, by executing the following script in Terminal/Command line:

<...>/Assets/UTNotifications/Editor/DemoServer/start_demoserver.sh (macOS / Linux)

or

<...>\Assets\UTNotifications\Editor\DemoServer\start_demoserver.bat (Windows)

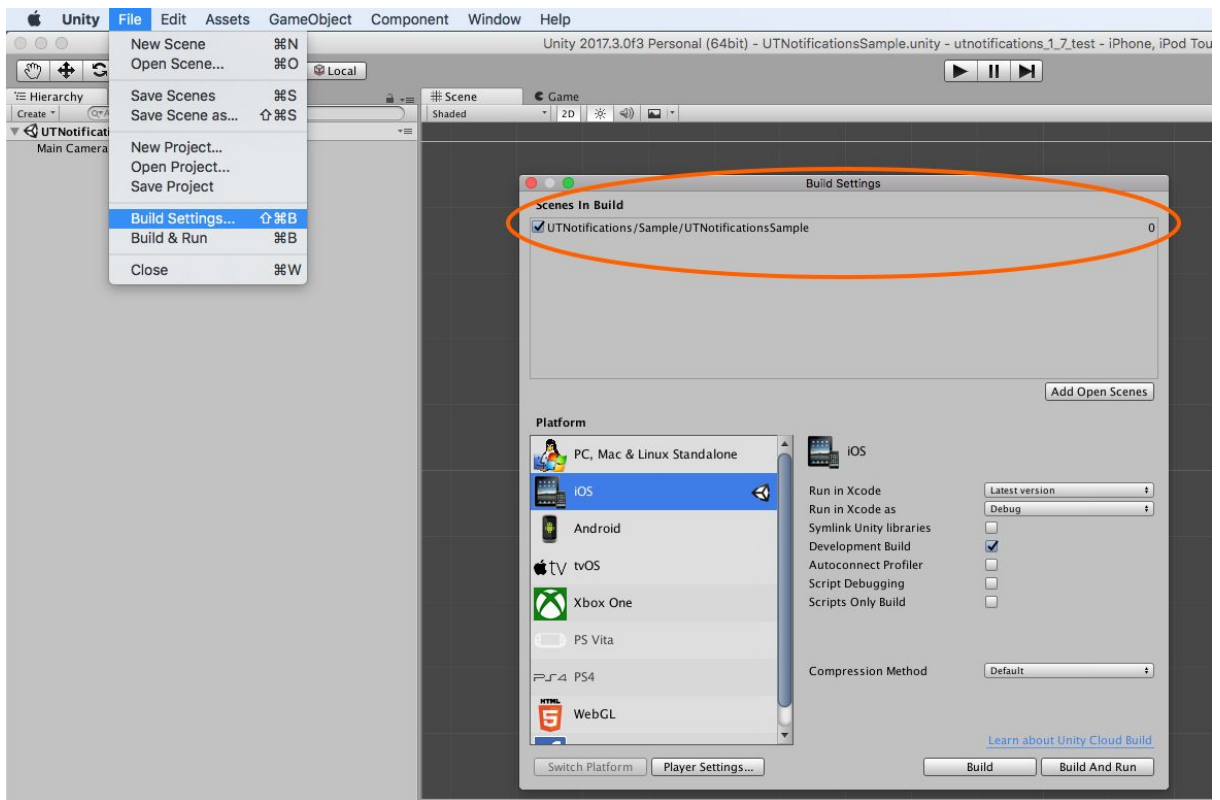
Note that you'll need [JDK](#) and [Maven](#).



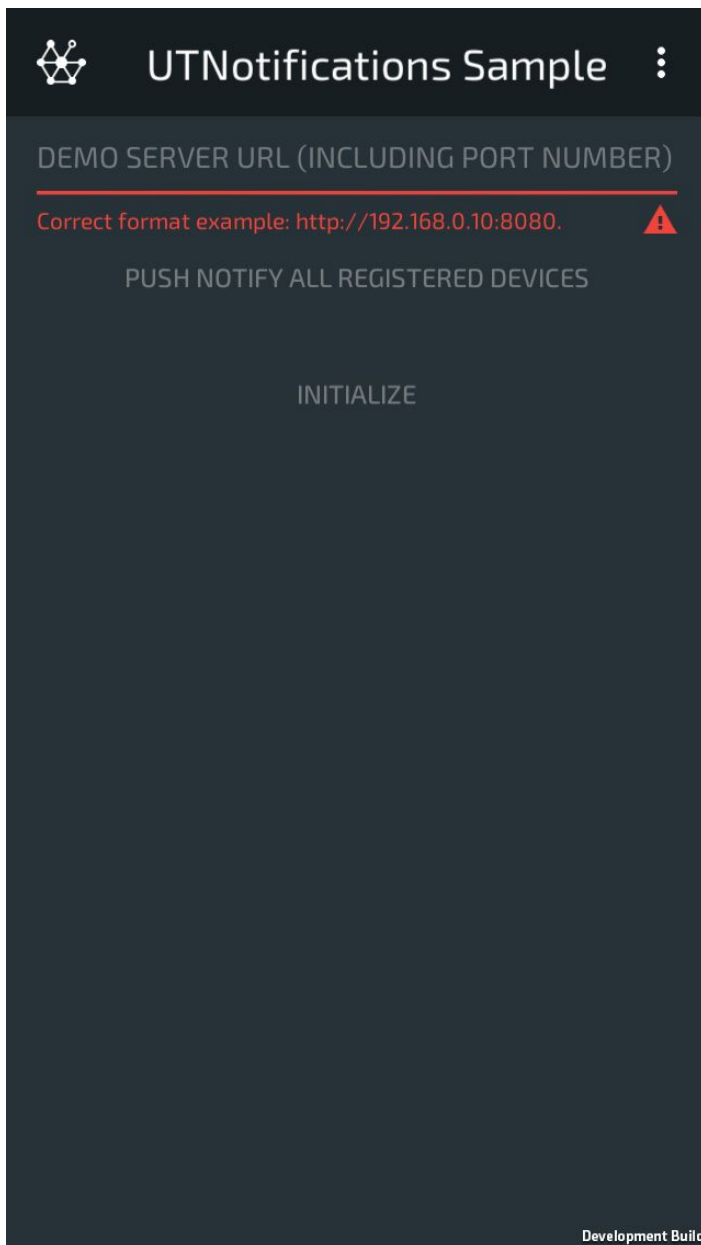
```
iTerm2 Shell Edit View Session Profiles Toolbelt Window Help
iMac-Yuriy:utnotifications_1_7_test universal$ Assets/UTNotifications/Editor/DemoServer/start_demoserver.sh
/Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications/Editor/DemoServer
Apache Maven 3.5.2 (138ed61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T08:58:13+01:00)
Maven home: /Users/universal/maven
Java version: 1.8.0_131, vendor: Oracle Corporation
Java home: /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/jre
Default locale: en_IE, platform encoding: UTF-8
OS name: "mac os x", version: "10.13.3", arch: "x86_64", family: "mac"
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building demoserver 1.7.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ demoserver ---
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ demoserver ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications
[INFO]
[INFO] --- maven-compiler-plugin:3.7.0:compile (default-compile) @ demoserver ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 3 source files to /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications/Editor/
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ demoserver ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications
[INFO]
[INFO] --- maven-compiler-plugin:3.7.0:testCompile (default-testCompile) @ demoserver ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ demoserver ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ demoserver ---
[INFO] Building jar: /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications/Editor/DemoServer/target
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ demoserver ---
[INFO] Installing /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications/Editor/DemoServer/target
[INFO] Installing /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications/Editor/DemoServer/pom.xml
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] -----
[INFO] Total time: 5.793 s
[INFO] Finished at: 2018-02-14T21:31:32Z
[INFO] Final Memory: 17M/193M
[INFO]
[INFO] -----
objc[19506]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Ho
o will be used. Which one is undefined.
The demo server is running as http://169.254.188.196:8080
```

5. The running DemoServer will print its hostname (ip address) and port. Please note, that the ip address it prints can be either a local network address (usually 192.168.*.*) or an external (like on the screenshot above). In later case, please find your internal IP address in the OS network settings.

