
Amazon Pinpoint

Developer Guide



Amazon Pinpoint: Developer Guide

Copyright © 2017 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What Is Amazon Pinpoint?	1
Define Audience Segments	1
Engage Your Audience with Messaging Campaigns	1
Analyze User Behavior	1
Integrating Amazon Pinpoint with Mobile Apps	2
Setting Up Push Notifications	3
Setting Up iOS Push Notifications	3
Step 1: Create an App ID	4
Step 2: Create an APNs SSL Certificate	6
Step 3: Register a Test Device	9
Step 4: Create an iOS Distribution Certificate	10
Step 5: Create a Provisioning Profile	11
Setting Up Android Push Notifications	13
Step 1: Create a Firebase Project	13
Step 2: Get Push Messaging Credentials for Android	14
Getting Started with Mobile Apps	16
Getting Started With iOS Apps	16
Adding an iOS App	17
Building the Sample iOS App	17
Getting Started With Android Apps	23
Adding an Android App to Amazon Pinpoint	23
Creating an Android Virtual Device	24
Building the Sample Android App From AWS Mobile Hub	29
Testing the Sample App	30
Integrating with Mobile Apps	32
Integrating Amazon Pinpoint With iOS Apps	32
Setting Up SDK Support	32
Initializing the Amazon Pinpoint Client	34
Registering Endpoint Profiles	37
Reporting Events	37
Setting Up Deep Linking	38
Integrating Amazon Pinpoint with Android Apps	40
Setting up SDK Support	40
Initializing the Amazon Pinpoint Client	41
Managing Sessions	44
Registering Endpoint Profiles	45
Reporting Events	45
Setting up Deep Linking	46
Adding Endpoints	48
Adding Endpoints With the AWS SDK for Java	48
Creating Segments	51
Building Segments	51
Building Segments With the AWS SDK for Java	51
Importing Segments	52
Importing a Segment	53
Creating Campaigns	55
Creating Standard Campaigns	55
Creating Campaigns With the AWS SDK for Java	55
Creating A/B Test Campaigns	56
Creating A/B Test Campaigns With the AWS SDK for Java	56
Streaming Events	58
Setting up Event Streaming	58
AWS CLI	58
AWS SDK for Java	59

Disabling Event Streaming	59
AWS CLI	59
AWS SDK for Java	59
Event Data	60
Mobile push events	60
Email Events	61
SMS Events	62
Event Attributes	63
Permissions	65
IAM Policies for Users	65
Example Policies	66
API Actions for IAM Policies	67
App User Authentication	72
Unauthenticated Role	73
AWS Mobile Hub Service Role	74
IAM Role for Importing Segments	75
Trust Policy	75
Creating the IAM Role (AWS CLI)	76
IAM Role for Streaming Events to Amazon Kinesis	77
Permissions Policies	77
Trust Policy	78
Creating the IAM Role (AWS CLI)	78
IAM Role for Exporting Events to Amazon S3	79
Permissions Policy	79
Trust Policy	80
Creating the IAM Role (AWS CLI)	80
Document History	82

What Is Amazon Pinpoint?

Amazon Pinpoint is an AWS service that you can use to improve user engagement. Use Amazon Pinpoint to create campaigns that reach audience segments with tailored messages.

Amazon Pinpoint supports multiple messaging channels. You can choose to send push notifications, emails, or text messages (SMS) depending on the purpose of your campaign and the type of message.

With Amazon Pinpoint, you can do the following:

Define Audience Segments

To reach the right audience with your messages, [define audience segments \(p. 51\)](#). You can define dynamic segments based on data reported by your application, such as operating system or mobile device.

You can also import segments that you define outside of Amazon Pinpoint.

A segment designates which users receive the messages delivered by a campaign.

Engage Your Audience with Messaging Campaigns

Engage your audience segment by [creating a messaging campaign \(p. 55\)](#). A campaign sends tailored messages according to a schedule that you define. You can create a campaign that sends messages through any channel that is supported by Amazon Pinpoint: mobile push, email, or SMS.

To experiment with alternative campaign strategies, set up your campaign as an A/B test, and analyze the results with Amazon Pinpoint analytics.

Analyze User Behavior

Using the analytics provided by Amazon Pinpoint, you can gain insights about your audience and the effectiveness of your campaigns. You can view trends about your users' level of engagement, purchase activity, and demographics. You can monitor your message traffic with metrics for messages sent and opened. Through the Amazon Pinpoint API, your application can report custom data, which Amazon Pinpoint makes available for analysis.

To analyze or store the analytics data outside of Amazon Pinpoint, you can configure Amazon Pinpoint to [stream the data \(p. 58\)](#) to Amazon Kinesis or Amazon S3.

Integrating Amazon Pinpoint with Mobile Apps

Amazon Pinpoint can capture and present information about user engagement in mobile apps. To do this, you [integrate Amazon Pinpoint into your app code \(p. 16\)](#) so the app can connect to the service and then report events about app usage. Amazon Pinpoint compiles the app usage events into user engagement analytics that you use to identify and engage with specific segments of users.

Setting Up Push Notifications for Amazon Pinpoint

Before using Amazon Pinpoint, you must add your app as a project in AWS Mobile Hub. When you add your project, you provide the credentials that authorize Amazon Pinpoint to send messages to your app through the push notification services for iOS and Android:

- For iOS apps, you provide an SSL certificate, which you obtain from the Apple Developer portal. The certificate authorizes Amazon Pinpoint to send messages to your app through Apple Push Notification service (APNs).
- For Android apps, you provide an API Key and a sender ID, which you obtain from the Firebase console or the Google API console. These credentials authorize Amazon Pinpoint to send messages to your app through Firebase Cloud Messaging or its predecessor, Google Cloud Messaging.

After you have the credentials, you can complete the steps in [Getting Started: Creating an Mobile App With Amazon Pinpoint Support \(p. 16\)](#).

If you already have the credentials, you can skip this section.

Topics

- [Setting Up iOS Push Notifications \(p. 3\)](#)
- [Setting Up Android Push Notifications \(p. 13\)](#)

Setting Up iOS Push Notifications

Push notifications for iOS apps are sent using Apple Push Notification service (APNs). Before you can send push notifications to iOS devices, you must create an app ID on the Apple Developer portal, and you must create the required certificates.

This section describes how to use the Apple Developer portal to obtain iOS and APNs credentials. These credentials enable you to create an iOS project in AWS Mobile Hub and launch a sample app that can receive push notifications.

You do not need an existing iOS app to complete the steps in this section. After you create an iOS project in Mobile Hub, you can download and launch a working sample app. Mobile Hub automatically provisions the AWS resources that your app requires.

After completing the steps in this section, you will have the following items in your Apple Developer account:

- An app ID.
- An SSL certificate, which authorizes you to send push notifications to your app through APNs.
- A registration for your test device, such as an iPhone, with your Apple Developer account.
- An iOS distribution certificate, which enables you to install your app on your test device.
- A provisioning profile, which allows your app to run on your test device.

If you already have these items, you can skip this section and complete the steps in [Getting Started With iOS Apps \(p. 16\)](#).

Before you begin, you must have an account with the Apple Developer Program as an individual or as part of an organization, and you must have agent or admin privileges in that account.

The steps in this tutorial are based on the following versions of Mac OS software:

- OS X El Capitan version 10.11.6
- Xcode version 8.1

Topics

- [Step 1: Create an App ID \(p. 4\)](#)
- [Step 2: Create an APNs SSL Certificate \(p. 6\)](#)
- [Step 3: Register a Test Device \(p. 9\)](#)
- [Step 4: Create an iOS Distribution Certificate \(p. 10\)](#)
- [Step 5: Create a Provisioning Profile \(p. 11\)](#)

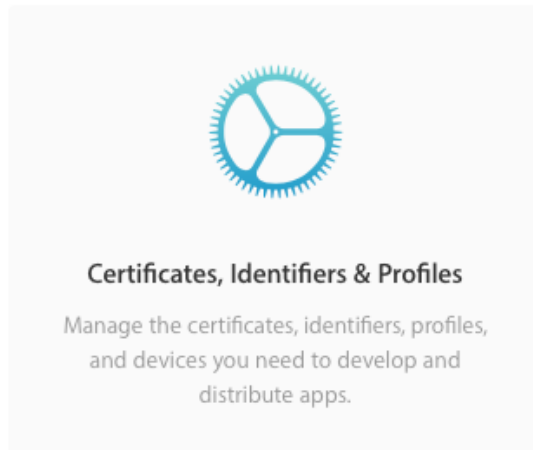
Step 1: Create an App ID

Create an app ID to identify your app in the Apple Developer portal. You need this ID when you create an SSL certificate for sending push notifications, when you create an iOS distribution certificate, and when you create a provisioning profile.

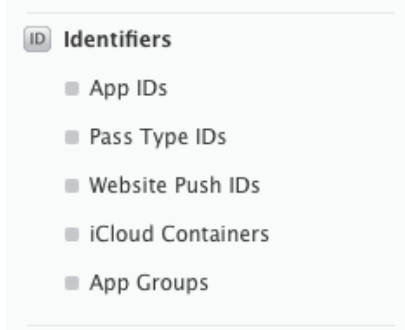
If you already have an ID assigned to your app, you can skip this step. You can use an existing app ID provided it doesn't contain a wildcard character ("*").

To assign an App ID to your app

1. Sign in to your Apple Developer account at <https://developer.apple.com/membercenter/index.action>.
2. Choose **Certificates, Identifiers & Profiles**.



3. In the **Identifiers** section, choose **App IDs**.



4. In the **iOS App IDs** pane, choose the **Add** button (+).



5. In the **Registering an App ID** pane, for **Name**, type a custom name for your app ID that makes it easy to recognize later.

ID Registering an App ID

The App ID string contains two parts separated by a period (.)—an App ID Prefix that is defined as your Team ID by default and an App ID Suffix that is defined as a Bundle ID search string. Each part of an App ID has different and important uses for your app. [Learn More](#)

App ID Description

Name:
You cannot use special characters such as @, &, *, ', "

App ID Prefix

Value: (Team ID)

6. Choose the default selection for an App ID Prefix.
7. For **App ID Suffix**, select **Explicit App ID**, and type a bundle ID for your app. If you already have an app, use the bundle ID assigned to it. You can find this ID in the app project in Xcode on your Mac. Otherwise, take note of the bundle ID because you will assign it to your app in Xcode later.

App ID Suffix

Explicit App ID

If you plan to incorporate app services such as Game Center, In-App Purchase, Data Protection, and iCloud, or want a provisioning profile unique to a single app, you must register an explicit App ID for your app.

To create an explicit App ID, enter a unique string in the Bundle ID field. This string should match the Bundle ID of your app.

Bundle ID:

We recommend using a reverse-domain name style string (i.e., com.domainname.appname). It cannot contain an asterisk (*).

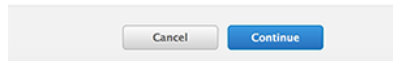
Wildcard App ID

This allows you to use a single App ID to match multiple apps. To create a wildcard App

8. Under **App Services** select **Push Notifications**.

Include CloudKit support
(requires Xcode 6)

- In-App Purchase
- Inter-App Audio
- Wallet
- Push Notifications
- VPN Configuration & Control



9. Choose **Continue**. In the **Confirm your App ID** pane, check that all values were entered correctly. The identifier should match your app ID and bundle ID.

10. Choose **Register** to register the new app ID.

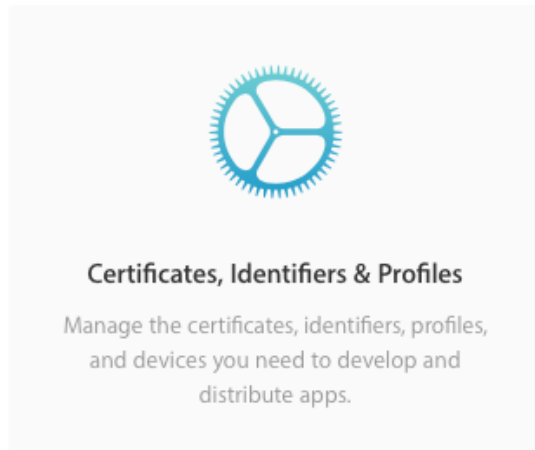
Step 2: Create an APNs SSL Certificate

Create an APNs SSL certificate so that you can deliver push notifications to your app through APNs. You will create and download the certificate from your Apple Developer account. Then, you will install the certificate in Keychain Access and export it as a .p12 file.

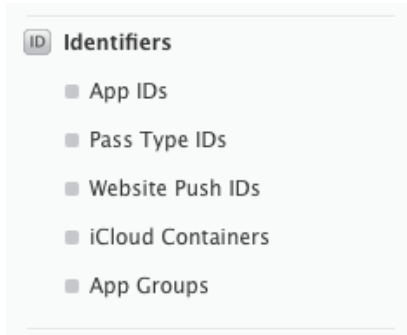
If you already have an SSL certificate for your app, you can skip this step.

To create an SSL certificate for push notifications

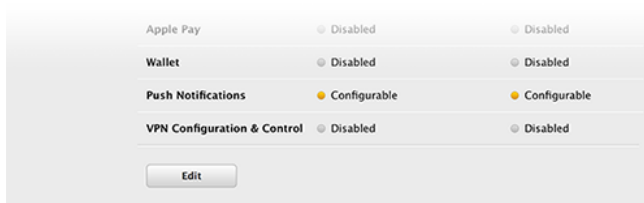
1. Sign in to your Apple Developer account at <https://developer.apple.com/membercenter/index.action>.
2. Choose **Certificates, Identifiers & Profiles**.



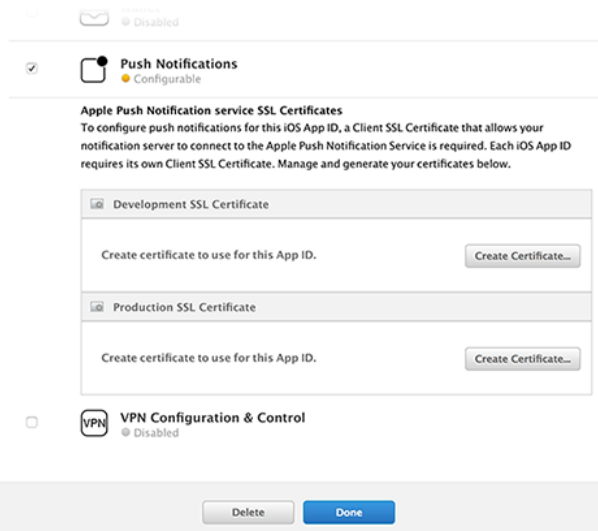
3. From the **Identifiers** section, choose **App IDs**.



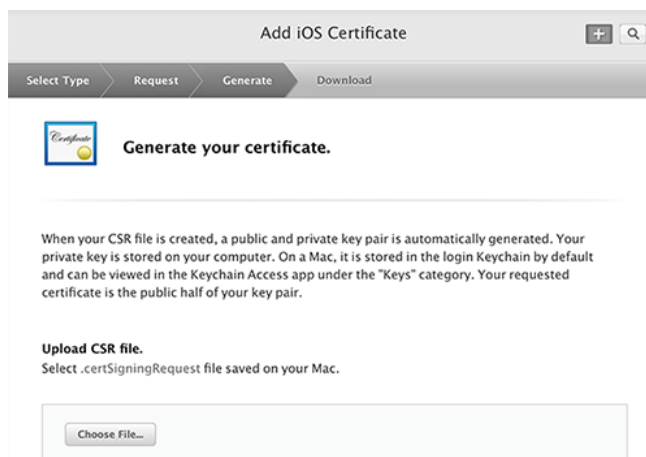
4. From the list of iOS app IDs, select the app ID that you created in [Step 1: Create an App ID \(p. 4\)](#).
5. Choose **Edit**.



6. Under **Push Notifications**, in the **Production SSL Certificate** section, choose **Create Certificate....**



7. In the **About Creating a Certificate Signing Request (CSR)** pane, follow the instructions for creating a Certificate Signing Request (CSR) file. You use the Keychain Access application on your Mac to create the request and save it on your local disk. When you are done, choose **Continue**.
8. In the **Generate your certificate** pane, choose **Choose File...**, and then select the CSR file you created.