6. Temporarily make UTNotifications/Sample/UTNotificationsSample scene default for your Unity project:

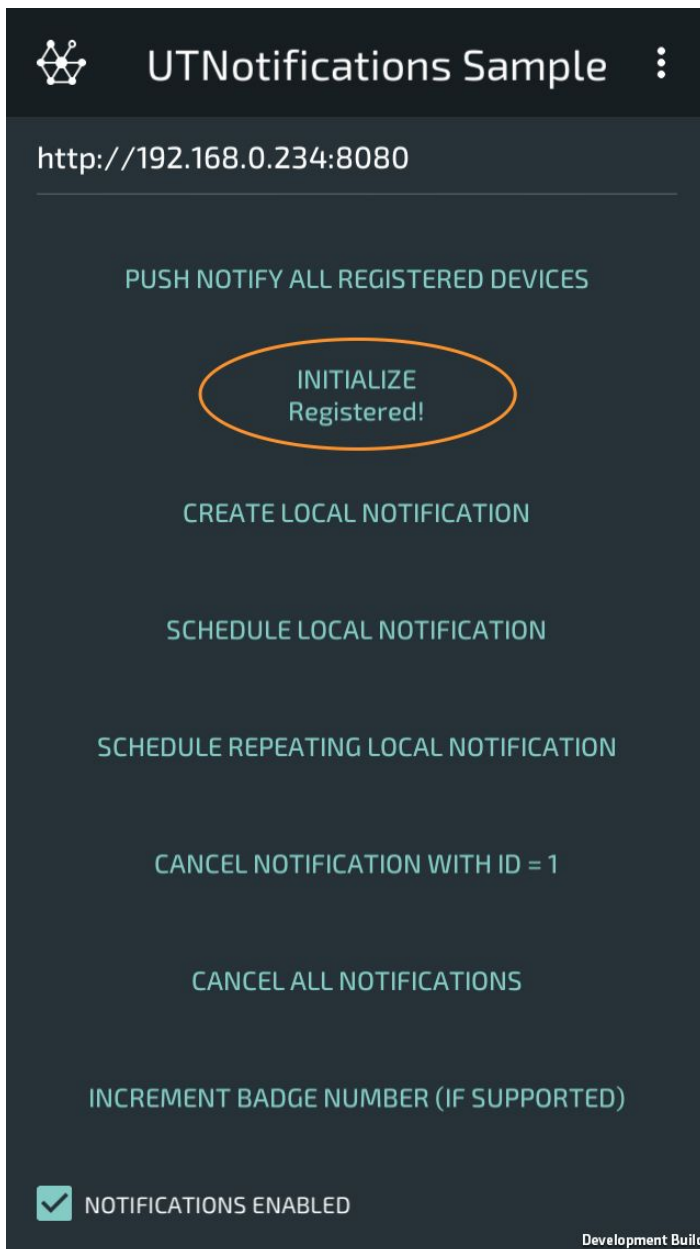


7. Build and run the application in a target Android device. The sample scene should start requesting an URL of DemoServer in order to continue:



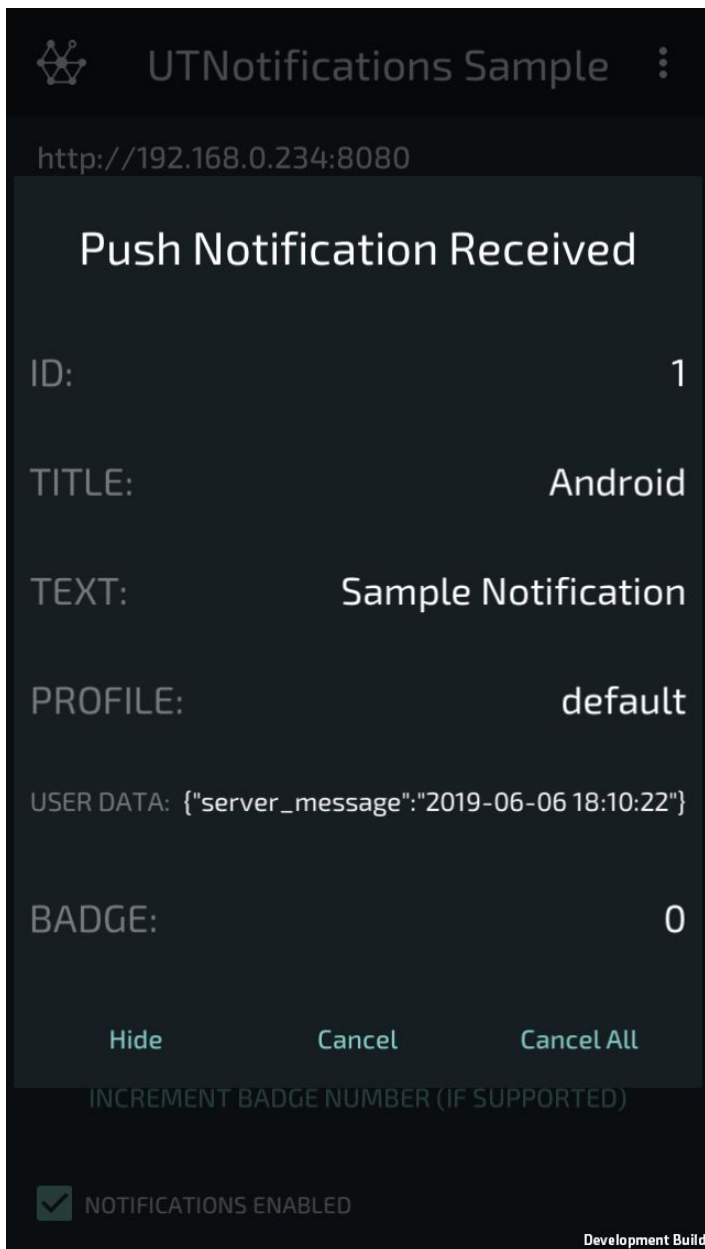
8. Make sure your test device and the computer running DemoServer belong to the same local network (f.e. connected to the same Wi-Fi router). Specify the full URL of the running DemoServer (as **http://<ip address>:8080**) and press **INITIALIZE**:

```
<< /register uid=DB058E75-958D-4707-9E9D-DD86C867B219&provider=APNS&id=c0cbc62f080d528c54cafa7c
>> HTTP/1.1 200 OK
>> Server: UTNotificationsDemoServer
>> Content-Type: text/html
>> Content-Length: 11
>> Connection: close
>>
>> Registered!
```



9. Press **PUSH NOTIFY ALL REGISTERED DEVICES** to send a push notification to all the DemoServer-registered devices. If everything was configured correctly, you should see how

UTNotifications SampleScene handled the push message:



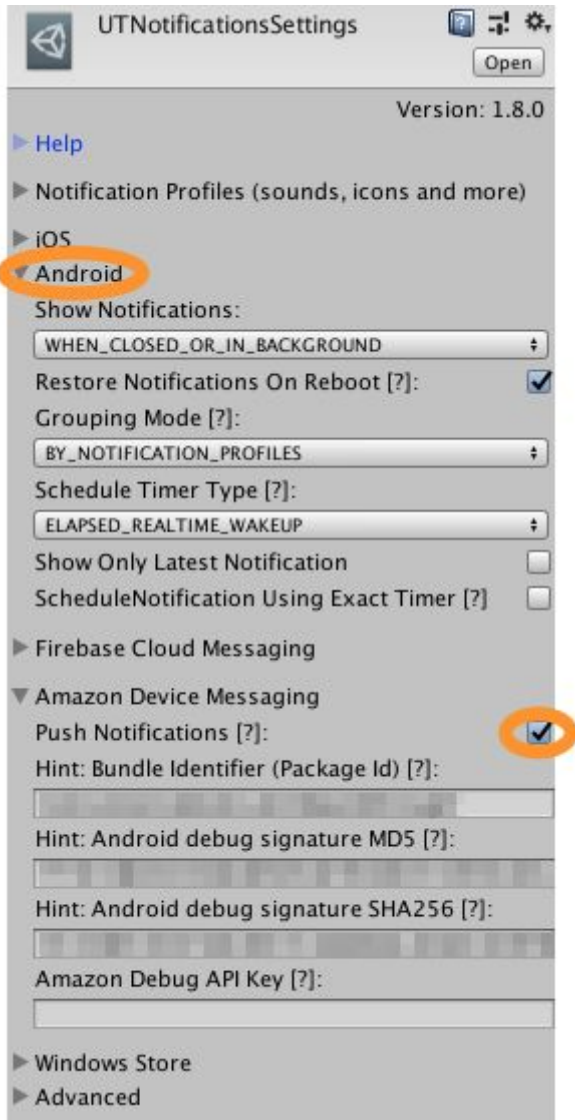
Configuring the Amazon Device Messaging (ADM)

Based on ADM official documentation:

<https://developer.amazon.com/public/apis/engage/device-messaging/tech-docs/02-obtaining-adm-credentials>


Getting Your OAuth Credentials and API Key

1. In UTNotifications Settings (*Edit -> Project Settings -> UTNotifications*) enable Android -> Amazon Device Messaging toggle:



2. Create an account on the [Amazon Apps & Games Developer Portal](https://developer.amazon.com) and add your app, if you have not already done so.
3. In **Apps & Services > My Apps**, select the app with which you want to use ADM or create a new one.

4. Locate **App Services > Device Messaging**.



Device Messaging

Amazon Device Messaging (ADM) lets you send messages to Amazon devices that run your app, so you app, so you keep user up to date and involved. Whether you're giving a user a game update or letting them know that a message from their buddy has arrived, ADM helps you stay in touch.

Get Started

You need a security profile to identify your App. Your security profile credentials, client ID and client secret - allows your device to securely identify itself to the services. For more details check the technical documentation on how to create it.

✓ Select existing security profile or create new

Security Profile

- Select Security Profile -

Enable Security Profile

OR


Create Security Profile

5. To assign a security profile to your app, choose an existing security profile from **Select Profile** or click **Create Security Profile**. A security profile provides the OAuth credentials that you use when sending messages with ADM.

Note: You can share the use of a security profile with more than one app. Sharing a profile allows apps to share some types of data. For example, you may have a "My Cat - Free" app and a "My Cat - HD" app. If you apply a single security profile to both apps, data accessed by that profile is available to both apps. For a shared profile, choose a name that applies to both, for example, "My Cat Apps profile".

6. Click **Enable Device Messaging** button after assigning a security profile to the app.

7. Click **View Security Profile**.



Device Messaging

Amazon Device Messaging (ADM) lets you send messages to Amazon devices that run your app, so you app, so you keep user up to date and involved. Whether you're giving a user a game update or letting them know that a message from their buddy has arrived, ADM helps you stay in touch.

Get Started

You need a security profile to identify your App. Your security profile credentials, client ID and client secret - allows your device to securely identify itself to the services. For more details check the technical documentation on how to create it.

✓ Device messaging was successfully enabled on your app using "utn_1_8_workthrough" security profile

✓ Attached security profile and related details

Security Profile Name

Security Profile Description

Security Profile ID

View Security Profile ↗

8. Store somewhere the **Client ID** and **Client Secret** values from the **Web Settings** tab.

Security Profile Management

← [Profile Name] - Security Profile

General

Web Settings

Android/Kindle Settings

iOS Settings

TVs and Other Devices Settings

To use Login with Amazon with a website, you must specify either an allowed JavaScript origin (for the Implicit grant) or an allowed return URL (for the Authorization Code grant). [Learn More](#)

Client ID

Client Secret

Allowed Origins ?

Allowed Return URLs ?

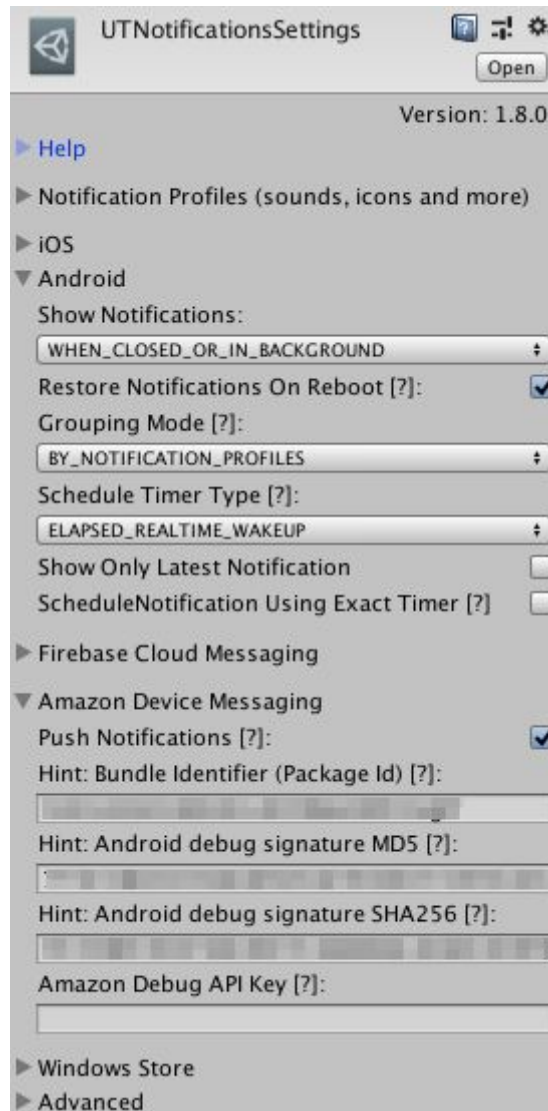
9. Then click the **Android/Kindle Settings** tab.
10. Create an **API Key**. Your app requires one or more API Keys.
 - **(Required)** For a pre-release or "debug" version of your app. In all cases, you must create an API Key for the debug version of your app, in order to test your app with ADM.
 - **(Optional)** For a release or "production" version of your app. If you sign the release version of your app using your own certificate, you must create an additional API Key for the release version of your app. If you allow Amazon to sign your app on your behalf, you do not need to create an additional API Key.

To create an API Key, you must provide both the package name (for example, `com.mycompany.bestapplication`) for the app and its signature:

- **Debug** application signature for the pre-release version of your app.
 - a. In Unity open the UTNotifications Settings in menu: `Edit -> Project Settings -> UTNotifications` (Unity restart may be required to see this menu item first time) and enable **Push Notifications** toggle in the **Amazon Device Messaging**.
 - b. Copy and paste the **Package Name**, **Android debug signature MD5** and **Android debug signature SHA256** hints from **UTNotifications Settings / Android / Amazon Device Messaging** to the Amazon **Security Profile** fields **Package**, **MD5 Signature** and **SHA256 Signature** appropriately.

Note: *If you don't see the **Android debug signature MD5** hint value please build the Android version at least once successfully. If getting the **Android debug signature MD5** is still failed after that, please see*

<https://developer.amazon.com/public/apis/engage/device-messaging/tech-docs/02-obtaining-adm-credentials>.



- **Release** application signature for the production version of your app. If you sign the release version of your app using your own certificate, provide the MD5 signature for that certificate to create an additional API Key (more details at <https://developer.amazon.com/public/apis/engage/device-messaging/tech-docs/02-obtaining-adm-credentials>). If you allow Amazon to sign your app on your behalf, you do not need to obtain an API Key for the release signature.

11. Click **Generate New Key**.

Security Profile Management

Security Profile

General Web Settings **Android/Kindle Settings** iOS Settings TVs and Other Devices Settings

An API Key allows Amazon to verify your app's identity. An API Key is generated based on the values you provide below. If different versions of your app have different signatures or package names, such as for one or more testing versions and a production version, each version requires its own API Key. [Learn More](#)

API Key Name *
Identifies the app you will use with this API key.

Package *
The package name of your Android project. For example, com.mycompany.bestapp.

MD5 Signature *
The MD5 signature of the certificate used to sign your app. [Get instructions on obtaining this value.](#)

SHA256 Signature *
The SHA256 signature of the certificate used to sign your app. [Get instructions on obtaining this value.](#)

Generate New Key

12. Store the retrieved **API Key** somewhere.

Note: It shouldn't contain any spaces or newline characters.

API Key Details

API Key Name *
Identifies the app you will use with this API key.

Package *
The package name of your Android project. For example, com.mycompany.bestapp.

MD5 Signature *
The MD5 signature of the certificate used to sign your app. [Get instructions on obtaining this value.](#)

SHA256 Signature *
The SHA256 signature of the certificate used to sign your app. [Get instructions on obtaining this value.](#)

API key *

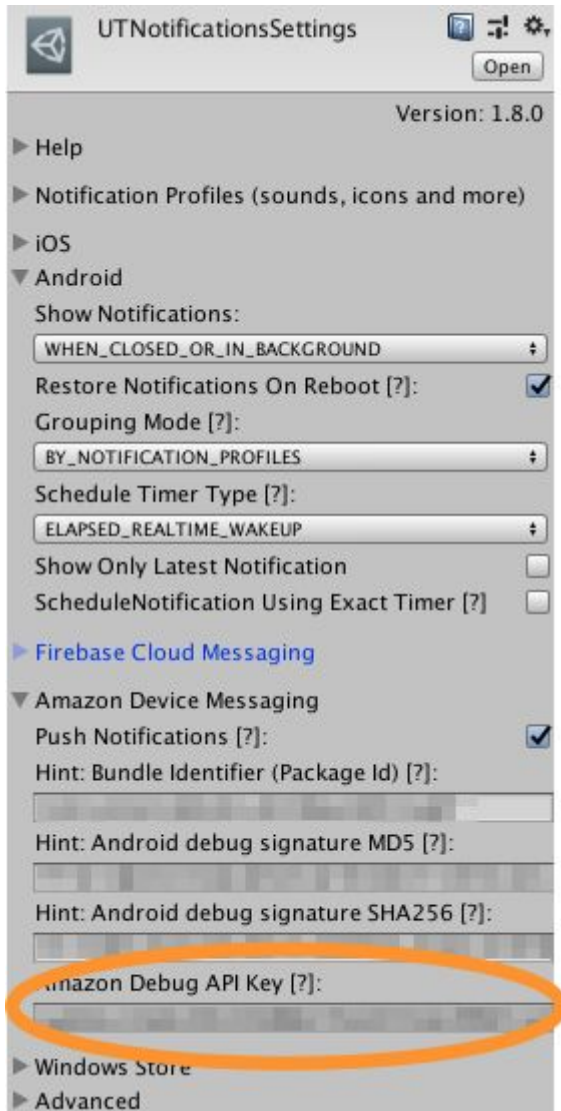
API Key

Show

Add an API Key

Apply Credentials and Test

1. Specify the **API Key** from step 12 of [Getting Your OAuth Credentials and API Key](#) as the value for **Amazon Debug API Key** in UTNotifications settings in Unity:



- Specify the credentials in `Assets/UTNotifications/Editor/DemoServer/src/main/java/com/universal_tools/demoserver/PushNotificator.java`:

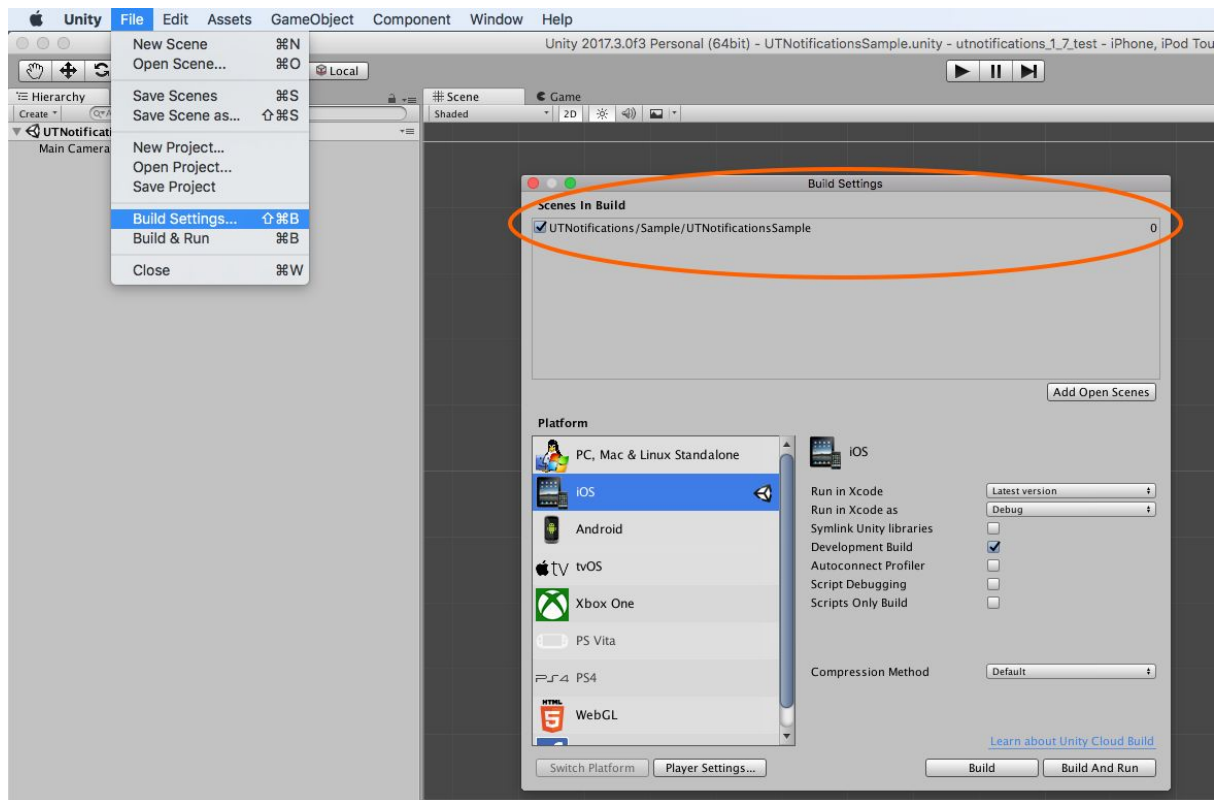
[illegible]

- **AMAZON_CLIENT_ID**: Client ID value you've got in step 8 of [Getting Your OAuth Credentials and API Key](#).
 - **AMAZON_CLIENT_SECRET**: Client Secret value you've got in step 8 of [Getting Your OAuth Credentials and API Key](#).
3. Save *PushNotificator.java*, build and start DemoServer, by executing the following script in Terminal/Command line:
- <...>/Assets/UTNotifications/Editor/DemoServer/start_demoserver.sh (macOS / Linux)
- or
- <...>\Assets\UTNotifications\Editor\DemoServer\start_demoserver.bat (Windows)

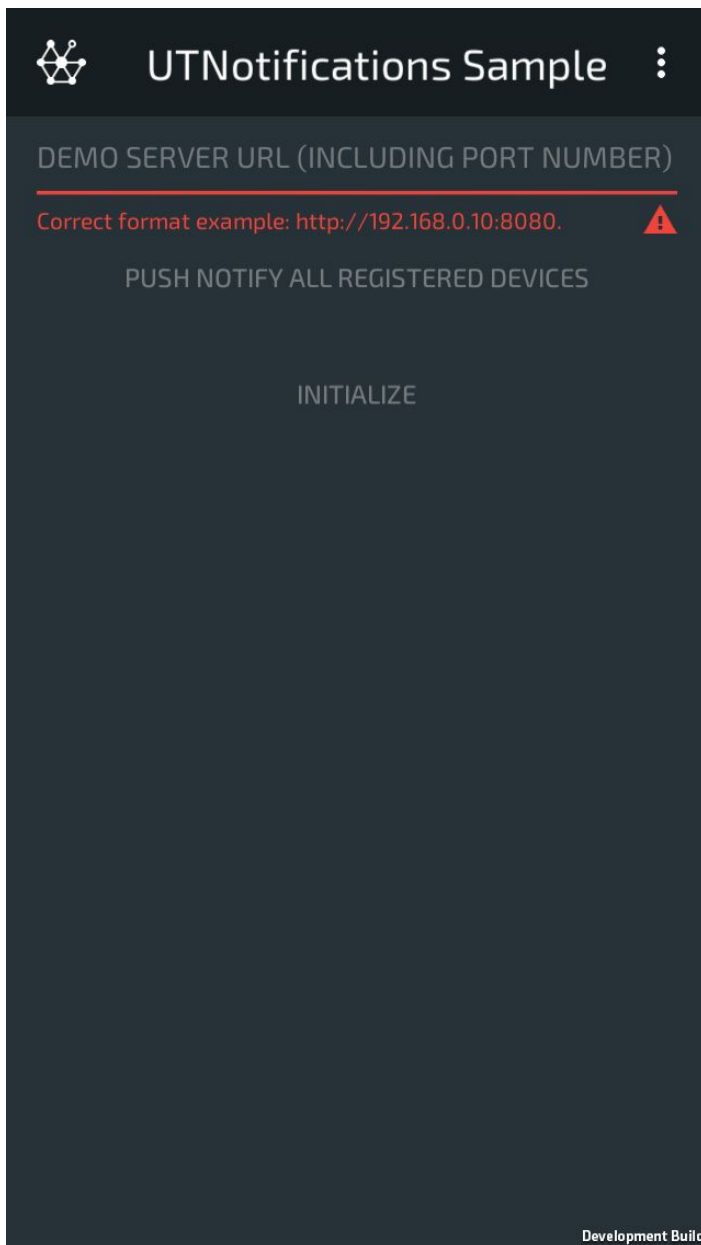
Note that you'll need [JDK](#) and [Maven](#).

```
iTerm2  Shell  Edit  View  Session  Profiles  Toolbelt  Window  Help
iMac-Yuriy:utnotifications_1_7_test universal$ Assets/UTNotifications/Editor/DemoServer/start_demoserver.sh
/Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications/Editor/DemoServer
Apache Maven 3.5.2 (138ed61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T08:58:13+01:00)
Maven home: /Users/universal/maven
Java version: 1.8.0_131, vendor: Oracle Corporation
Java home: /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/jre
Default locale: en_IE, platform encoding: UTF-8
OS name: "mac os x", version: "10.13.3", arch: "x86_64", family: "mac"
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building demoserver 1.7.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ demoserver ---
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ demoserver ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications
[INFO]
[INFO] --- maven-compiler-plugin:3.7.0:compile (default-compile) @ demoserver ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 3 source files to /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications/Editor/
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ demoserver ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications
[INFO]
[INFO] --- maven-compiler-plugin:3.7.0:testCompile (default-testCompile) @ demoserver ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ demoserver ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ demoserver ---
[INFO] Building jar: /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications/Editor/DemoServer/tar
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ demoserver ---
[INFO] Installing /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications/Editor/DemoServer/target
[INFO] Installing /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications/Editor/DemoServer/pom.xml
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO]
[INFO] Total time: 5.793 s
[INFO] Finished at: 2018-02-14T21:31:32Z
[INFO] Final Memory: 17M/193M
[INFO] -----
objc[19506]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Ho
o will be used. Which one is undefined.
The demo server is running as http://169.254.188.196:8080
```

4. The running DemoServer will print its hostname (ip address) and port. Please note, that the ip address it prints can be either a local network address (usually 192.168.*.*) or an external (like on the screenshot above). In later case, please find your internal IP address in the OS network settings.
5. Temporarily make UTNotifications/Sample/UTNotificationsSample scene default for your Unity project:

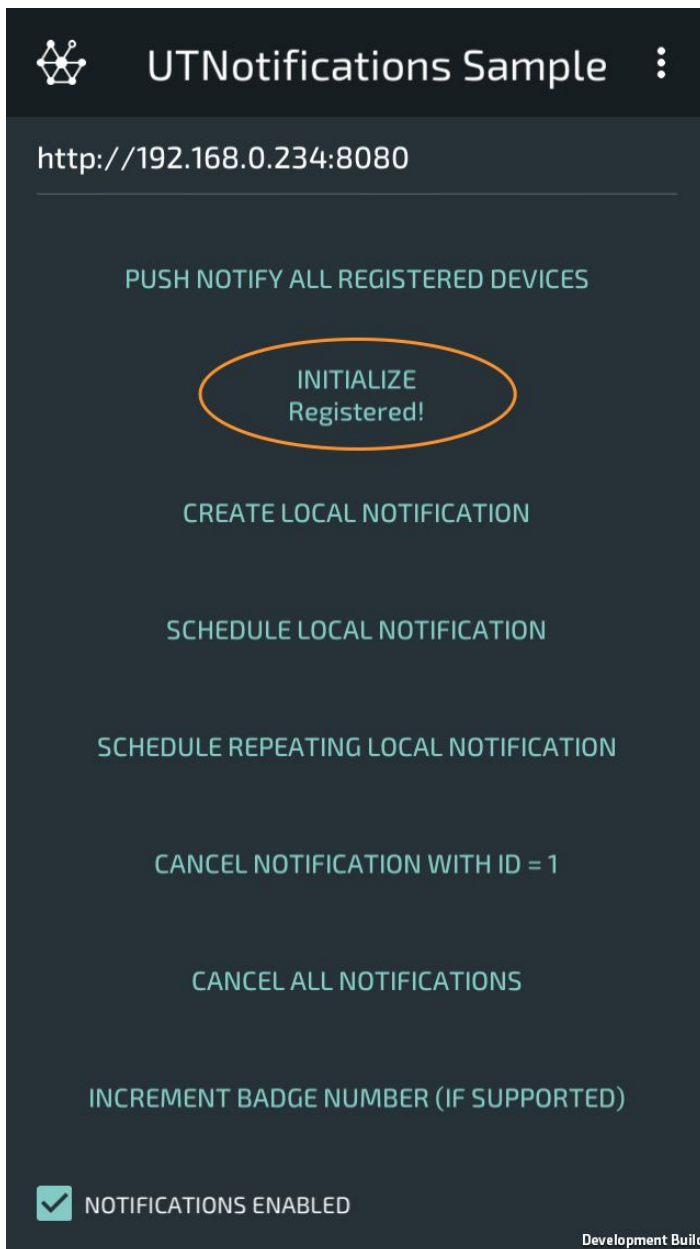


6. Build and run the application in a target Kindle Fire device. The sample scene should start requesting an URL of DemoServer in order to continue:



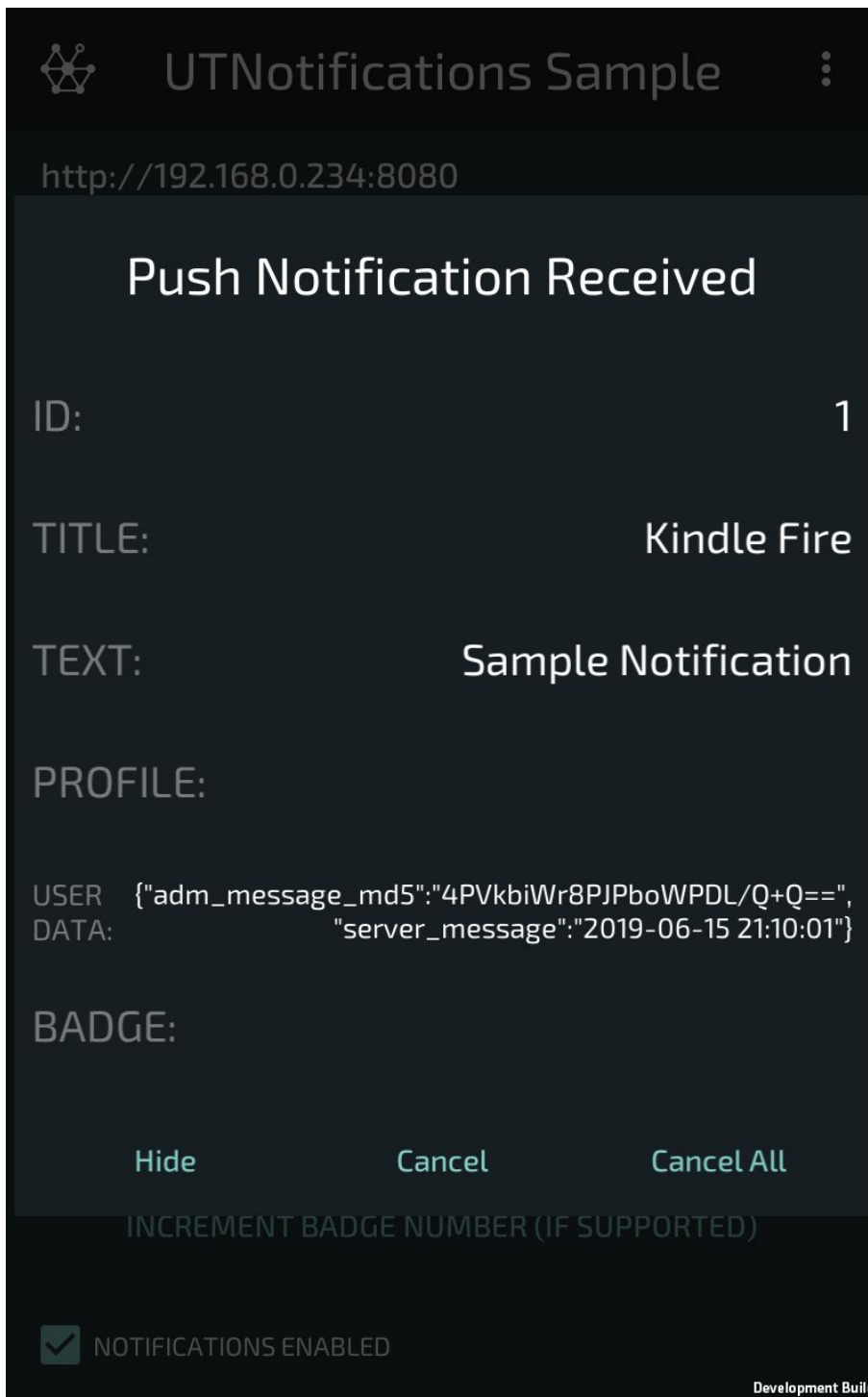
7. Make sure your test device and the computer running DemoServer belong to the same local network (f.e. connected to the same Wi-Fi router). Specify the full URL of the running DemoServer (as **http://<ip address>:8080**) and press **INITIALIZE**:

```
<< /register uid=DB058E75-958D-4707-9E9D-DD86C867B219&provider=APNS&id=c0cbc62f080d528c54cafa7c
>> HTTP/1.1 200 OK
>> Server: UTNotificationsDemoServer
>> Content-Type: text/html
>> Content-Length: 11
>> Connection: close
>>
>> Registered!
```

8. Press **PUSH NOTIFY ALL REGISTERED DEVICES** to send a push notification to all the DemoServer-registered devices. If everything was configured correctly, you should see how

UTNotifications SampleScene handled the push message:



Configuring the Windows Push Notification Services (WNS)

Based on WNS official documentation:

<https://msdn.microsoft.com/en-us/library/windows/apps/hh465407.aspx>.