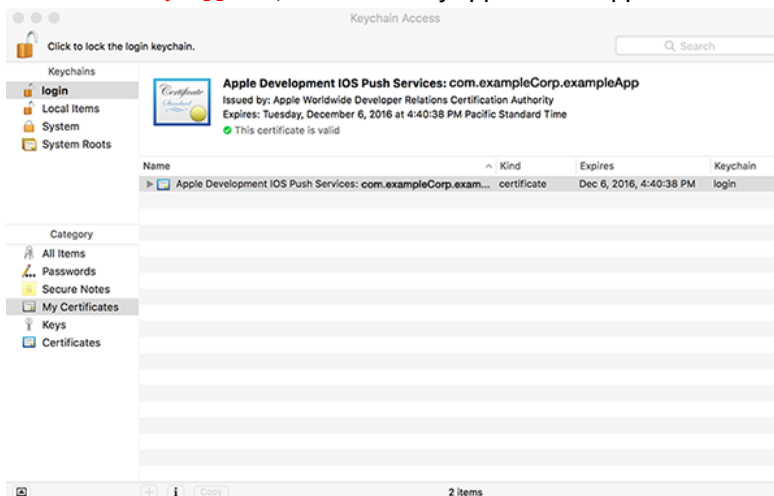
9. Choose **Continue**.
10. When the certificate is ready, choose **Download** to save the certificate to your computer.



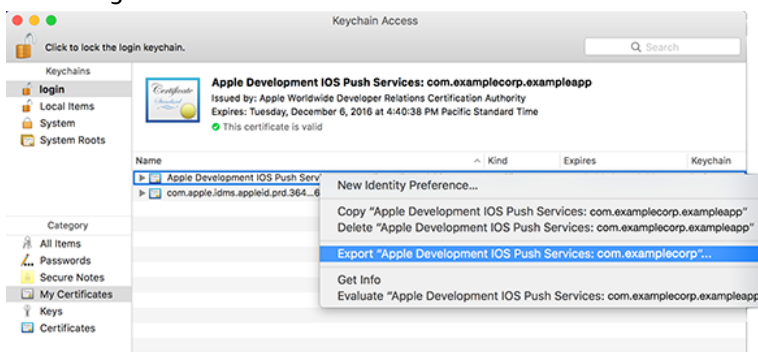
Documentation
For more information on using and managing your certificates read:

11. Double-click the downloaded certificate to install it to the Keychain on your Mac.
12. On your Mac, start the Keychain Access application.

13. In **My Certificates**, find the certificate you just added. The certificate is named "Apple Push Services:*com.my.app.id*", where *com.my.app.id* is the app ID for which the certificate was created.



14. Context-select the push certificate and then select **Export...** from the context menu to export a file containing the certificate.



15. Type a name for the certificate that is easy to recognize and save it to your computer. Do not provide an export password when prompted. You need to upload this certificate when creating your app in AWS Mobile Hub.

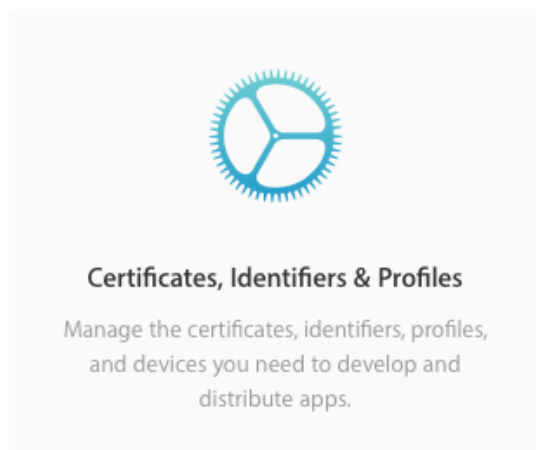
Step 3: Register a Test Device

Register a test device with your Apple Developer account so that you can test your app on that device. Later, you associate this test device with your provisioning profile, which allows your app to launch on your device.

If you already have a registered device, you can skip this step.

To add a device

1. Sign in to your Apple Developer account at <https://developer.apple.com/membercenter/index.action>.
2. Choose **Certificates, Identifiers & Profiles**.



3. In the **Devices** section, choose the type of device that you want to add, such **iPhone**.
4. Choose the **Add** button (+).
5. In the **Register Device** section, for **Name**, type a name that is easy to recognize later.
6. For **UDID**, type the unique device ID. For an iPhone, you can find the UDID by completing the following steps:
 - a. Connect your iPhone to your Mac with a USB cable.
 - b. Open the iTunes app.
 - c. In the top left corner of the iTunes window, a button with an iPhone icon is shown. Choose this button. iTunes displays the summary page for your iPhone.
 - d. In the top box, the summary page provides your iPhone's **Capacity**, **Phone Number**, and **Serial Number**. Click **Serial Number**, and the value changes to **UDID**.
 - e. Context-select your UDID, and choose **Copy**.
 - f. Paste your UDID into the **UDID** field in the Apple Developer website.
7. Choose **Continue**.
8. On the **Review and register** pane, verify the details for your device, and choose **Register**. Your device name and identifier are added to the list of devices.

Step 4: Create an iOS Distribution Certificate

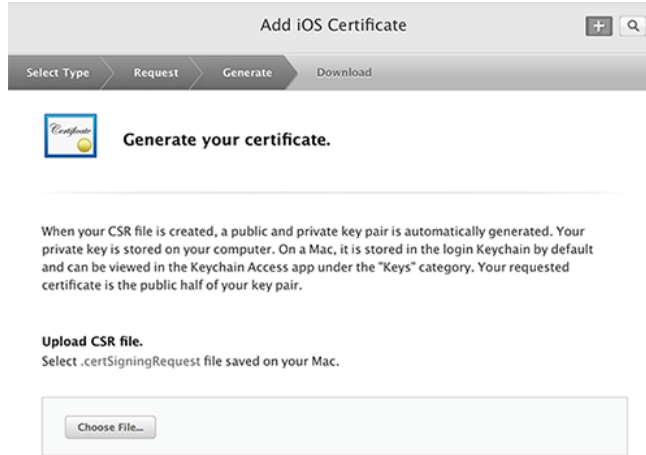
An iOS distribution certificate enables you to install your app on a test device and deliver push notifications to that device. You specify your iOS distribution certificate later when you create a provisioning profile for your app.

If you already have an iOS distribution certificate, you can skip this step.

To create an iOS distribution certificate

1. On the **Certificates, Identifiers & Profiles** page of your Apple Developer account, in the **Certificates** section, choose **Production**.
2. In the **iOS Certificates (Production)** pane, choose the Add button (+). The **Add iOS Certificate** pane opens.
3. In the **Production** section, select **App Store and Ad Hoc**, and then choose **Continue**.
4. On the **About Creating a Certificate Signing Request (CSR)** page, choose **Continue**.

5. In the **About Creating a Certificate Signing Request (CSR)** pane, follow the instructions for creating a Certificate Signing Request (CSR) file. You use the Keychain Access application on your Mac to create the request and save it on your local disk. When you are done, choose **Continue**.
6. In the **Generate your certificate** pane, choose **Choose File...**, and then select the CSR file you created.



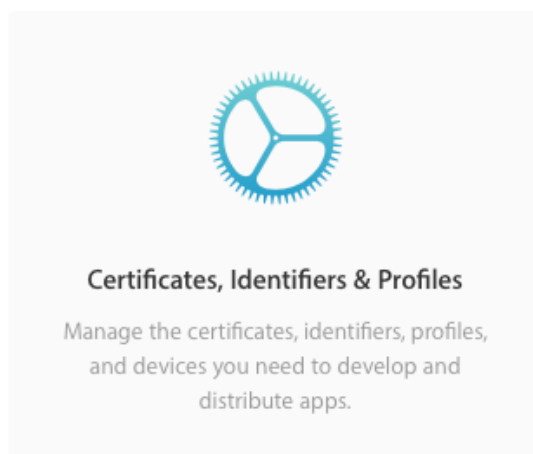
7. Choose **Continue**.
8. When the certificate is ready, choose **Download** to save the certificate to your computer.
9. Double-click the downloaded certificate to install it in Keychain on your Mac.

Step 5: Create a Provisioning Profile

A provisioning profile allows your app to run on your test device. You create and download a provisioning profile from your Apple Developer account and then install the provisioning profile in Xcode.

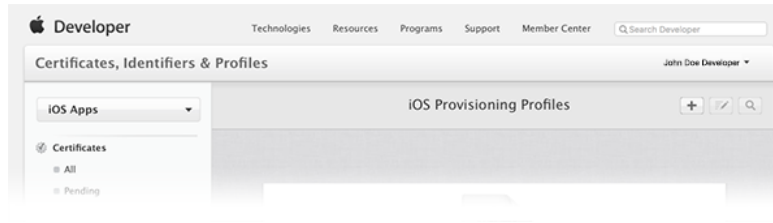
To create a provisioning profile

1. Sign in to your Apple Developer account at <https://developer.apple.com/membercenter/index.action>.
2. Select **Certificates, Identifiers & Profiles**.

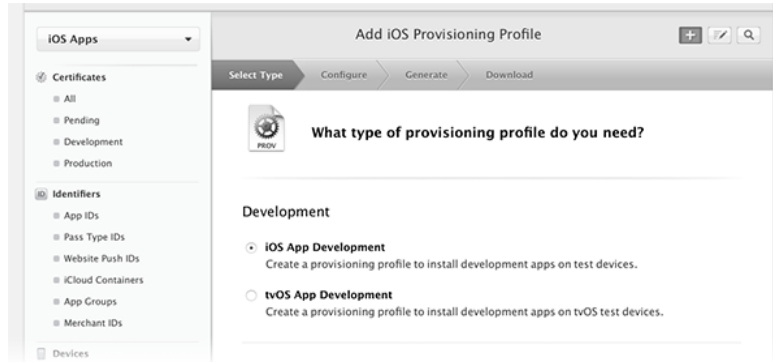


3. In the **Provisioning Profiles** section, choose **Distribution**.

- In the **iOS Provisioning Profiles (Distribution)** pane, choose the add button (+). The **Add iOS Provisioning Profiles (Distribution)** pane is shown.



- In the **Distribution** section, select **Ad Hoc**, and then choose **Continue**.



- For **App ID**, select the app ID you created for your app, and then choose **Continue**.



- Select your iOS Development certificate and then choose **Continue**.
- For **Select devices**, select the device that you registered for testing, and choose **Continue**.
- Type a name for this provisioning profile, such as **AppnsDistributionProfile**, and choose **Continue**.
- Select **Download** to download the generated provisioning profile.
- Install the provisioning profile by double-clicking the downloaded file. The Xcode app opens in response.
- To verify that the provisioning profile is installed, check the list of installed provisioning profiles in Xcode by doing the following:
 - In the Xcode menu bar, choose **Xcode** and then choose **Preferences**.
 - In the preferences window, choose **Accounts**.
 - In the **Accounts** tab, select your Apple ID, and then choose **View Details**.
 - Check that your provisioning profile is listed in the **Provisioning Profiles** section.

Setting Up Android Push Notifications

This section describes how to obtain the credentials required to send push notifications to Android apps. The platform notification services you can use for push notifications on Android are Firebase Cloud Messaging (FCM) and its predecessor, Google Cloud Messaging (GCM). Your FCM or GCM credentials enable you to create an Android project in AWS Mobile Hub and launch a sample app that can receive push notifications.

You do not need an existing Android app to complete the steps in this section. After you create an Android project in Mobile Hub, you can download and launch a working sample app. Mobile Hub automatically provisions the AWS resources that your app requires.

After completing the steps in this section, you will have an API key and sender ID in the Firebase console or the Google Cloud Platform console. If you already have these credentials, you can skip this section and complete the steps in [Getting Started With Android Apps \(p. 23\)](#).

Topics

- [Step 1: Create a Firebase Project \(p. 13\)](#)
- [Step 2: Get Push Messaging Credentials for Android \(p. 14\)](#)

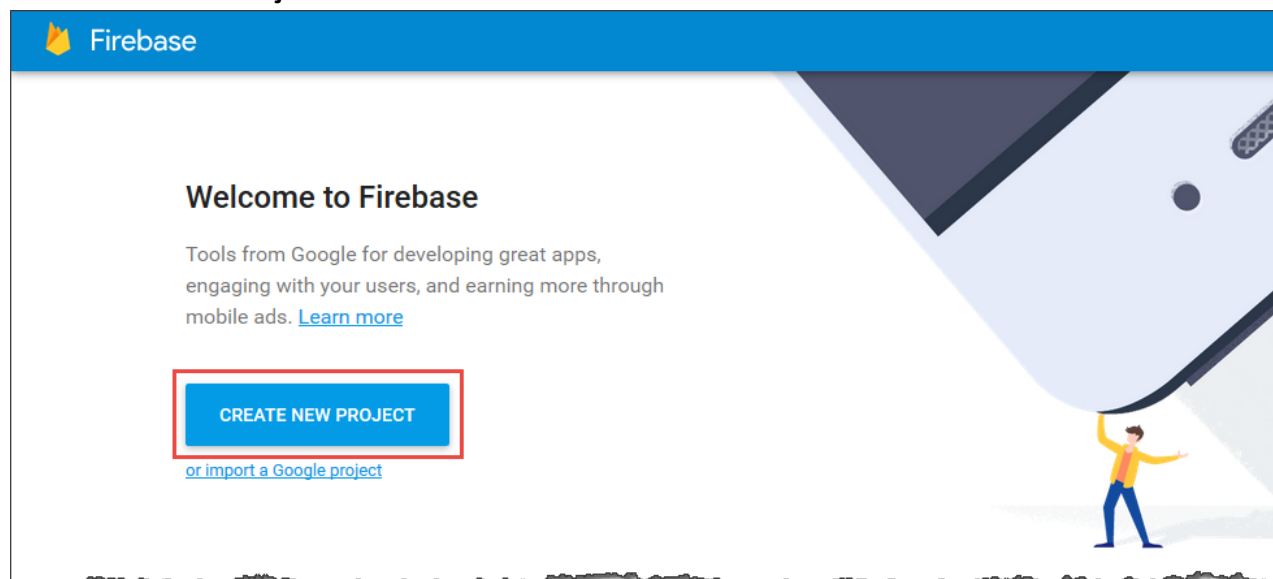
Step 1: Create a Firebase Project

To send push notifications to Android apps, you must have a project that is enabled with an Android push notification service. The push notification services for Android are Firebase Cloud Messaging (FCM) and its predecessor, Google Cloud Messaging (GCM).

If you are new to push messaging on Android, you must create a Firebase project, as this topic describes. However, if you have an existing Google Cloud Messaging project that has push messaging enabled, you can skip this step and use that project instead.

To create a Firebase project

1. Go to the Firebase console at <https://console.firebase.google.com/>. If you are not signed in to Google, the link takes you to a sign-in page. After you sign in, you see the Firebase console.
2. Choose **Create New Project**.



3. Type a project name, and then choose **Create Project**.

Firebase projects support push messaging by default.

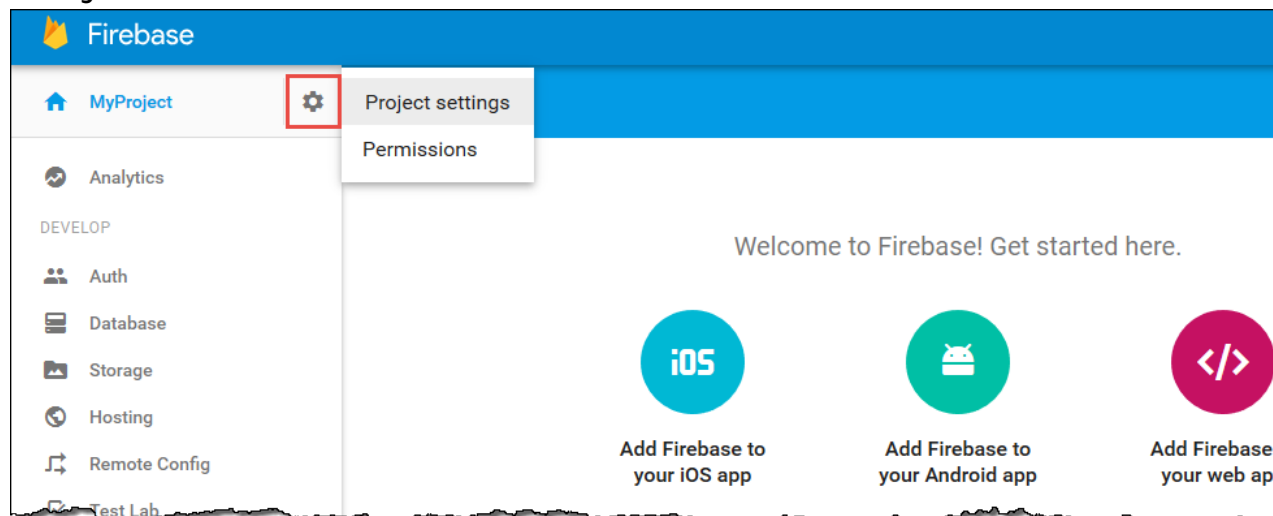
Step 2: Get Push Messaging Credentials for Android

To send push notifications to Android apps, you must have credentials from either Firebase Cloud Messaging (FCM) or its predecessor, Google Cloud Messaging (GCM). The credentials are an API key and a sender ID (also called project number). You get these credentials from a project that has push messaging enabled. This project could either be in the Firebase console or the Google Cloud Platform console, depending on where you created it.

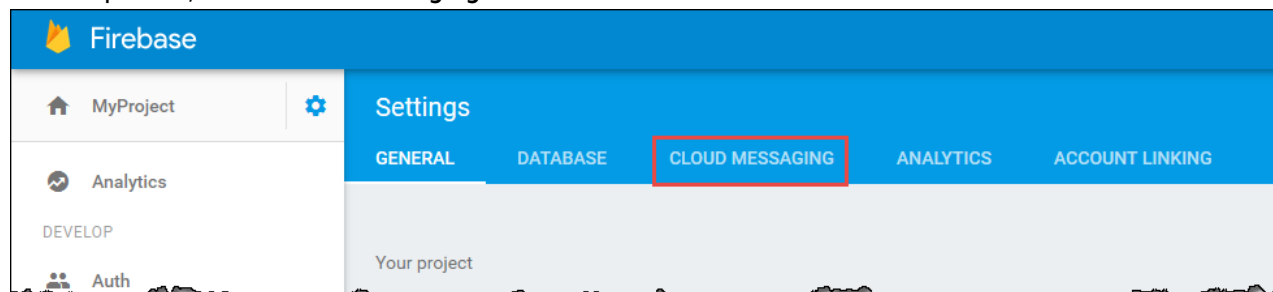
This topic describes how to retrieve your credentials from FCM or GCM. Use FCM for new Android apps. Use GCM only if you have a preexisting GCM project that you have not yet updated for FCM support.

To obtain your credentials from FCM

1. Go to the Firebase console at <https://console.firebase.google.com/> and open your project.
2. In the left pane, to the right of your project name, choose the gear icon, and then choose **Project Settings**.



3. In the top menu, choose **Cloud Messaging**.



4. Under **Project credentials**, you find the API key and sender ID. Save these values somewhere you can access later.

To obtain your credentials from GCM

1. Go to the Google API Console at <https://console.developers.google.com>.
2. In the left pane, choose **Credentials**.
3. If you already have credentials for your app, your server key is shown in the **API keys** section. Save this key somewhere you can access later.
4. If you don't have credentials for your app, the console displays the **Credentials** dialog box. Create a server key by completing the following steps:
 - a. Choose **Create credentials**.
 - b. Save the API key somewhere you can access later.
5. To retrieve your sender ID (also called project number), go to the Google Cloud Platform console at <https://console.cloud.google.com/>. Select your project from the **Project** menu. Then, choose the arrow next to the project name.
6. Save the displayed project number somewhere you can access later.

Note

Project number is another name for sender ID.

Getting Started: Creating a Mobile App With Amazon Pinpoint Support

Before you can use Amazon Pinpoint, you must add your app as a project in AWS Mobile Hub and integrate your app with Amazon Pinpoint.

Mobile Hub is an AWS service that helps you create and configure mobile app backend features and integrate them into your app. To add a project and enable Amazon Pinpoint support, you require credentials that authorize Amazon Pinpoint to send push notifications to your app through the push notifications services for Android or iOS. If you need to obtain these credentials, see [Setting Up Push Notifications for Amazon Pinpoint \(p. 3\)](#).

To integrate your app with Amazon Pinpoint, you download the necessary SDKs from Mobile Hub, and you add these SDKs to your app code.

To test the Amazon Pinpoint features that you enable in Mobile Hub, you download and launch a working sample app. Then, you can use Amazon Pinpoint to create a simple push notification campaign and send a message to the sample app.

If you want to use your own app instead of the Mobile Hub sample app, you can examine the sample app code for guidance on how to include support for push notifications in your own app. The Mobile Hub console also provides steps for integrating the mobile SDKs in your app and enabling Amazon Pinpoint support.

Topics

- [Getting Started With iOS Apps \(p. 16\)](#)
- [Getting Started With Android Apps \(p. 23\)](#)
- [Testing the Sample App With Amazon Pinpoint \(p. 30\)](#)

Getting Started With iOS Apps

To add an iOS app to Amazon Pinpoint, create a project in AWS Mobile Hub that has Amazon Pinpoint support.

After you create the project, you can download a functional sample app from Mobile Hub, and run the sample app on a test device that is registered with your Apple Developer account. Then, you can send a push notification to the sample app using Amazon Pinpoint.

Topics

- [Adding an iOS App to Amazon Pinpoint \(p. 17\)](#)
- [Building the Sample iOS App From AWS Mobile Hub \(p. 17\)](#)

Adding an iOS App to Amazon Pinpoint

Add an iOS app to Amazon Pinpoint by creating a project in AWS Mobile Hub.

Prerequisite

To complete this task, you need an SSL certificate as a `.p12` file that authorizes Amazon Pinpoint to send push notifications to your app through Apple Push Notification service (APNs). For more information, see [Setting Up iOS Push Notifications \(p. 3\)](#).

To add an iOS app

1. Sign in to the AWS Management Console and open the Mobile Hub console at <https://console.aws.amazon.com/mobilehub>.
2. If you have other Mobile Hub projects, choose **Create new mobile project**. If this is your first project, skip this step because you are taken directly to the page for creating a new project.
3. Enter a project name. The name you enter will be the name of your project in the Amazon Pinpoint console.
4. For the region, keep **US East (Virginia)**.
5. Choose **Create project**. Mobile Hub creates the project and shows the **Pick and configure features for your project** page.
6. Choose **Messaging & Analytics**.
7. On the **Messaging & Analytics** page, for **What engagement features do you want to enable?**, choose **Messaging**.
8. For **What Messaging Channels do you want to enable?**, choose **Mobile push**.
9. For **Choose the platforms for which you want to enable push notifications**, choose **iOS**.
10. For **P12 Certificate**, provide the P12 certificate that authorizes Amazon Pinpoint to send push notifications to your app through Apple Push Notification service (APNs). Then, choose **Upload**.
11. Choose **Enable**.

Mobile Hub helps you integrate Amazon Pinpoint with your app in two ways:

1. You can download a working sample app that demonstrates Amazon Pinpoint features. Then, you can [build the sample app \(p. 17\)](#) and send it a push notification with Amazon Pinpoint.
2. You can download a package that includes the required SDKs for your project. Then, you can follow the instructions in the AWS Mobile Hub console to integrate Amazon Pinpoint with your app.

Now that you have created a Mobile Hub project for your app, you can see your app in the Amazon Pinpoint console at <https://console.aws.amazon.com/pinpoint/>.

Building the Sample iOS App From AWS Mobile Hub

To test the Amazon Pinpoint features that you enabled in AWS Mobile Hub, build the sample iOS app and launch it on a test device. Then, create a simple push notification campaign in Amazon Pinpoint and send a message to the sample app.

Prerequisites

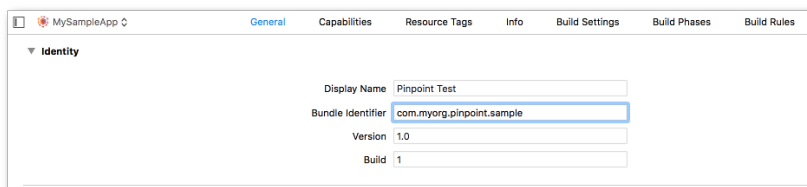
To complete this task, you need:

- A Mobile Hub sample iOS app with Amazon Pinpoint support. If you need to create the sample app, see [Adding an iOS App to Amazon Pinpoint \(p. 17\)](#).
- A Mac with Xcode installed. The following procedure is based on OS X El Capitan version 10.11.6 and Xcode version 8.1.
- The following items from the Apple Developer portal:
 - An app ID.
 - A registration for a test device (such as an iPhone).
 - An iOS distribution certificate installed in Keychain on your Mac.
 - An Ad Hoc distribution provisioning profile installed in Xcode on your Mac.

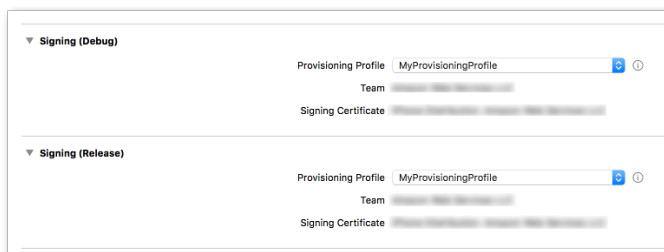
If you need to obtain these items, see [Setting Up iOS Push Notifications \(p. 3\)](#).

To build the sample app

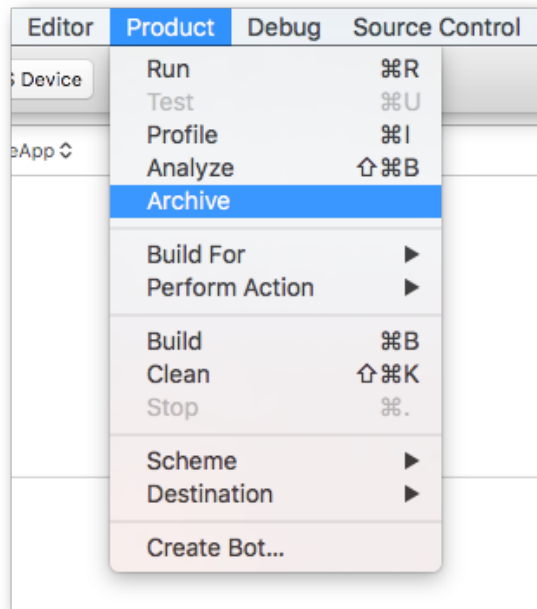
1. In the AWS Mobile Hub console, on the **Integrate** page for your app, choose **Download a sample app**, and save the sample app to your local drive.
2. Browse to the package you downloaded and open the `MySampleApp.xcodeproj` file to open the sample app in Xcode.
3. Open the target for the sample app project, and choose **General**.
4. In the **Identity** section, for **Display Name**, type a custom display name to make the sample app easy to recognize after you install it on your device, such as **Pinpoint Sample**.
5. For **Bundle Identifier**, type the bundle ID associated with the provisioning profile for your app.



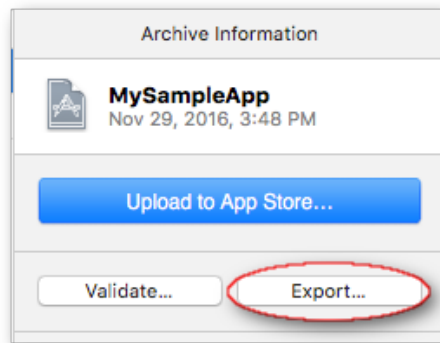
6. In the **Signing (Debug)** and **Signing (Release)** sections, for **Provisioning Profile**, select your provisioning profile. Xcode shows only those provisioning profiles that are associated with the bundle ID that you provided.



7. In the **Product** menu, choose **Archive**.



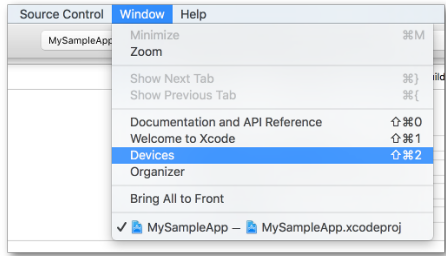
8. In the **Archives** window, choose **Export**.



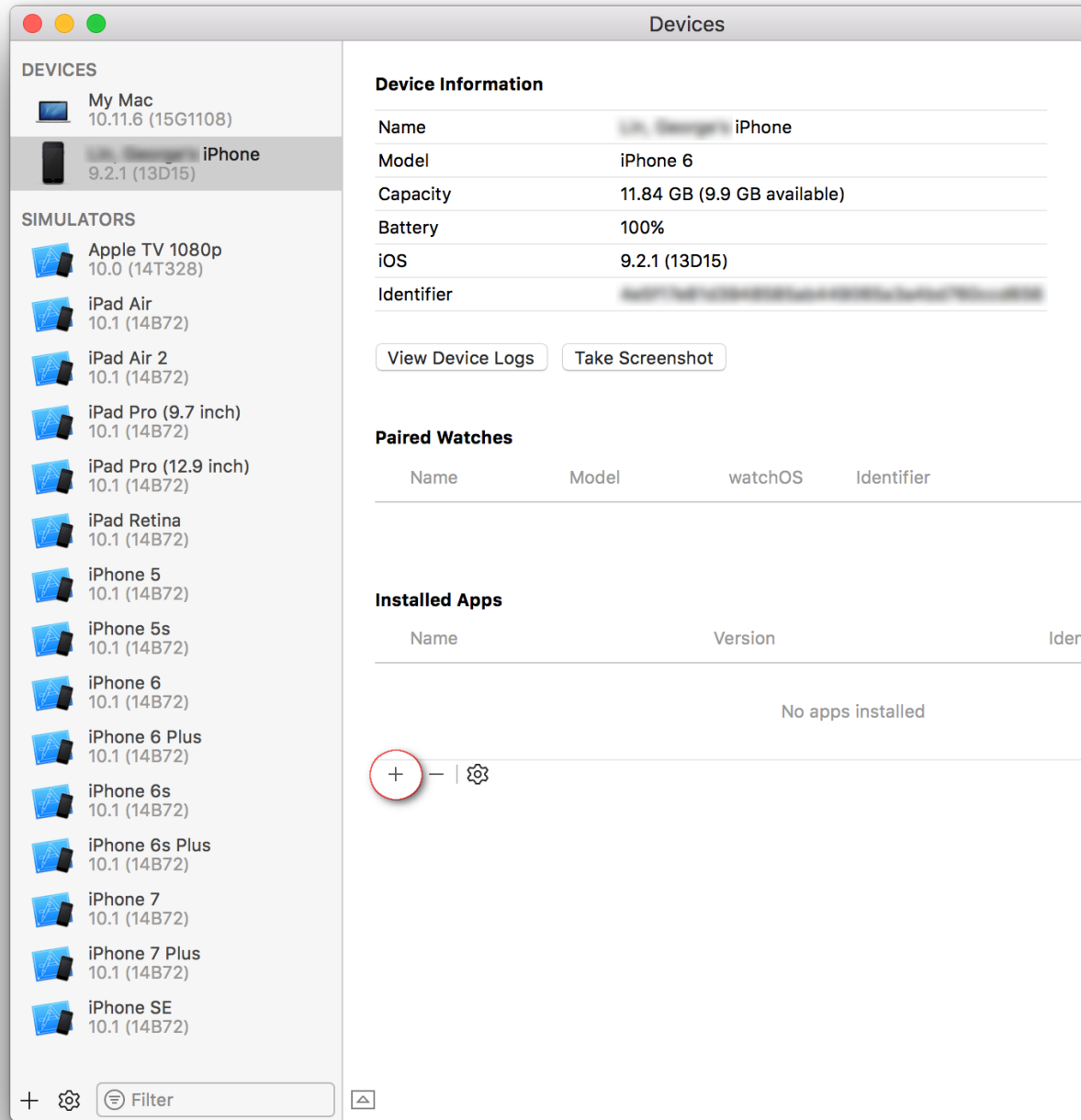
9. In the **Select a method for export** window, select **Save for Ad Hoc Deployment**, and choose **Next**.
10. When prompted, choose your development team.
11. In the **Device Support** window, select **Export one app for all compatible devices**, and choose **Next**.
12. In the **Summary** window, choose **Next**.
13. Browse to the location on your local drive where you want to save the exported `.ipa` file, and choose **Export**.