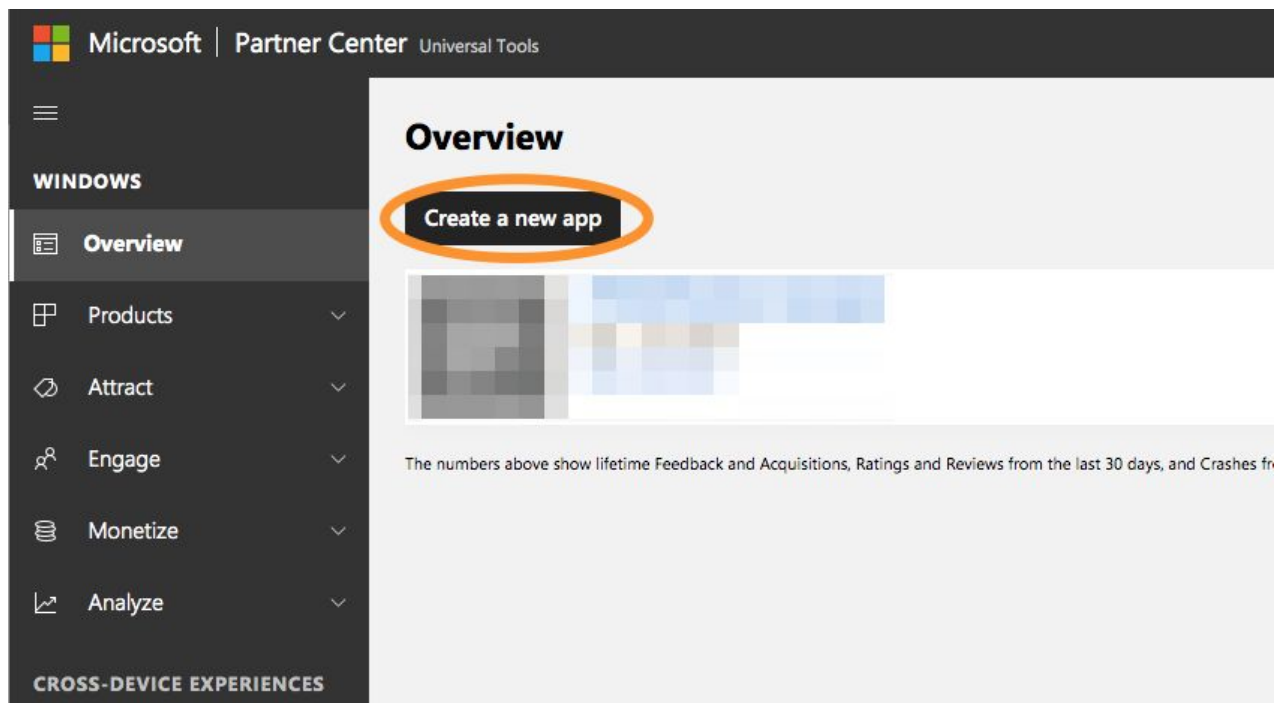
Register your app with the Dashboard

Before you can send notifications through WNS, you must register your app. Do so through the [Dashboard](#), the developer portal that enables you to submit, certify, and manage your Windows Store

apps. When you register your app through the Dashboard, you are given credentials—a Package security identifier (SID) and a secret key—which your cloud service uses to authenticate itself with WNS.

To register:

1. Go to the [Windows Store apps page](#) of the Windows Dev Center and sign in with your Microsoft account.
*Please **note**, that you need a valid Windows developer account in order to proceed.*
2. Once you have signed in, click the [Dashboard](#) link.
3. On the Dashboard, click **Create a new app**.

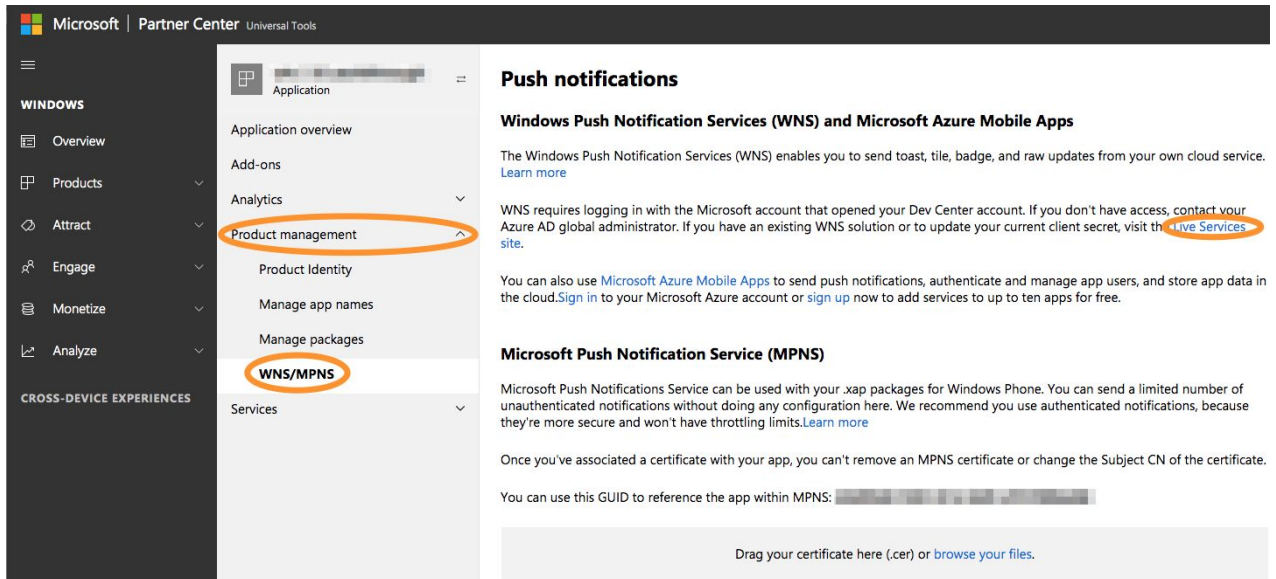


4. Choose a name and click “Reserve product name” to register an app.

Obtain the identity values for your app

When you reserved a name for your app, the Windows Store created your associated credentials. It also assigned associated identity values—name and publisher.

1. Click at **Product Management** -> **WNS/MPNS** in the left menu.
2. Press on a link **Live Services** site.



3. Save somewhere the following values: **Application Secret**, **Package SID**, **Identity Name** & **Publisher**.

My applications / [redacted]



Registration

[Click here for help integrating your application with Microsoft.](#)

Properties

Name

[redacted]

Application Id

[redacted]

Application Secrets

[Generate New Password](#)

[redacted]

Version 0

Current

Windows Store

Package SID

This is the unique identifier for your Windows Store app.

ms-app://

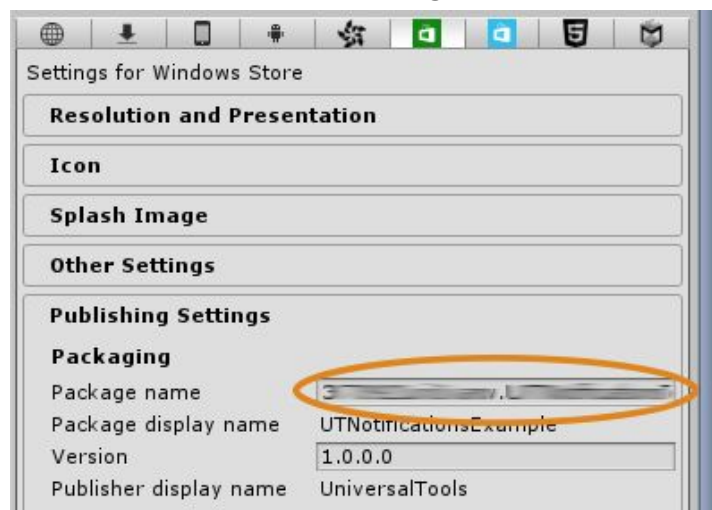
Application Identity

To set your application's identity values manually, open the AppManifest.xml file in a text editor and set these attributes of the <identity> element using the values shown here.

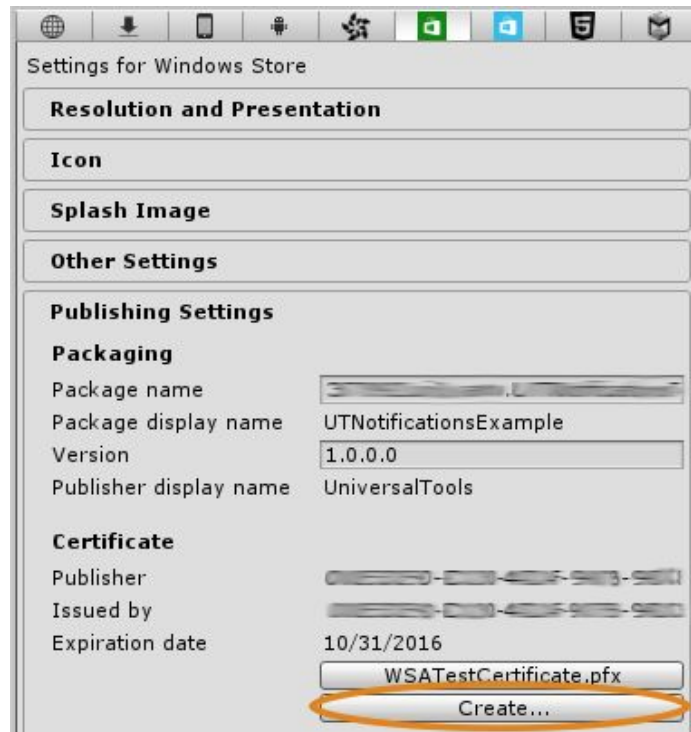
<Identity Name=[redacted] Publisher='CN=[redacted]' />

[Link to different app](#)

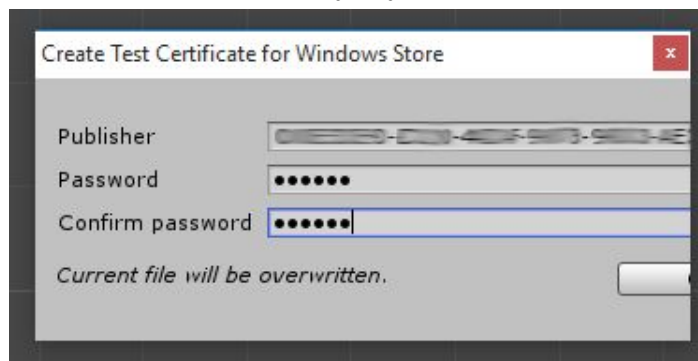
4. In Unity open the UTNotifications Settings in menu: Edit -> Project Settings -> UTNotifications (Unity restart may be required to see this menu item first time) and enable **Push Notifications** toggle in the **Windows Store Settings**.
5. Open Windows Store player settings: File -> Build Settings... -> Windows Store -> Player Settings.
6. Use **Identity Name** value from 3rd step as **Package Name**.



7. Press Create button to create a certificate.



8. Use **Publisher** from the 3rd step for **Publisher**. Don't include starting **CN=** to this value, only the rest. Note, that at least the certificate creation dialog in Unity is buggy (it's not optimized for such a long values of Publisher). It works fine anyway.



Apply Credentials and Test

1. Specify the credentials in `Assets/UTNotifications/Editor/DemoServer/src/main/java/com/universal_tools/demoserver/PushNotificator.java`:

```

/// <summary>
/// The sample class showing how you can send push notifications for different "providers", such as APNS, FCM, ADM and WNS.
/// </summary>
public class PushNotificator {
// private
    // Please provide the required values. Find more details in the manual: Assets/UTNotifications/Documentation/Manual.pdf
    private static final String FIREBASE_SERVER_KEY = null;
    private static final String AMAZON_CLIENT_ID = null;
    private static final String AMAZON_CLIENT_SECRET = null;
    private static final String APNS_AUTH_KEY = null;
    private static final String APNS_TEAM_ID = null;
    private static final String APNS_KEY_ID = null;
    private static final String APNS_BUNDLE_ID = null;
    private static final boolean APNS_DEVELOPMENT = true;
    private static final String WINDOWS_PACKAGE_SID = "ms-app://";
    private static final String WINDOWS_CLIENT_SECRET = "";

```

- **WINDOWS_PACKAGE_SID**: Package SID you got in 3rd step of [Obtain the identity values for your app](#) section.

- **WINDOWS_CLIENT_SECRET**: Client secret you got in 3rd step of [Obtain the identity values for your app](#) section.

2. Save *PushNotificator.java*, build and start DemoServer, by executing the following script in Terminal/Command line:

<...>/Assets/UTNotifications/Editor/DemoServer/start_demoserver.sh (macOS / Linux)

or

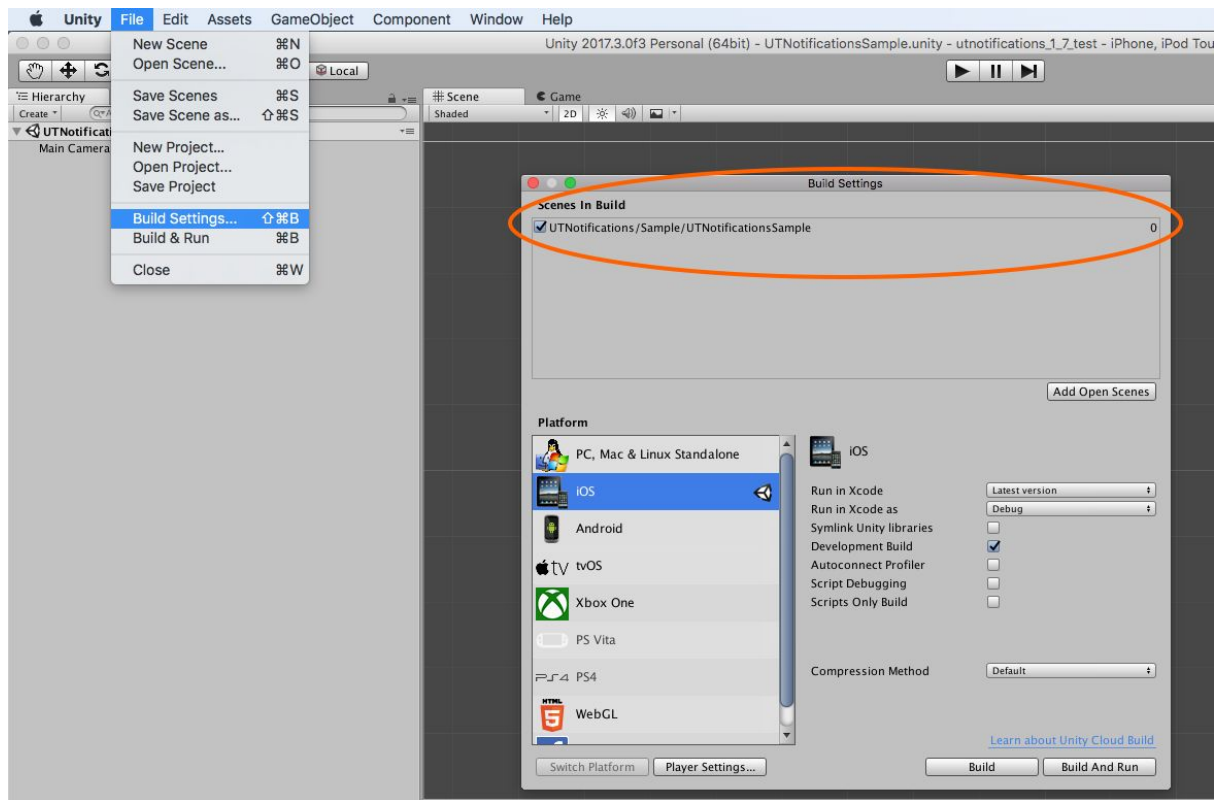
<...>\Assets\UTNotifications\Editor\DemoServer\start_demoserver.bat (Windows)

Note that you'll need [JDK](#) and [Maven](#).

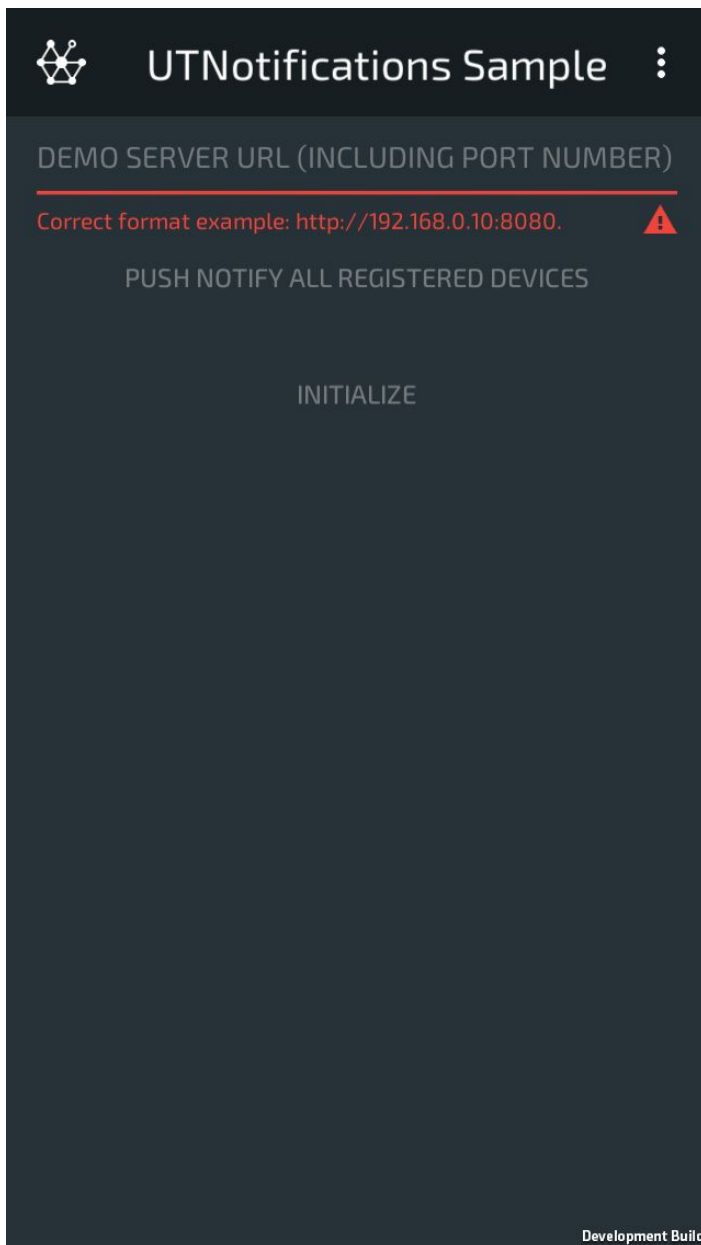

```
iTerm2  Shell  Edit  View  Session  Profiles  Toolbelt  Window  Help
iMac-Yuriy:utnotifications_1_7_test universal$ Assets/UTNotifications/Editor/DemoServer/start_demoServer.sh
/Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications/Editor/DemoServer
Apache Maven 3.5.2 (138eddd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T08:58:13+01:00)
Maven home: /Users/universal/maven
Java version: 1.8.0_131, vendor: Oracle Corporation
Java home: /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/jre
Default locale: en_IE, platform encoding: UTF-8
OS name: "mac os x", version: "10.13.3", arch: "x86_64", family: "mac"
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building demoServer 1.7.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ demoServer ---
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ demoServer ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications
[INFO]
[INFO] --- maven-compiler-plugin:3.7.0:compile (default-compile) @ demoServer ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 3 source files to /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications/Editor/
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ demoServer ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications
[INFO]
[INFO] --- maven-compiler-plugin:3.7.0:testCompile (default-testCompile) @ demoServer ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ demoServer ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ demoServer ---
[INFO] Building jar: /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications/Editor/DemoServer/target
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ demoServer ---
[INFO] Installing /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications/Editor/DemoServer/target
[INFO] Installing /Users/universal/projects/temp/utnotifications_1_7_test/Assets/UTNotifications/Editor/DemoServer/pom.xml
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 5.793 s
[INFO] Finished at: 2018-02-14T21:31:32Z
[INFO] Final Memory: 17M/193M
[INFO] -----
objc[19506]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Ho
o will be used. Which one is undefined.
The demo server is running as http://169.254.188.196:8080
```