To launch the sample app on your test device

1. Connect your device to your Mac with a USB cable.
2. In the Xcode menu bar, choose **Window**, and then choose **Devices**.



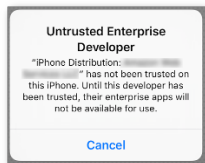
3. In the **Devices** section, select your device.
4. In the **Installed Apps** section, choose the plus icon.



5. Browse to the location of your exported .ipa file, select it, and choose **Open**. The sample app is installed, and you can see the app icon on your device.



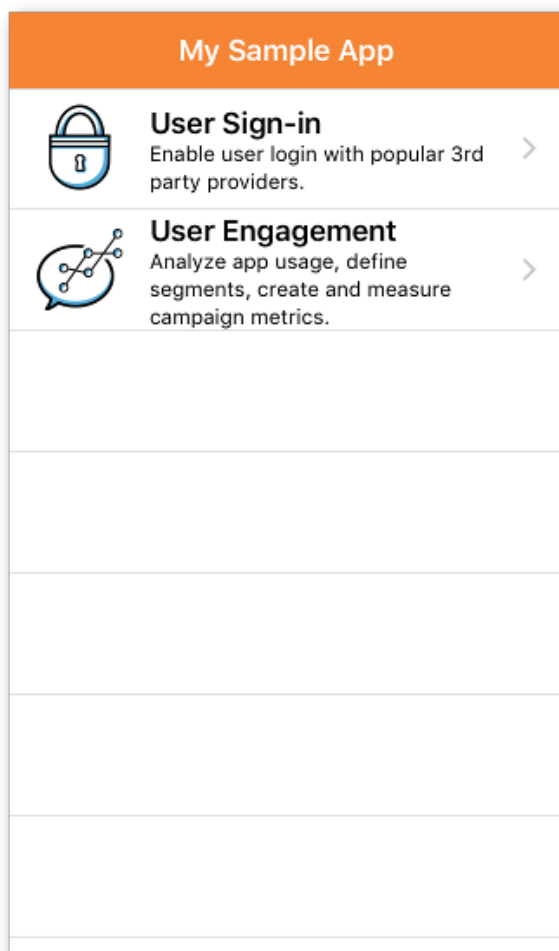
6. Launch the app on your device. If this your first time launching an app signed with your distribution certificate, you receive an **Untrusted Developer** message, and the app won't launch.



Before you can open your app, you must trust your distribution certificate on your device. Tap **Settings**, **General**, and finally **Device Management**. Then, tap the distribution certificate, tap **Trust**, and when prompted, tap **Trust**.

Launch the app again.

7. When the app requests permission to send you notifications, tap **Allow**.
8. In the sample app, you can tap **User Sign-in** or **User Engagement** to learn more about these features.



Now that you have built the sample iOS app, you can [test the app \(p. 30\)](#) by sending it a push notification from Amazon Pinpoint.

Getting Started With Android Apps

To add an Android app to Amazon Pinpoint, create a project in AWS Mobile Hub that has Amazon Pinpoint support.

After you create the project, you can download a functional sample app from Mobile Hub, and run the sample app in an Android emulator using Android Studio. Then, you can send a push notification to the sample app using Amazon Pinpoint.

Topics

- [Adding an Android App to Amazon Pinpoint \(p. 23\)](#)
- [Creating an Android Virtual Device \(p. 24\)](#)
- [Building the Sample Android App From AWS Mobile Hub \(p. 29\)](#)

Adding an Android App to Amazon Pinpoint

Add an Android app to Amazon Pinpoint by creating a project in AWS Mobile Hub.

Prerequisite

To complete this task, you need:

- A project that has push messaging enabled with Firebase or Google Cloud Platform. For more information, see [Step 1: Create a Firebase Project \(p. 13\)](#).
- A Firebase Cloud Messaging (FCM) or Google Cloud Messaging (GCM) API key and a sender ID (also called project number). For more information, see [Step 2: Get Push Messaging Credentials for Android \(p. 14\)](#).

To add an Android app

1. Sign in to the AWS Management Console and open the Mobile Hub console at <https://console.aws.amazon.com/mobilehub>.
2. If you have other Mobile Hub projects, choose **Create new mobile project**. If this is your first project, skip this step because you are taken directly to the page for creating a new project.
3. Enter a project name. The name you enter will be the name of your project in the Amazon Pinpoint console.
4. For the region, keep **US East (Virginia)**.
5. Choose **Create project**. Mobile Hub creates the project and shows the **Pick and configure features for your project** page.
6. Choose **Messaging & Analytics**.
7. On the **Messaging & Analytics** page, for **What engagement features do you want to enable?**, choose **Messaging**.
8. For **What Messaging Channels do you want to enable?**, choose **Mobile push**.
9. For **Choose the platforms for which you want to enable push notifications**, choose **Android**.
10. Enter your **API key** and **Sender ID**, which are available from your project that has push messaging enabled in Firebase or Google Cloud Platform. In Google Cloud Platform, the sender ID is called project number.
11. Choose **Enable**.

Mobile Hub helps you integrate Amazon Pinpoint with your app in two ways:

1. You can download a working sample app that demonstrates Amazon Pinpoint features. Then, you can [create a virtual device \(p. 24\)](#) and [build the sample app \(p. 29\)](#) so that you can send it a push notification with Amazon Pinpoint.
2. You can download a package that includes the required SDKs for your project. Then, you can follow the instructions in the AWS Mobile Hub console to integrate Amazon Pinpoint with your app.

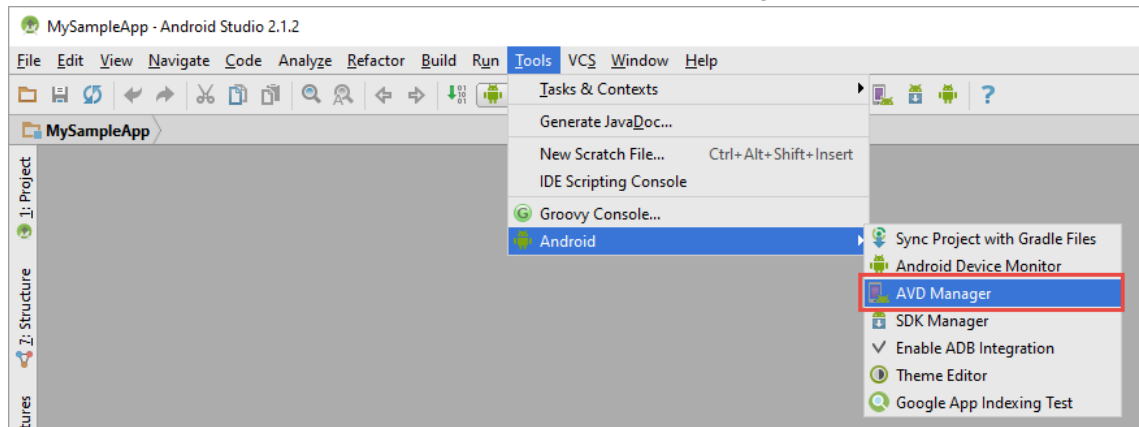
Creating an Android Virtual Device

To send push notifications from Amazon Pinpoint to an Android app, you must install the app on an Android device. For testing purposes, you can use a virtual device. This topic describes how to create a virtual device in Android Studio. If you have a physical Android device and know how to install apps onto it using Android Studio, you can skip this step.

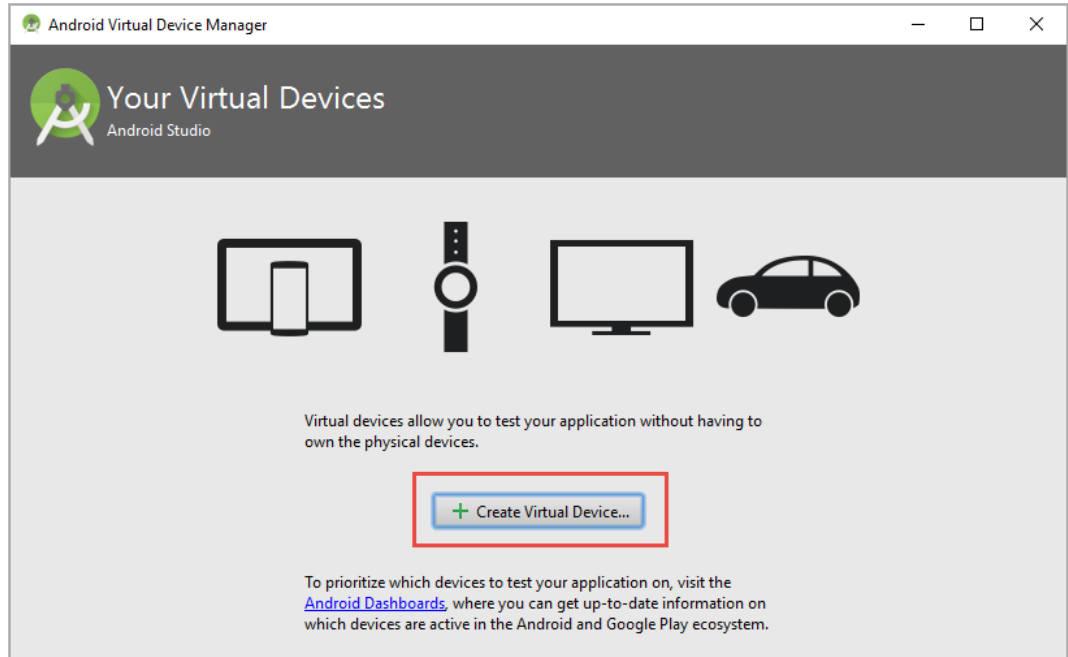
This topic helps you create a virtual device of type Nexus 6 phone with Lollipop API Level 22 x86 Android 5.1, but the AWS Mobile Hub quickstart app is not limited to this device. You can use any device that uses Android OS 4.0.3 (IceCreamSandwich) API Level 15 or higher.

To create a virtual device with Android Studio

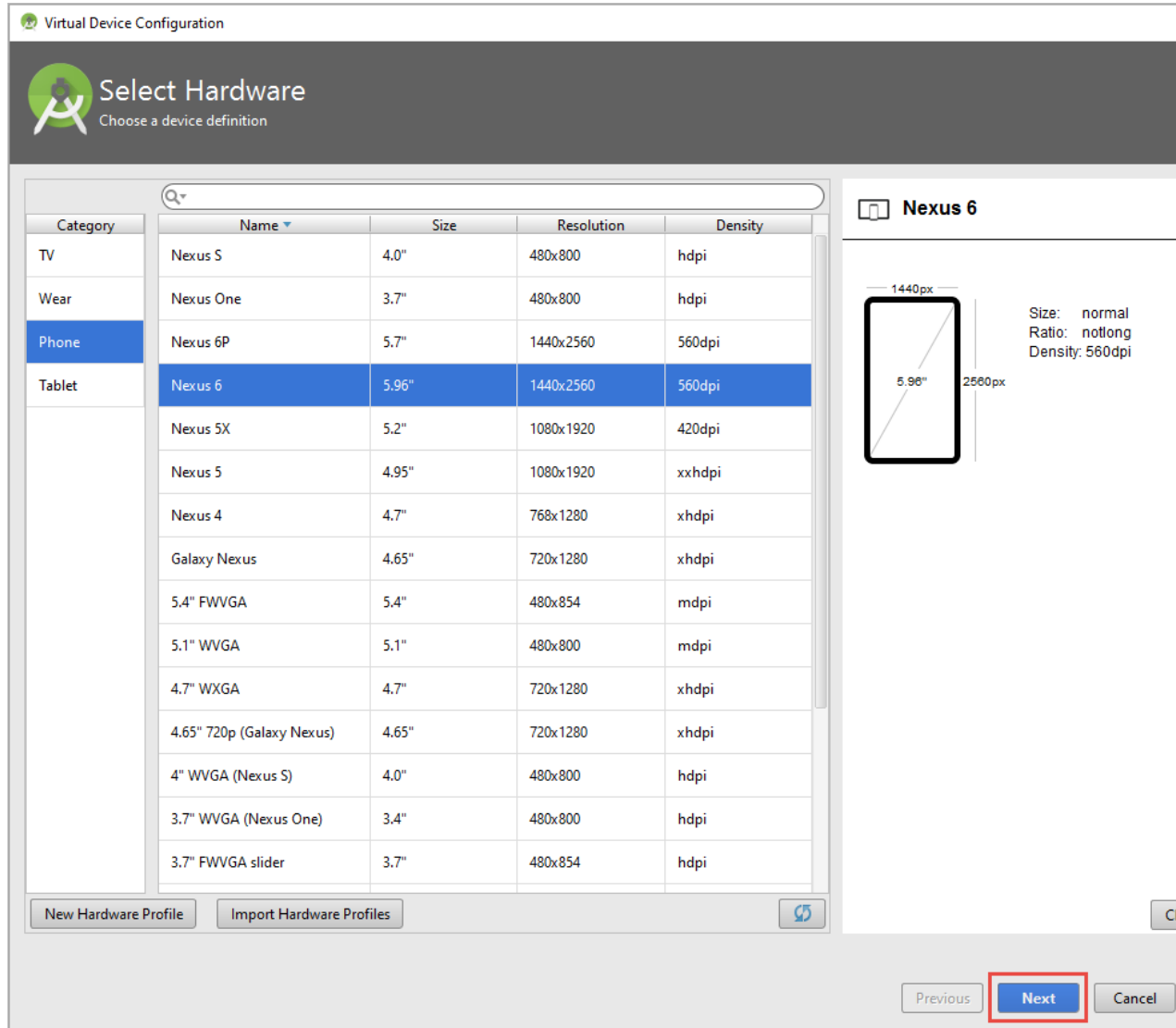
1. Open Android Studio.
2. From the **Tools** menu, choose **Android**, and then choose **AVD Manager**.



3. On the **Your Virtual Devices** page, choose **Create Virtual Device**.



4. On the **Select Hardware** page, choose **Nexus 6**, and then choose **Next**.



5. On the **System Image** page, choose the system image with **Release name** Lollipop, **API Level** 22, **ABI** x86, and **Target** Android 5.1 (with Google APIs), and then choose **Next**.

Virtual Device Configuration


System Image

Select a system image

Recommended x86 Images Other Images

Release Name	API Level	ABI	Target
Marshmallow	23	x86	Android 6.0 (with Google APIs)
Marshmallow	23	x86_64	Android 6.0 (with Google APIs)
Lollipop	22	x86	Android 5.1 (with Google APIs)
Lollipop	22	x86_64	Android 5.1 (with Google APIs)
KitKat Download	19	x86	Android 4.4 (with Google APIs)
Jelly Bean Download	18	x86	Android 4.3 (with Google APIs)
Jelly Bean Download	17	x86	Android 4.2 (with Google APIs)
Jelly Bean Download	16	x86	Android 4.1 (with Google APIs)
Gingerbread Download	10	x86	Android 2.3.3 (with Google APIs)

Lollipop



API Level
22

Android
5.1

Google Inc.

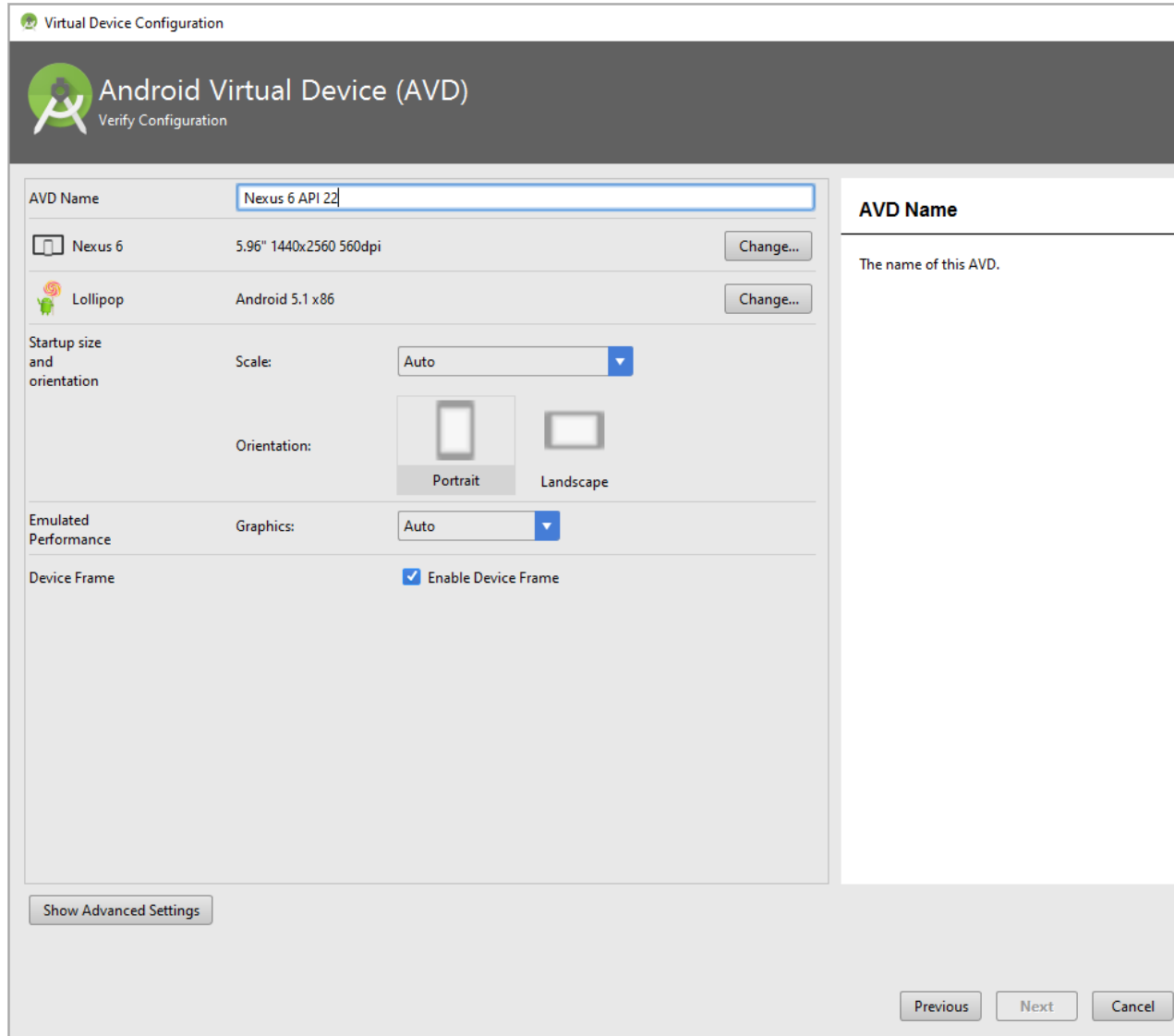
System Image
x86

These images are recommended because they run the fastest and include support for Google APIs

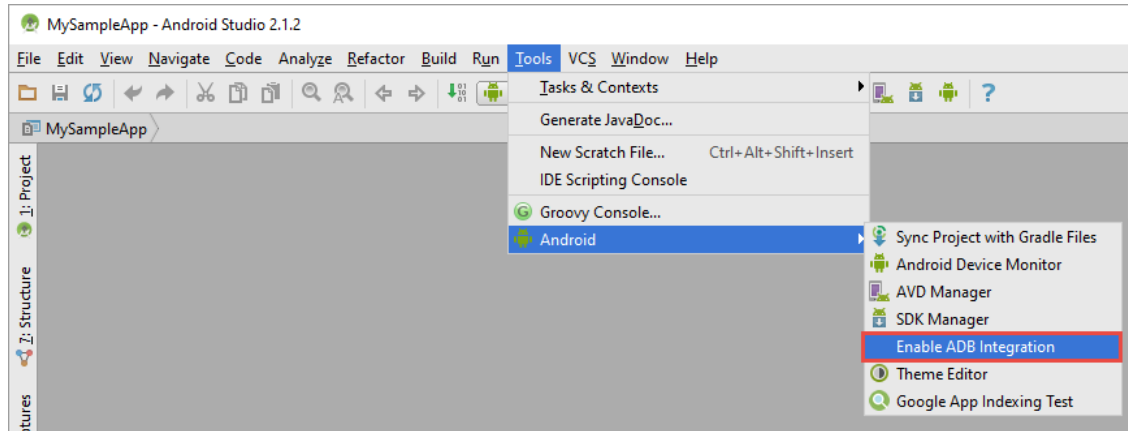
Questions on API level?
See the [API level distribution chart](#)

Previous **Next** Cancel

- On the **Android Virtual Device** page, choose **Finish**.



7. To enable Android Studio to install an app on the virtual device, go to the **Tools** menu, choose **Android**, and then choose **Enable ADB Integration**.



Building the Sample Android App From AWS Mobile Hub

This topic describes how to import and build the AWS Mobile Hub sample app code in Android Studio. We assume that you have generated the quickstart app code using the AWS Mobile Hub console, and downloaded the code in the .zip file to your computer.

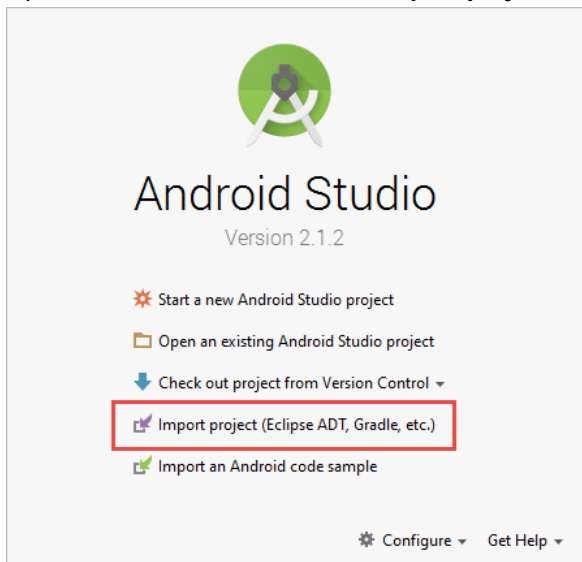
Prerequisites

To complete this task, you need:

- A Mobile Hub sample Android app with Amazon Pinpoint support. If you need to create the sample app, see [Adding an Android App to Amazon Pinpoint \(p. 23\)](#).
- Android Studio with the following SDK support, which you can download from <https://developer.android.com/studio/index.html>:
 - Android Studio 1.4 or newer
 - Android SDK 4.4 (KitKat) API Level 19 or newer
 - Android SDK Build-tools 23.0.1
- A physical or virtual Android device with Android OS 4.0.3 (IceCreamSandwich) API Level 15 or newer and Google Play Services.

To build the AWS Mobile Hub sample app

1. Unzip the .zip file that you downloaded from AWS Mobile Hub.
2. Open Android Studio and choose **Import project (Eclipse ADT, Gradle, etc.)**.

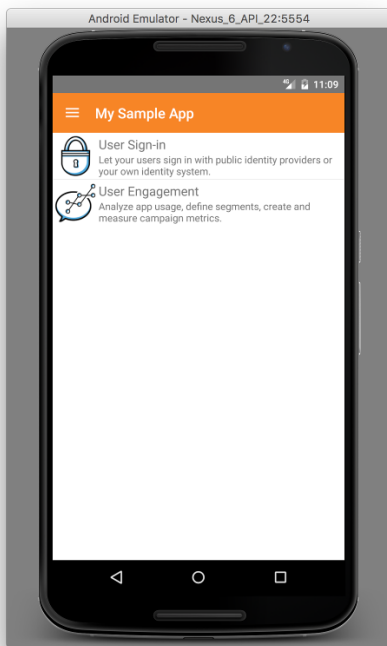


3. Open the folder where you extracted the .zip file contents and choose the **MySampleApp** folder.
Android Studio imports the project and builds it using Gradle.
4. If a dialog box appears recommending that you update your Gradle Plug-in, choose **Don't remind me again for this project**.

After the build completes, you can install and run the app on a physical or virtual Android device.

To run the app on a virtual Android device

1. In Android Studio, choose the run icon in the toolbar.
2. In the **Select a Deployment Target** window, select your virtual device, and choose **OK**. The Android emulator opens, and the sample app launches in the virtual device. You can choose **User Sign-in** or **User Engagement** to learn more about these features.



3. In the Android emulator, sign in to your Google account so that the sample app can receive push notifications from Amazon Pinpoint:
 - a. Open **Settings**.
 - b. Choose **Accounts**.
 - c. Choose **Add account, Google**, and provide your Google email address and password.

Testing the Sample App With Amazon Pinpoint

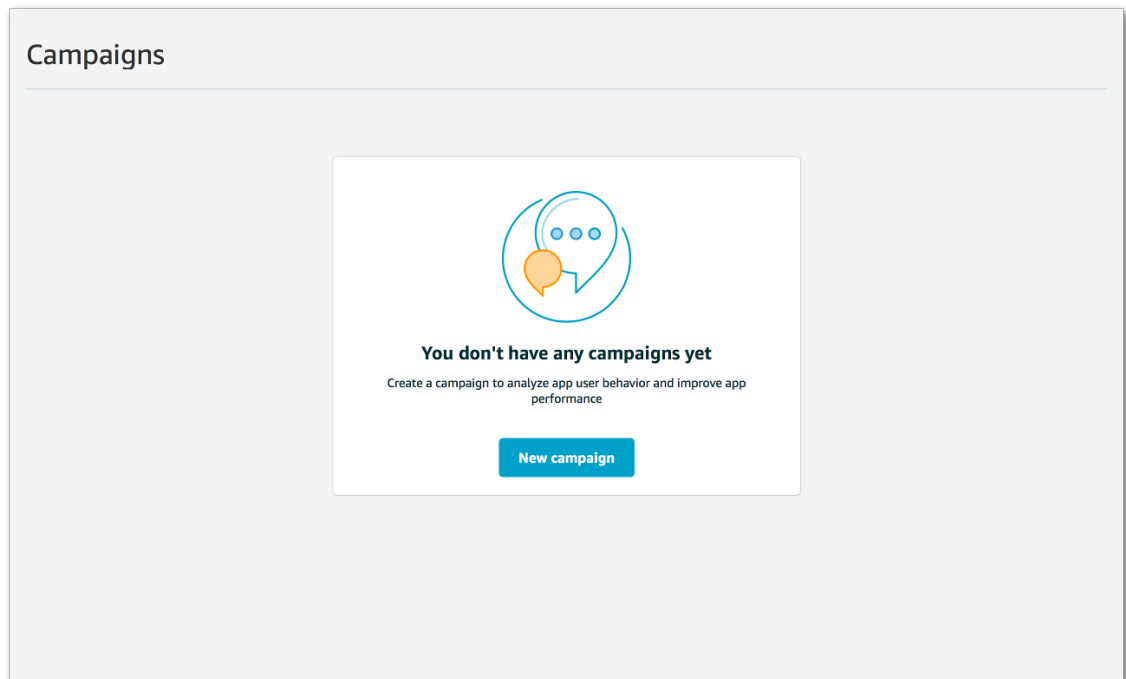
To verify that your AWS Mobile Hub sample app can receive push notifications from Amazon Pinpoint, use Amazon Pinpoint to create a simple campaign and send a message to the app.

Prerequisite

To complete this task, you need a Mobile Hub sample app with Amazon Pinpoint support. To receive push notifications, you must build the sample app and run it. For more information, see [Getting Started With iOS Apps \(p. 16\)](#) or [Getting Started With Android Apps \(p. 23\)](#).

To send a push notification to the sample app

1. Sign in to the AWS Management Console and open the Amazon Pinpoint console at <https://console.aws.amazon.com/pinpoint/>.
2. On the **Apps** page, choose the app that you created in AWS Mobile Hub.
3. Unless you have already created a campaign for your app, the console shows the **Campaigns** page. Choose **New campaign** to create a campaign.



4. The **Create a campaign** page displays at the **Details** step.
5. For **Name your campaign**, type a name to make the campaign easy to recognize later.
6. For **Choose the type of your campaign**, choose **Standard campaign**.
7. Choose **Next step**, and the console displays the **Segments** step.
8. Keep the default options for the segment, which include users who have used your app in the last 30 days. The **Segment estimate** count indicates how many user endpoints your campaign will deliver messages to. Because you built the sample app for iOS or Android and ran that app, you will see a segment estimate of 1 if your app successfully registered as an endpoint with Amazon Pinpoint.
9. Choose **Next step**. The console displays the **Message** step.
10. Keep **Standard notification** selected, and type a title and message for your test push notification.
11. For **Action**, keep **Open app**.
12. Choose **Next step**. The console displays the **Schedule** step.
13. Choose **Immediate**, and then choose **Next step**. The console displays the **Review and launch** step.
14. Choose **Launch campaign**. Amazon Pinpoint delivers your test push notification to the sample app on your test device.

Integrating Amazon Pinpoint with a Mobile App

To take advantage of Amazon Pinpoint, your mobile app must incorporate code that records and submits analytics data to the Amazon Pinpoint service. This section describes concepts you need to know to incorporate Amazon Pinpoint into a mobile app using the AWS Mobile SDK APIs.

To integrate the code that makes data collection possible, use the APIs provided by the AWS Mobile SDK for iOS and AWS Mobile SDK for Android. When working on platforms other than those supported by the AWS Mobile SDK, use the REST API to access Amazon Pinpoint.

Topics

- [Integrating Amazon Pinpoint With iOS Apps \(p. 32\)](#)
- [Integrating Amazon Pinpoint with Android Apps \(p. 40\)](#)

Integrating Amazon Pinpoint With iOS Apps

To use Amazon Pinpoint with an iOS app, you must integrate code that connects your app to Amazon Pinpoint. This enables the app to send event data used by the service to track user activity and gather engagement metrics.

Topics

- [Setting Up the AWS Mobile SDK for iOS \(p. 32\)](#)
- [Initializing the Amazon Pinpoint Client \(p. 34\)](#)
- [Registering Endpoint Profiles \(p. 37\)](#)
- [Reporting Events \(p. 37\)](#)
- [Setting Up Deep Linking \(p. 38\)](#)

Setting Up the AWS Mobile SDK for iOS

Before modifying your app to use Amazon Pinpoint, set up the AWS Mobile SDK for iOS.