3. The running DemoServer will print its hostname (ip address) and port. Please note, that the ip address it prints can be either a local network address (usually 192.168.*.*) or an external (like on the screenshot above). In later case, please find your internal IP address in the OS network settings.

4. Temporarily make UTNotifications/Sample/UTNotificationsSample scene default for your Unity project:

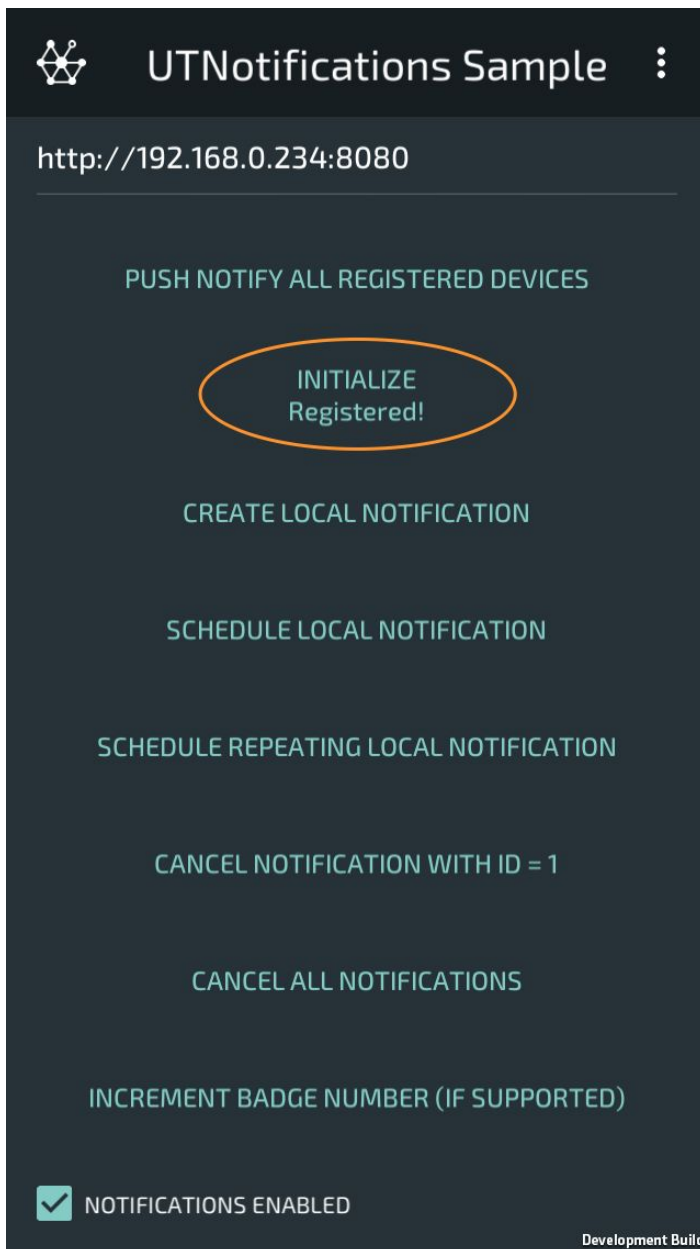


5. Build and run the application in a target UWP device (a Windows 8.1+ computer / Windows Phone). The sample scene should start requesting an URL of DemoServer in order to continue:



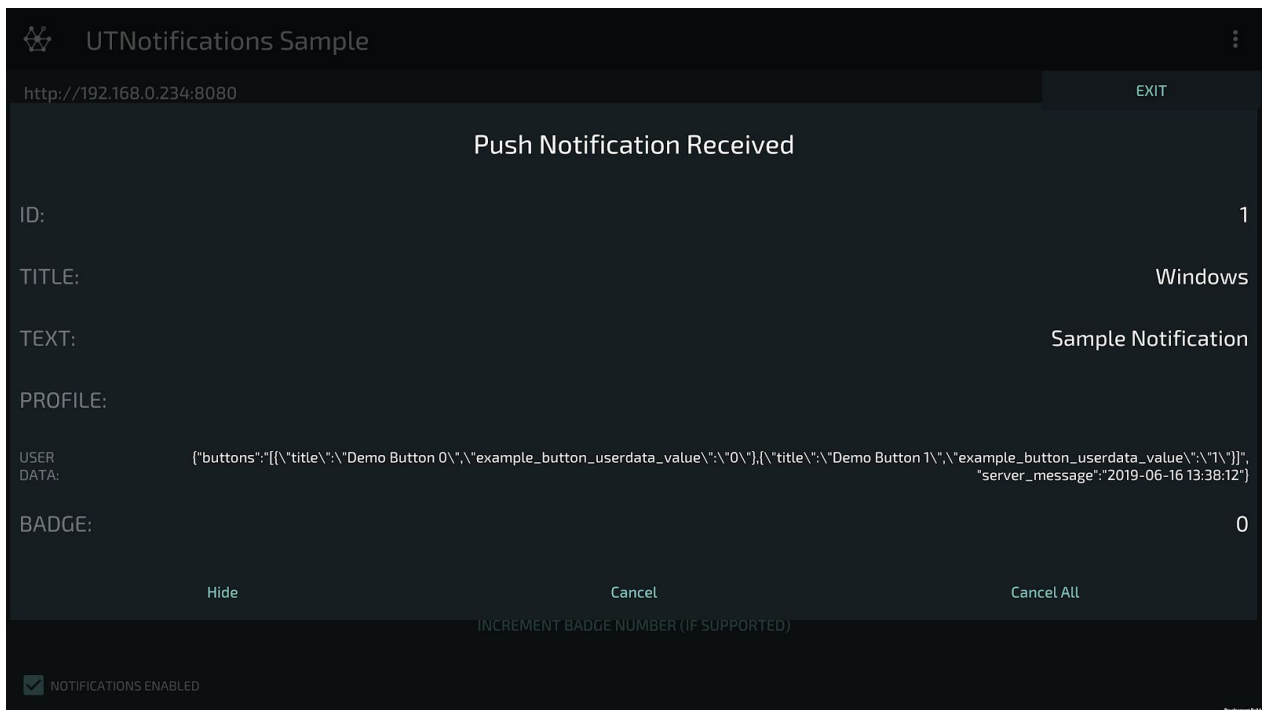
6. Make sure your test device and the computer running DemoServer belong to the same local network (f.e. connected to the same Wi-Fi router). Specify the full URL of the running DemoServer (as **http://<ip address>:8080**) and press **INITIALIZE**:

```
<< /register uid=DB058E75-958D-4707-9E9D-DD86C867B219&provider=APNS&id=c0cbc62f080d528c54cafa7c
>> HTTP/1.1 200 OK
>> Server: UTNotificationsDemoServer
>> Content-Type: text/html
>> Content-Length: 11
>> Connection: close
>>
>> Registered!
```



7. Press **PUSH NOTIFY ALL REGISTERED DEVICES** to send a push notification to all the DemoServer-registered devices. If everything was configured correctly, you should see how

UTNotifications SampleScene handled the push message:



Contacts

If you got any questions please feel free to contact us: universal.tools.contact@gmail.com.

You can post bugs and feature requests at

<https://github.com/universal-tools/UTNotificationsFeedback/issues>.

If you liked using UTNotifications, please [rate it](#), but any criticism is also welcome - please help us make the asset better!

Thank you for using UTNotifications!

Your Universal Tools team.