To set up the AWS Mobile SDK for iOS

1. Open your app project in Xcode.
2. Context-click in your project tree view, and select **Add files to "<project name>"...** from the context menu.
3. Locate the `AWSCore.framework` and `AWSPinpoint.framework` files from the downloaded .zip file. Select the files, and choose **Add**.
4. Open a target for your project, select **Build Phases**, expand **Link Binary With Libraries**. Then, click the + button, and add:
 - libsqlite3
 - libz
 - SystemConfiguration.framework
5. Open the target for your project, choose **General** and expand the **Embedded Binaries** section. Then, choose the plus icon (+).
6. In the window that opens, select the `AWSCore` and `AWSPinpoint` frameworks.
7. Choose **Capabilities**. Enable push notification and remote notifications in background modes.
8. Edit your `Info.plist` file to include your AWS information:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>AWS</key>
<dict>
<key>CredentialsProvider</key>
<dict>
<key>CognitoIdentity</key>
<dict>
<key>Default</key>
<dict>
<key>PoolId</key>
<string>IDENTITY_POOL_ID</string>
<key>Region</key>
<string>us-east-1</string>
</dict>
</dict>
</dict>
<key>PinpointAnalytics</key>
<dict>
<key>Default</key>
<dict>
<key>AppId</key>
<string>APP_ID</string>
<key>Region</key>
<string>us-east-1</string>
</dict>
</dict>
<key>PinpointTargeting</key>
<dict>
<key>Default</key>
<dict>
<key>Region</key>
<string>us-east-1</string>
</dict>
</dict>
</dict>
</dict>
```

9. If, when a user taps a push notification sent to your app, you want your app to open a URL, add the following:

```
<key>LSApplicationQueriesSchemes</key>
<array>
  <string>http</string>
  <string>https</string>
</array>
```

Initializing the Amazon Pinpoint Client

After you have included AWS Mobile SDK for iOS support in your app, which enables the app to call AWS services, modify the app code to accomplish these tasks:

- Initialize the Amazon Pinpoint client
- Register for push notifications
- Handle notification callbacks

Initialize the Amazon Pinpoint Client

Add the following import to ApplicationDelegate:

Objective-C

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

Swift

```
import AWSPinpoint
```

There are several approaches you can use to initialize the Amazon Pinpoint client. Examples of each approach follow.

To initialize the Amazon Pinpoint client using the default initializer

- In the `application:didFinishLaunchingWithOptions:` method in the `ApplicationDelegate` for your app, initialize the Amazon Pinpoint client with the Amazon Pinpoint completion block and pass in the `launchOptions` dictionary to the Amazon Pinpoint initialization block as follows:

Objective-C

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:
(NSDictionary *)launchOptions {
    _pinpoint = [AWSPinpoint
pinpointWithConfiguration:[AWSPinpointConfiguration
defaultPinpointConfigurationWithLaunchOptions:launchOptions]];
    ...
}
```

Swift

```
func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[NSObject: AnyObject]?) -> Bool {
```

```
pinpoint =  
AWSPinpoint.init(configuration:AWSPinpointConfiguration.defaultPinpointConfigurationWithLaunchOptions)
```

To initialize the Amazon Pinpoint client using a custom configuration

- In the application:didFinishLaunchingWithOptions: method in the AppDelegate for your app, initialize the Amazon Pinpoint client using a custom client ID as follows:

Objective-C

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:  
(NSDictionary *)launchOptions {  
    AWSCognitoCredentialsProvider *cognitoCredentialsProvider =  
[[AWSCognitoCredentialsProvider alloc] initWithRegionType:AWSRegionUSEast1  
identityPoolId:@"IDENTITY_POOL_ID"];  
    AWSServiceConfiguration *pinpointConfig = [[AWSServiceConfiguration  
alloc] initWithRegion:AWSRegionUSEast1  
credentialsProvider:cognitoCredentialsProvider];  
    [[AWSServiceManager defaultManager]  
setDefaultServiceConfiguration:pinpointConfig];  
    AWSPinpointConfiguration *configuration = [[AWSPinpointConfiguration  
alloc] initWithAppId:@"APP_ID" launchOptions:launchOptions];  
    _pinpoint = [AWSPinpoint pinpointWithConfiguration:configuration];  
}
```

Swift

```
func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions:  
[NSObject: AnyObject]?) -> Bool {  
  
    let cognitoProvider:AWSCognitoCredentialsProvider =  
AWSCognitoCredentialsProvider.init(regionType: AWSRegionType.USEast1, identityPoolId:  
"IDENTITY_POOL_ID");  
    let serviceConfig:AWSServiceConfiguration = AWSServiceConfiguration.init(region:  
AWSRegionType.USEast1, credentialsProvider: cognitoProvider);  
    AWSServiceManager.defaultServiceManager().defaultServiceConfiguration =  
serviceConfig;  
    let pinpointConfig:AWSPinpointConfiguration = AWSPinpointConfiguration.init(appId:  
"APP_ID", launchOptions: launchOptions);  
    pinpoint = AWSPinpoint.init(configuration:pinpointConfig)
```

Registering for Push Notifications

Prompt the user to accept receiving push notifications during app launch or when the app requests receipt of push notifications. After the user grants permission, the app should re-register when launching the app because the device token can change.

Objective-C

```
UIUserNotificationType userNotificationTypes =  
(UIUserNotificationTypeAlert | UIUserNotificationTypeBadge |  
UIUserNotificationTypeSound);  
UIUserNotificationSettings *notificationSettings =  
[UIUserNotificationSettings settingsForTypes:userNotificationTypes  
categories:nil];  
[[UIApplication sharedApplication]  
registerUserNotificationSettings:notificationSettings];
```

Swift

```
let settings = UIUserNotificationSettings(forTypes: [.Sound, .Alert, .Badge], categories: nil)
UIApplication.sharedApplication().registerUserNotificationSettings(settings)
UIApplication.shared().registerForRemoteNotifications()
```

If you are using iOS 10 or greater with the UserNotification framework, add the following:

Objective-C

```
[UNUserNotificationCenter currentNotificationCenter]
  requestAuthorizationWithOptions:(UNAuthorizationOptionAlert + UNAuthorizationOptionSound)
  completionHandler:^(BOOL granted, NSError * _Nullable error) {
    // Enable or disable features based on authorization.
  }];
[[UIApplication sharedApplication] registerForRemoteNotifications];
```

Swift

```
UNUserNotificationCenter.current().requestAuthorization(options:[.badge, .alert, .sound])
{ (granted, error) in
    // Enable or disable features based on authorization.
}
UIApplication.shared().registerForRemoteNotifications()
```

Handle Notification Callbacks

To enable Amazon Pinpoint to report the notifications that are opened for a campaign, you must intercept some notification callbacks.

In the `application:didRegisterForRemoteNotificationsWithDeviceToken:` method in the `AppDelegate` for your app, call the interceptor:

Objective-C

```
- (void)application:(UIApplication *)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken {
    [_pinpoint.notificationManager
    interceptDidRegisterForRemoteNotificationsWithDeviceToken:deviceToken];
}
```

Swift

```
func application(application: UIApplication,
didRegisterForRemoteNotificationsWithDeviceToken deviceToken: NSData) {
    pinpoint!.notificationManager.interceptDidRegisterForRemoteNotificationsWithDeviceToken(deviceToken)
}
```

In the `application:didReceiveNotification:fetchCompletionHandler:` method in the `AppDelegate` for your app, call the interceptor:

Objective-C

```
- (void) application:(UIApplication *)application didReceiveRemoteNotification:
(nonnull NSDictionary *)userInfo fetchCompletionHandler:(nonnull void (^)(
UIBackgroundFetchResult))completionHandler {
    [_pinpoint.notificationManager
    interceptDidReceiveRemoteNotification:userInfofetchCompletionHandler:completionHandler];
    completionHandler(UIBackgroundFetchResultNewData);
}
```


Swift

```
func application(application: UIApplication, didReceiveRemoteNotification
userInfo: [NSObject : AnyObject], fetchCompletionHandler completionHandler:
(UIBackgroundFetchResult) -> Void) {
    pinpoint!.notificationManager.interceptDidReceiveRemoteNotification(userInfo,
    fetchCompletionHandler: completionHandler)
```

If you are using iOS 10 or greater with the UserNotification framework, add the following:

Objective-C

```
- (void)userNotificationCenter:(UNUserNotificationCenter *)center
didReceiveNotificationResponse:(UNNotificationResponse *)response
withCompletionHandler:(void (^)(void))completionHandler {
    [[[AWSMobileClient sharedInstance] pinpoint].notificationManager
interceptDidReceiveRemoteNotification:response.notification.request.content.userInfo
fetchCompletionHandler:^(UIBackgroundFetchResult result) {}];
```

Swift

```
@available(iOS 10.0, *)
func userNotificationCenter(center: UNUserNotificationCenter,
didReceiveNotificationResponse response: UNNotificationResponse, withCompletionHandler
completionHandler: () -> Void) {
    pinpoint!.notificationManager.interceptDidReceiveRemoteNotification(response.notification.request.content
{ (UIBackgroundFetchResult) in}
```

Registering Endpoint Profiles

The SDK automatically updates the endpoint profile when the session is started (when the app comes to the foreground). The endpoint profile is used by Amazon Pinpoint to segment and target devices based on attributes of the device, as well as custom attributes you define.

Adding Custom Attributes

You can add custom attributes to the device profile as shown in the following example, which adds a `favoriteTeams` custom attribute.

Objective-C

```
[[_pinpoint.targetingClient addAttribute:@"Lakers",@"Clippers"] forKey:@"favoriteTeams"];
[_pinpoint.targetingClient updateEndpointProfile];
```

Swift

```
let pinpointTargetingClient = AWSMobileClient.sharedInstance.pinpoint!.targetingClient
pinpointTargetingClient.addAttribute(["Lakers", "Clippers"], forKey: "favoriteTeams")
pinpointTargetingClient.updateEndpointProfile()
```

Reporting Events

The AWS Mobile SDK for iOS automatically records session events when the app goes into the foreground and background. This is configurable and you may choose to implement your own session handling. You can also record custom events and monetization events.

Reporting a Custom Event

To record and submit events, use the `AnalyticsClient`.

A custom event must have an event type, and you can add custom attributes and metrics to that event.

Objective-C

```
AWSPinpointEvent *event = _pinpoint.analyticsClient
createEventWithEventType:@"MyCustomEvent"];
[event addAttribute:@"MyAttributeValue1" forKey:@"MyAttribute1"];
[event addAttribute:@"MyAttributeValue2" forKey:@"MyAttribute2"];
[event addMetric:[NSNumber numberWithInt:(arc4random() % 65535)] forKey:@"MyMetric"];
_pinpoint.analyticsClient recordEvent:event];
_pinpoint.analyticsClient submitEvents];
```

Swift

```
let pinpointAnalyticsClient = pinpoint!.analyticsClient
let event = pinpointAnalyticsClient.createEventWithEventType("MyCustomEvent")
    event.addAttribute("MyAttributeValue1", forKey: "MyAttribute1")
    event.addAttribute("MyAttributeValue2", forKey: "MyAttribute2")
    event.addMetric(Int(arc4random() % 65535), forKey: "MyMetric")
pinpointAnalyticsClient.recordEvent(event)
pinpointAnalyticsClient.submitEvents()
```

Reporting a Monetization Event

Use the `AnalyticsClient` to record and submit events.

A custom event must have an event type, and you can add custom attributes and metrics to that event.

In the following example, the `AWSMobileClient` class is provided in the AWS Mobile Hub sample code to reference the Amazon Pinpoint object.

Objective-C

```
AWSPinpointEvent *event = _pinpoint.analyticsClient
createVirtualMonetizationEventWithProductId:@"PRODUCT_ID"
withItemPrice:1.00
withQuantity:1
withCurrency:@"USD"];
[[AWSMobileClient sharedInstance].pinpoint.analyticsClient recordEvent:event];
[[AWSMobileClient sharedInstance].pinpoint.analyticsClient submitEvents];
```

Swift

```
let pinpointAnalyticsClient = pinpoint!.analyticsClient
let event =
    pinpoint!.analyticsClient.createVirtualMonetizationEventWithProductId("PRODUCT_ID",
withItemPrice: 1.00, withQuantity: 1, withCurrency: "USD")
pinpointAnalyticsClient.recordEvent(event)
pinpointAnalyticsClient.submitEvents()
```

Setting Up Deep Linking

Amazon Pinpoint campaigns can take one of three actions when a user taps a notification. One of those possible actions is a deep link, which opens the app to a specified activity.

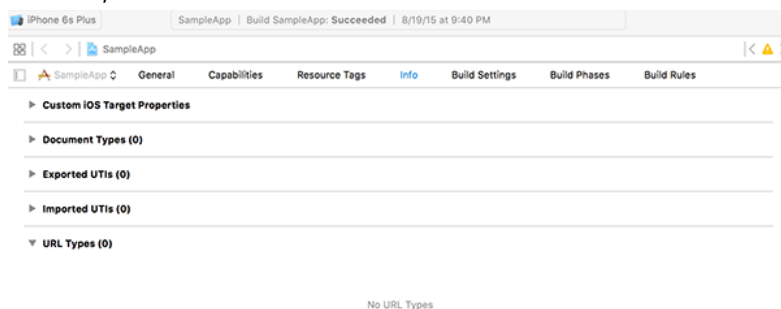
Registering a Custom URL Scheme

To specify a destination activity for deep links, the app must have set up deep linking. This setup requires registering a custom URL scheme the deep links will use. To register a custom URL identifier, go to your Xcode project's target **Info** tab and expand the **URL Types** section.

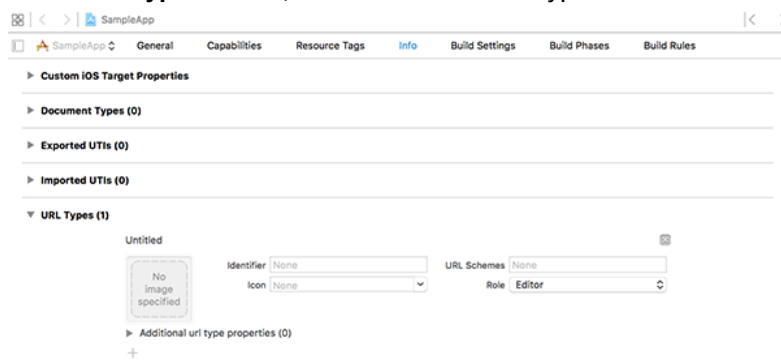
To open the app via a `pinpoint://` URL scheme you need to assign a unique identifier to the scheme. Apple recommends reverse DNS notation to avoid name collisions on the platform. The following example uses `com.exampleCorp.exampleApp`:

To register a custom URL scheme in Xcode:

1. In Xcode, select the **Info** tab.



2. In the **URL Types** section, select + to add a URL type.



3. Enter the reverse DNS notation identifier for this URL type in **Identifier**.
4. Enter the URL you want to use for your app in **URL Schemes**.
5. Save the project.

After this custom URL scheme is registered, test it in the iOS simulator. Open Safari and navigate to your custom URL; in this example, `pinpoint://`. Your app launches and opens to its home screen.

Listening for Custom URLs

To direct the app to a specific view, implement a callback in your `AppDelegate` that is called when your app launches from a deep link. The scheme launches the app, using the host and path to go to a separate screen within the app. The following example from the sample application shows how to implement a deep link in your application. This link takes the user to `page1` if it receives a `pinpoint://deeplink/page1` link.

Objective-C

```
-(BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:
(NSString *)sourceApplication annotation:(id)annotation {
    NSLog(@"%s", __func__);
    if([[url host] isEqualToString:@"deeplink"]) {
        if([[url path] isEqualToString:@"/page1"]) {
            dispatch_after(dispatch_time(DISPATCH_TIME_NOW, (int64_t)(1 * NSEC_PER_SEC)),
            dispatch_get_main_queue(), ^{
                [((UINavigationController*)self.window.rootViewController)
                popToRootViewControllerAnimated:NO];
                [((UINavigationController*)self.window.rootViewController).viewControllers
                firstObject] performSegueWithIdentifier:@"PAGE1_SEGUE"
                sender:self.window.rootViewController];
            });
        } else if ([[url path] isEqualToString:@"/page2"]) {
            dispatch_after(dispatch_time(DISPATCH_TIME_NOW, (int64_t)(1 * NSEC_PER_SEC)),
            dispatch_get_main_queue(), ^{
                [((UINavigationController*)self.window.rootViewController)
                popToRootViewControllerAnimated:NO];
                [((UINavigationController*)self.window.rootViewController).viewControllers
                firstObject] performSegueWithIdentifier:@"PAGE2_SEGUE"
                sender:self.window.rootViewController];
            });
        } else {
            return NO;
        }
    } else {
        return NO;
    }
    return YES;
}
```

Integrating Amazon Pinpoint with Android Apps

To use Amazon Pinpoint with an Android app, you must integrate code that connects your app to Amazon Pinpoint. This enables the app to send event data used by the service to track user activity and gather engagement metrics.

Topics

- [Setting up the AWS Mobile SDK for Android \(p. 40\)](#)
- [Initializing the Amazon Pinpoint Client \(p. 41\)](#)
- [Managing Sessions \(p. 44\)](#)
- [Registering Endpoint Profiles \(p. 45\)](#)
- [Reporting Events \(p. 45\)](#)
- [Setting up Deep Linking \(p. 46\)](#)

Setting up the AWS Mobile SDK for Android

Before modifying your app to use Amazon Pinpoint, you need to set up the AWS Mobile SDK for Android.

To set up AWS Mobile SDK for Android support in your app project

1. Copy these AWS Mobile SDK for Android .jar files into the app\libs directory of your Android Studio project:
 - aws-android-sdk-core-2.3.4.jar

- aws-android-sdk-cognito-2.3.4.jar
 - aws-android-sdk-pinpoint-2.3.4.jar
2. Make sure your `build.gradle` file contains the following build dependencies:

```
compile fileTree(include: ['*.jar'], dir: 'libs')
compile 'com.google.firebase:firebase-messaging:9.6.0'
```

Or you can list and compile each `.jar` file individually, as follows:

```
compile files('libs/aws-android-sdk-core-2.3.4.jar')
compile files('libs/aws-android-sdk-cognito-2.3.4.jar')
compile files('libs/aws-android-sdk-pinpoint-2.3.4.jar')
```

3. After the dependencies brackets, push:

```
apply plugin: 'com.google.gms.google-services'
```

Initializing the Amazon Pinpoint Client

After you have included AWS Mobile SDK for Android support in your app, which enables the app to call AWS services, modify the app code to accomplish these tasks:

- Set up the manifest file
- Initialize the Amazon Pinpoint client
- Register the GCM token
- Handle the GCM message

Set up the Manifest File

If you are using Firebase Cloud Messaging (FCM), add the following entries to your `AndroidManifest.xml` file before the `<application>` tag.

```
<receiver
    android:name="com.amazonaws.mobileconnectors.pinpoint.targeting.notification.PinpointNotificationReceiver"
    android:exported="false" >
    <intent-filter>
        <action android:name="com.amazonaws.intent.fcm.NOTIFICATION_OPEN" />
    </intent-filter>
</receiver>
```

If you are using Google Cloud Messaging(GCM), to provide permissions for your app to register, receive, and respond to Google Cloud Messaging (GCM) notifications, add the following entries to your `AndroidManifest.xml` file before the `<application>` tag.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
<permission android:name="PACKAGE_NAME.permission.C2D_MESSAGE"
    android:protectionLevel="signature" />
```

```
<uses-permission android:name="PACKAGE_NAME.permission.C2D_MESSAGE" />
```

Install a listener for push notification messages from Google servers. Add the following entries to your `AndroidManifest.xml` file inside the `<application>` tag:

```
<receiver  
    android:name="com.google.android.gms.gcm.GcmReceiver"  
    android:exported="true"  
    android:permission="com.google.android.c2dm.permission.SEND" >  
    <intent-filter>  
        <action android:name="com.google.android.c2dm.intent.RECEIVE" />  
        <category android:name="@string/google_cloud_messaging_package" />  
    </intent-filter>  
</receiver>
```

Register a service that extends the Google `GcmListenerService` to listen for push notifications. Here is an example of such a service called `PushListenerService`.

```
<service  
    android:name="PACKAGE_NAME.PushListenerService"  
    android:exported="false" >  
    <intent-filter>  
        <action android:name="com.google.android.c2dm.intent.RECEIVE" />  
    </intent-filter>  
</service>
```

Initialize the Amazon Pinpoint Client

One of the first things to do when modifying your app is to initialize the Amazon Pinpoint client. Begin by adding this import to your main activity:

```
import com.amazonaws.mobileconnectors.pinpoint.*;
```

Depending on how your app authenticates calls, you may also need to add the following import:

```
import com.amazonaws.regions.Regions;
```

Important

For simplicity, the code in these examples initializes the Amazon Pinpoint client in the `onCreate` method. This approach can take several seconds to execute. In production code, this initialization is best done in a background thread. For more information about how to do this, see the Google Developer website.

To initialize the Amazon Pinpoint client using the default initializer

- Initialize the Amazon Pinpoint client as follows:

```
CognitoCachingCredentialsProvider cognitoCachingCredentialsProvider = new  
    CognitoCachingCredentialsProvider(context, "IDENTITY_POOL_ID", Regions.US_EAST_1);  
  
PinpointConfiguration config = new PinpointConfiguration(context, "APP_ID",  
    Regions.US_EAST_1, cognitoCachingCredentialsProvider);  
  
this.pinpointManager = new PinpointManager(config);
```

Registering the GCM Token

You must also register the Google Cloud Messaging (GCM) token with the Amazon Pinpoint client.

In the following example, the `AWSMobileClient` class is provided in the AWS Mobile Hub sample code to reference the Amazon Pinpoint object.

```
InstanceID instanceID = InstanceID.getInstance(this);
String gcmToken =
    instanceID.getToken(
        getString(R.string.gcm_defaultSenderId),
        GoogleCloudMessaging.INSTANCE_ID_SCOPE,
        null);
AWSMobileClient.defaultMobileClient().getPinpointManager().getNotificationClient().registerGCMDeviceToken
```

If you are using Firebase Cloud Messaging (FCM), call this object inside your class that extends `FirebaseInstanceIdService`:

```
/**
 * Called if InstanceID token is updated. This may occur if the security of
 * the previous token has been compromised. Note that this is called when the InstanceID
 * token
 * is initially generated so this is where you would retrieve the token.
 */
// [START refresh_token]
@Override
public void onTokenRefresh() {
    // Get updated InstanceID token.
    String refreshedToken = FirebaseInstanceId.getInstance().getToken();
    Log.d(TAG, "Refreshed token: " + refreshedToken);

    // If you want to send messages to this application instance or
    // manage this apps subscriptions on the server side, send the
    // Instance ID token to your app server.

    AWSMobileClient.defaultMobileClient().getPinpointManager().getNotificationClient().registerGCMDeviceToken
}
// [END refresh_token]
```

Handling the GCM Message

You must add a hook to handle the GCM message. You do this in the class in your app that extends `GcmListenerService` in the `onMessageReceived` method:

```
@Override
public void onMessageReceived(final String from, final Bundle data) {
    AWSMobileClient.initializeMobileClientIfNecessary(this.getApplicationContext());
    final NotificationClient notificationClient = AWSMobileClient.defaultMobileClient()
        .getPinpointManager().getNotificationClient();

    NotificationClient.CampaignPushResult pushResult =
        notificationClient.handleGCMCampaignPush(from, data, this.getClass());
}
```

Handling the FCM Message

You must add a hook to handle the FCM message. You do this in the class in your app that extends `FirebaseMessagingService` in the `onMessageReceived` method:

```
/**
 * Called when message is received.
 *
 * @param remoteMessage Object representing the message received from Firebase Cloud
 Messaging.
 */
// [START receive_message]
@Override
public void onMessageReceived(RemoteMessage remoteMessage) {
    AWSMobileClient.defaultMobileClient().getPinpointManager().getNotificationClient().handleFCMCampaignPushNotification(
        remoteMessage.getData());
}
// [END receive_message]
```

Managing Sessions

As users engage with your app, it reports information about app sessions to Amazon Pinpoint, such as session start times, session end times, and events that occur during sessions. To report this information, your app must include methods that handle events as your app enters the foreground and the background on the user's Android device.

When you use AWS Mobile Hub to create a project for your Amazon Pinpoint app, Mobile Hub provides a sample app that demonstrates how to integrate with Amazon Pinpoint. The Android version of the sample app includes the `AbstractApplicationLifecycleHelper` class to help you manage app sessions. Include this class in your Android app package.

For more information about creating the Mobile Hub sample app, see [Getting Started With Android Apps \(p. 23\)](#).

After you include the `AbstractApplicationLifecycleHelper` class, implement the abstract methods, `applicationEnteredForeground` and `applicationEnteredBackground`, in the `Application` file in your app package. These methods enable your app to report the following information to Amazon Pinpoint:

- Session start times (when the app enters the foreground).
- Session end times (when the app enters the background).
- The events that occur during the app session, such as monetization events. This information is reported when the app enters the background.

The following example shows how to implement the `applicationEnteredForeground` and `applicationEnteredBackground` abstract methods:

```
applicationLifecycleHelper = new AbstractApplicationLifecycleHelper(this) {
    @Override
    protected void applicationEnteredForeground() {
        final PinpointManager pinpointManager = AWSMobileClient.defaultMobileClient()
            .getPinpointManager();
        pinpointManager.getSessionClient().startSession();
        // Handles events that occur when your app enters the foreground.
    }

    @Override
    protected void applicationEnteredBackground() {
        Log.d(LOG_TAG, "Detected application has entered the background.");
        final PinpointManager pinpointManager = AWSMobileClient.defaultMobileClient()
            .getPinpointManager();
        pinpointManager.getSessionClient().stopSession();
        pinpointManager.getAnalyticsClient().submitEvents();
        // Handles events that occur when your app enters the background.
    }
}
```



```
}  
};
```

Registering Endpoint Profiles

The AWS Mobile SDK for Android automatically updates the endpoint profile when the session is started (when the app comes to the foreground). The endpoint profile is used by Amazon Pinpoint to segment and target devices based on attributes of the device, as well as custom attributes you can define.

Adding Custom Attributes

You can add custom attributes to the device profile as shown in the following example, which adds a `favoriteTeams` custom attribute.

In the following example, the `AWSMobileClient` class is provided in the AWS Mobile Hub sample code to reference the Amazon Pinpoint object.

```
AWSMobileClient.defaultMobileClient()  
    .getPinpointManager().getTargetingClient().addAttribute("favoriteTeams",  
    Arrays.asList(new String[]{"Lakers", "Clippers"}));  
AWSMobileClient.defaultMobileClient()  
    .getPinpointManager().getTargetingClient().updateEndpointProfile();
```

Reporting Events

To record and submit events, use the `AnalyticsClient`.

Reporting Custom Events

A custom event must have an event type, and you can add custom attributes and metrics to that event.

In the following example, the `AWSMobileClient` class is provided in the AWS Mobile Hub sample code to reference the Amazon Pinpoint object.

```
final AnalyticsEvent event = AWSMobileClient.defaultMobileClient()  
    .getPinpointManager().getAnalyticsClient().createEvent("MyCustomEvent")  
    // A music app use case might include attributes such as:  
    // .withAttribute("Playlist", "Amazing Songs 2016")  
    // .withAttribute("Artist", "Various")  
    // .withMetric("Song playtime", playTime);  
    .withAttribute("MyAttribute1", "MyAttributeValue1")  
    .withAttribute("MyAttribute2", "MyAttributeValue2")  
    .withMetric("MyMetric1", Math.random());  
  
AWSMobileClient.defaultMobileClient()  
    .getPinpointManager().getAnalyticsClient().recordEvent(event);  
AWSMobileClient.defaultMobileClient()  
    .getPinpointManager().getAnalyticsClient().submitEvents();
```

Recording a Monetization Event

To record and submit monetization events, use the `AnalyticsClient`.

```
final AnalyticsEvent event =  
    GooglePlayMonetizationEventBuilder.create(AWSMobileClient.defaultMobileClient()  
        .getPinpointManager().getAnalyticsClient())
```

```
.withCurrency("USD")
.withItemPrice(1.00)
.withProductId("PRODUCT_ID")
.withQuantity(1.0)
.withTransactionId("TRANSACTION_ID").build();
AWSMobileClient.defaultMobileClient()
    .getPinpointManager().getAnalyticsClient().recordEvent(event);
AWSMobileClient.defaultMobileClient()
    .getPinpointManager().getAnalyticsClient().submitEvents();
```

Setting up Deep Linking

Amazon Pinpoint campaigns can take one of three actions when a user taps a notification. One of those possible actions is a deep link, which opens the app to a specified activity.

To specify a destination activity for deep links, the app must have set up deep linking. This setup requires an intent filter that registers a URL scheme the deep links will use. After the app creates an intent filter, the data provided by the intent determines the activity to render.

Creating an Intent Filter

Begin to set up deep linking by creating an intent filter in your `AndroidManifest.xml` file. For example:

```
<!-- This activity allows your application to receive a deep link that navigates directly
to the
"Deeplink Page"-->
<activity
    android:name=".DeepLinkActivity"
    android:label="A deeplink!" >
    <intent-filter android:label="inAppReceiver">
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />
        <!-- Accepts URIs of type "pinpoint://deeplink" -->
        <data android:scheme="pinpoint"
            android:host="deeplink" />
    </intent-filter>
</activity>
```

The data element in the previous example registers a URL scheme, `pinpoint://`, as well as the host, `deeplink`. As a result, when given a URL in the form of `pinpoint://deeplink`, the manifest is prepared to execute the action.

Handling the Intent

Next, set up an intent handler to present the screen associated with the registered URL scheme and host. Intent data is retrieved in the `onCreate()` method, which then can use `Uri` data to create an activity. The following example shows an alert and tracks an event.

```
public class DeeplinkActivity extends Activity {

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        if (getIntent().getAction() == Intent.ACTION_VIEW) {
            Uri data = getIntent().getData();

            if (data != null) {
```

```
        // show an alert with the "custom" param
        new AlertDialog.Builder(this)
            .setTitle("An example of a Deeplink")
            .setMessage("Found custom param: "
+data.getQueryParameter("custom"))
            .setPositiveButton(android.R.string.yes, new
DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {
                    dialog.dismiss();
                }
            })
            .setIcon(android.R.drawable.ic_dialog_alert)
            .show();
    }
}
}
```

Adding Endpoints

An *endpoint* uniquely identifies a user device to which you can send push notifications with Amazon Pinpoint. You can assign attributes to endpoints to describe your users.

For more information, see [Endpoints](#).

If your app is enabled with Amazon Pinpoint support, your app automatically registers an endpoint with Amazon Pinpoint when a new user opens the app. You can also add endpoints programmatically with the Amazon Pinpoint API or the AWS SDK for Java.

Adding Endpoints With the AWS SDK for Java

The following example demonstrates how to add endpoints to Amazon Pinpoint with the AWS SDK for Java.

```
import com.amazonaws.services.pinpoint.AmazonPinpointClient;
import com.amazonaws.services.pinpoint.model.EndpointDemographic;
import com.amazonaws.services.pinpoint.model.EndpointLocation;
import com.amazonaws.services.pinpoint.model.EndpointRequest;
import com.amazonaws.services.pinpoint.model.EndpointResponse;
import com.amazonaws.services.pinpoint.model.EndpointUser;
import com.amazonaws.services.pinpoint.model.GetEndpointRequest;
import com.amazonaws.services.pinpoint.model.GetEndpointResult;
import com.amazonaws.services.pinpoint.model.UpdateEndpointRequest;
import com.amazonaws.services.pinpoint.model.UpdateEndpointResult;

import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.UUID;

public class PinpointEndpointSample {

    public EndpointResponse createEndpoint(AmazonPinpointClient client, String appId) {
        String endpointId = UUID.randomUUID().toString();
        System.out.println("Endpoint ID: " + endpointId);

        EndpointRequest endpointRequest = createEndpointRequestData();
```

```
UpdateEndpointRequest updateEndpointRequest = new UpdateEndpointRequest()
    .withApplicationId(appId)
    .withEndpointId(endpointId)
    .withEndpointRequest(endpointRequest);

UpdateEndpointResult updateEndpointResponse =
client.updateEndpoint(updateEndpointRequest);
System.out.println("Update Endpoint Response: " +
updateEndpointResponse.getMessageBody());

GetEndpointRequest getEndpointRequest = new GetEndpointRequest()
    .withApplicationId(appId)
    .withEndpointId(endpointId);
GetEndpointResult getEndpointResult = client.getEndpoint(getEndpointRequest);

System.out.println("Got Endpoint: " +
getEndpointResult.getEndpointResponse().getId());
return getEndpointResult.getEndpointResponse();
}

private EndpointRequest createEndpointRequestData() {

HashMap<String, List<String>> customAttributes = new HashMap<>();
List<String> favoriteTeams = new ArrayList<>();
favoriteTeams.add("Lakers");
favoriteTeams.add("Warriors");
customAttributes.put("team", favoriteTeams);

EndpointDemographic demographic = new EndpointDemographic()
    .withAppVersion("1.0")
    .withMake("apple")
    .withModel("iPhone")
    .withModelVersion("7")
    .withPlatform("ios")
    .withPlatformVersion("10.1.1")
    .withTimezone("Americas/Los_Angeles");

EndpointLocation location = new EndpointLocation()
    .withCity("Los Angeles")
    .withCountry("US")
    .withLatitude(34.0)
    .withLongitude(-118.2)
    .withPostalCode("90068")
    .withRegion("CA");

Map<String,Double> metrics = new HashMap<>();
metrics.put("health", 100.00);
metrics.put("luck", 75.00);

EndpointUser user = new EndpointUser()
    .withUserId(UUID.randomUUID().toString());

EndpointRequest endpointRequest = new EndpointRequest()
    .withAddress(UUID.randomUUID().toString())
    .withAttributes(customAttributes)
    .withChannelType("APNS")
    .withDemographic(demographic)
    .withEffectiveDate(new Date().toString())
    .withLocation(location)
    .withMetrics(metrics)
    .withOptOut("NONE")
    .withRequestId(UUID.randomUUID().toString())
    .withUser(user);

return endpointRequest;
}
```

```
}  
}
```

When you run this example, the following is printed to the console window of your IDE.

```
Endpoint ID: 37d321e8-5419-4fa8-86ba-698905f262a4  
Update Endpoint Response: {Message: Accepted, RequestID: 74ef5959-b4d7-11e6-  
ae27-25eb3a23dee7}  
Get Endpoint ID: 37d321e8-5419-4fa8-86ba-698905f262a4
```

Creating Segments

A user *segment* represents a subset of your users based on shared characteristics, such as how recently the users have used your app or which device platform they use. A segment designates which users receive the messages delivered by a campaign. Define segments so that you can reach the right audience when you want to invite users back to your app, make special offers, or otherwise increase user engagement and purchasing.

After you create a segment, you can use it in one or more campaigns. A campaign delivers tailored messages to the users in the segment.

For more information, see [Segments](#).

Topics

- [Building Segments \(p. 51\)](#)
- [Importing Segments \(p. 52\)](#)

Building Segments

To reach the intended audience for a campaign, build a segment based on the data reported by your app. For example, to reach users who haven't used your app recently, you can define a segment for users who haven't used your app in the last 7 days.

Building Segments With the AWS SDK for Java

The following example demonstrates how to build a segment with the AWS SDK for Java.

```
import com.amazonaws.services.pinpoint.AmazonPinpointClient;
import com.amazonaws.services.pinpoint.model.AttributeDimension;
import com.amazonaws.services.pinpoint.model.AttributeType;
import com.amazonaws.services.pinpoint.model.CreateSegmentRequest;
import com.amazonaws.services.pinpoint.model.CreateSegmentResult;
import com.amazonaws.services.pinpoint.model.RecencyDimension;
import com.amazonaws.services.pinpoint.model.SegmentBehaviors;
import com.amazonaws.services.pinpoint.model.SegmentDemographics;
import com.amazonaws.services.pinpoint.model.SegmentDimensions;
```

```
import com.amazonaws.services.pinpoint.model.SegmentLocation;
import com.amazonaws.services.pinpoint.model.SegmentResponse;
import com.amazonaws.services.pinpoint.model.WriteSegmentRequest;

import java.util.HashMap;
import java.util.Map;

public class PinpointSegmentSample {

    public SegmentResponse createSegment(AmazonPinpointClient client, String appId) {
        Map<String, AttributeDimension> segmentAttributes = new HashMap<>();
        segmentAttributes.put("Team", new
AttributeDimension().withAttributeType(AttributeType.INCLUSIVE).withValues("Lakers"));

        SegmentBehaviors segmentBehaviors = new SegmentBehaviors();
        SegmentDemographics segmentDemographics = new SegmentDemographics();
        SegmentLocation segmentLocation = new SegmentLocation();

        RecencyDimension recencyDimension = new RecencyDimension();
        recencyDimension.withDuration("DAY_30").withRecencyType("ACTIVE");
        segmentBehaviors.setRecency(recencyDimension);

        SegmentDimensions dimensions = new SegmentDimensions()
            .withAttributes(segmentAttributes)
            .withBehavior(segmentBehaviors)
            .withDemographic(segmentDemographics)
            .withLocation(segmentLocation);

        WriteSegmentRequest writeSegmentRequest = new WriteSegmentRequest()
            .withName("MySegment").withDimensions(dimensions);

        CreateSegmentRequest createSegmentRequest = new CreateSegmentRequest()
            .withApplicationId(appId).withWriteSegmentRequest(writeSegmentRequest);

        CreateSegmentResult createSegmentResult =
client.createSegment(createSegmentRequest);

        System.out.println("Segment ID: " +
createSegmentResult.getSegmentResponse().getId());

        return createSegmentResult.getSegmentResponse();
    }
}
```

When you run this example, the following is printed to the console window of your IDE:

```
Segment ID: 09cb2967a82b4a2fbab38fead8d1f4c4
```

Importing Segments

With Amazon Pinpoint, you can define a user segment by importing information about the user devices that belong to the segment.

Importing segments is useful if you have segments for your users outside of Amazon Pinpoint but you want to engage your users with Amazon Pinpoint campaigns.

When you import a segment, Amazon Pinpoint gets the segment's endpoints from Amazon Simple Storage Service (Amazon S3). Before you import, you add the endpoints to Amazon S3, and you create

an IAM role that grants Amazon Pinpoint access to Amazon S3. Then, you give Amazon Pinpoint the Amazon S3 location where the endpoints are stored, and Amazon Pinpoint adds each endpoint to the segment.

To create the IAM role, see [IAM Role for Importing Segments \(p. 75\)](#). For information about creating the Amazon S3 bucket, creating endpoint files, and importing a segment by using the console, see [Importing Segments](#) in the *Amazon Pinpoint User Guide*.

Importing a Segment

The following example demonstrates how to import a segment with the AWS SDK for Java.

```
import com.amazonaws.services.pinpoint.AmazonPinpointClient;
import com.amazonaws.services.pinpoint.model.AttributeDimension;
import com.amazonaws.services.pinpoint.model.AttributeType;
import com.amazonaws.services.pinpoint.model.CreateImportJobRequest;
import com.amazonaws.services.pinpoint.model.CreateImportJobResult;
import com.amazonaws.services.pinpoint.model.CreateSegmentRequest;
import com.amazonaws.services.pinpoint.model.CreateSegmentResult;
import com.amazonaws.services.pinpoint.model.Format;
import com.amazonaws.services.pinpoint.model.GetImportJobRequest;
import com.amazonaws.services.pinpoint.model.GetImportJobResult;
import com.amazonaws.services.pinpoint.model.GetSegmentRequest;
import com.amazonaws.services.pinpoint.model.GetSegmentResult;
import com.amazonaws.services.pinpoint.model.ImportJobRequest;
import com.amazonaws.services.pinpoint.model.JobStatus;
import com.amazonaws.services.pinpoint.model.RecencyDimension;
import com.amazonaws.services.pinpoint.model.SegmentBehaviors;
import com.amazonaws.services.pinpoint.model.SegmentDemographics;
import com.amazonaws.services.pinpoint.model.SegmentDimensions;
import com.amazonaws.services.pinpoint.model.SegmentLocation;
import com.amazonaws.services.pinpoint.model.SegmentResponse;
import com.amazonaws.services.pinpoint.model.WriteSegmentRequest;

import java.util.HashMap;
import java.util.Map;

public class PinpointImportSample {

    public SegmentResponse createImportSegment(AmazonPinpointClient client, String appId,
                                              String bucket, String key,
                                              String roleArn, String externalId) throws
    Exception {

        // Create the job.
        ImportJobRequest importRequest = new ImportJobRequest()
            .withDefineSegment(true)
            .withRegisterEndpoints(true)
            .withRoleArn(roleArn)
            .withExternalId(externalId)
            .withFormat(Format.JSON)
            .withS3Url("s3://" + bucket + "/" + key);

        CreateImportJobRequest jobRequest = new CreateImportJobRequest()
            .withImportJobRequest(importRequest)
            .withApplicationId(appId);

        CreateImportJobResult jobResponse = client.createImportJob(jobRequest);

        GetImportJobRequest fetchRequest = new GetImportJobRequest()
            .withApplicationId(appId)
            .withJobId(jobResponse.getImportJobResponse().getId());
```

```
// Wait for job to finish.

GetImportJobResult getJobResponse;

do {
    // Lets only check once every 10 seconds if done, busy waits are bad.

    Thread.sleep(10 * 1000);

    getJobResponse = client.getImportJob(fetchRequest);

    if
    (getJobResponse.getImportJobResponse().getJobStatus().equals(JobStatus.FAILED)) {
        throw new Exception("Failed to process import job successfully");
    }
} while(!
getJobResponse.getImportJobResponse().getJobStatus().equals(JobStatus.COMPLETED));

// Finally get the import segment that was created.

GetSegmentRequest segmentRequest = new GetSegmentRequest()
    .withApplicationId(appId)
    .withSegmentId(getJobResponse.getImportJobResponse().getId());

GetSegmentResult getSegmentResult = client.getSegment(segmentRequest);

// Print out what we got

System.out.println("Segment ID: " + getSegmentResult.getSegmentResponse().getId()
    + " with size " +
getSegmentResult.getSegmentResponse().getImportDefinition().getSize());

return getSegmentResult.getSegmentResponse();
}
}
```

Creating Campaigns

To help increase engagement between your app and its users, use Amazon Pinpoint to create and manage push notification campaigns that reach out to particular segments of users.

For example, your campaign might invite users back to your app who haven't run it recently or offer special promotions to users who haven't purchased recently.

A campaign sends a tailored message to a user segment that you specify. The campaign can send the message to all users in the segment, or you can allocate a holdout, which is a percentage of users who receive no messages.

You can set the campaign schedule to send the message once or at a recurring frequency, such as once a week. To prevent users from receiving the message at inconvenient times, the schedule can include a quiet time during which no messages are sent.

To experiment with alternative campaign strategies, set up your campaign as an A/B test. An A/B test includes two or more treatments of the message or schedule. Treatments are variations of your message or schedule. As your users respond to the campaign, you can view campaign analytics to compare the effectiveness of each treatment.

For more information, see [Campaigns](#).

Creating Standard Campaigns

A standard campaign sends a custom push notification to a specified segment according to a schedule that you define.

Creating Campaigns With the AWS SDK for Java

The following example demonstrates how to create a campaign with the AWS SDK for Java.

```
import com.amazonaws.services.pinpoint.AmazonPinpointClient;
import com.amazonaws.services.pinpoint.model.Action;
import com.amazonaws.services.pinpoint.model.CampaignResponse;
import com.amazonaws.services.pinpoint.model.CreateCampaignRequest;
import com.amazonaws.services.pinpoint.model.CreateCampaignResult;
import com.amazonaws.services.pinpoint.model.Message;
import com.amazonaws.services.pinpoint.model.MessageConfiguration;
```

```
import com.amazonaws.services.pinpoint.model.Schedule;
import com.amazonaws.services.pinpoint.model.WriteCampaignRequest;

import java.util.ArrayList;
import java.util.List;

public class PinpointCampaignSample {

    public CampaignResponse createCampaign(AmazonPinpointClient client, String appId,
String segmentId) {
        Schedule schedule = new Schedule()
            .withStartTime("IMMEDIATE");

        Message defaultMessage = new Message()
            .withAction(Action.OPEN_APP)
            .withBody("My message body.")
            .withTitle("My message title.");

        MessageConfiguration messageConfiguration = new MessageConfiguration()
            .withDefaultMessage(defaultMessage);

        WriteCampaignRequest request = new WriteCampaignRequest()
            .withDescription("My description.")
            .withSchedule(schedule)
            .withSegmentId(segmentId)
            .withName("MyCampaign")
            .withMessageConfiguration(messageConfiguration);

        CreateCampaignRequest createCampaignRequest = new CreateCampaignRequest()
            .withApplicationId(appId).withWriteCampaignRequest(request);

        CreateCampaignResult result = client.createCampaign(createCampaignRequest);

        System.out.println("Campaign ID: " + result.getCampaignResponse().getId());

        return result.getCampaignResponse();
    }
}
```

When you run this example, the following is printed to the console window of your IDE:

```
Campaign ID: b1c3de717aea4408a75bb3287a906b46
```

Creating A/B Test Campaigns

An A/B test behaves like a standard campaign, but enables you to define different treatments for the campaign message or schedule.

Creating A/B Test Campaigns With the AWS SDK for Java

The following example demonstrates how to create an A/B test campaign with the AWS SDK for Java.

```
import com.amazonaws.services.pinpoint.AmazonPinpointClient;
import com.amazonaws.services.pinpoint.model.Action;
import com.amazonaws.services.pinpoint.model.CampaignResponse;
import com.amazonaws.services.pinpoint.model.CreateCampaignRequest;
```

```
import com.amazonaws.services.pinpoint.model.CreateCampaignResult;
import com.amazonaws.services.pinpoint.model.Message;
import com.amazonaws.services.pinpoint.model.MessageConfiguration;
import com.amazonaws.services.pinpoint.model.Schedule;
import com.amazonaws.services.pinpoint.model.WriteCampaignRequest;
import com.amazonaws.services.pinpoint.model.WriteTreatmentResource;

import java.util.ArrayList;
import java.util.List;

public class PinpointCampaignSample {

    public CampaignResponse createAbCampaign(AmazonPinpointClient client, String appId,
String segmentId) {
        Schedule schedule = new Schedule()
            .withStartTime("IMMEDIATE");

        // Default treatment.
        Message defaultMessage = new Message()
            .withAction(Action.OPEN_APP)
            .withBody("My message body.")
            .withTitle("My message title.");

        MessageConfiguration messageConfiguration = new MessageConfiguration()
            .withDefaultMessage(defaultMessage);

        // Additional treatments
        WriteTreatmentResource treatmentResource = new WriteTreatmentResource()
            .withMessageConfiguration(messageConfiguration)
            .withSchedule(schedule)
            .withSizePercent(40)
            .withTreatmentDescription("My treatment description.")
            .withTreatmentName("MyTreatment");

        List<WriteTreatmentResource> additionalTreatments = new
ArrayList<WriteTreatmentResource>();
        additionalTreatments.add(treatmentResource);

        WriteCampaignRequest request = new WriteCampaignRequest()
            .withDescription("My description.")
            .withSchedule(schedule)
            .withSegmentId(segmentId)
            .withName("MyCampaign")
            .withMessageConfiguration(messageConfiguration)
            .withAdditionalTreatments(additionalTreatments)
            .withHoldoutPercent(10); // Hold out of A/B test

        CreateCampaignRequest createCampaignRequest = new CreateCampaignRequest()
            .withApplicationId(appId).withWriteCampaignRequest(request);

        CreateCampaignResult result = client.createCampaign(createCampaignRequest);

        System.out.println("Campaign ID: " + result.getCampaignResponse().getId());

        return result.getCampaignResponse();
    }
}
```

When you run this example, the following is printed to the console window of your IDE:

```
Campaign ID: b1c3de717aea4408a75bb3287a906b46
```

Streaming Amazon Pinpoint Events to Amazon Kinesis

The Amazon Kinesis platform offers services that you can use to load and analyze streaming data on AWS. You can configure Amazon Pinpoint to send events to Amazon Kinesis Firehose or Amazon Kinesis Streams. By streaming your events, you enable more flexible options for analysis and storage. For more information, and for instructions on how to set up event streaming in the Amazon Pinpoint console, see [Streaming App and Campaign Events with Amazon Pinpoint](#) in the *Amazon Pinpoint User Guide*.

Setting up Event Streaming

The following examples demonstrate how to configure Amazon Pinpoint to automatically send the event data from an app to an Amazon Kinesis stream or Firehose delivery stream.

Prerequisites

These examples require the following input:

- The app ID of an app that is integrated with Amazon Pinpoint and reporting events. For information about how to integrate, see [Integrating Amazon Pinpoint with a Mobile App](#) (p. 32).
- The ARN of an Amazon Kinesis stream or Firehose delivery stream in your AWS account. For information about creating these resources, see [Amazon Kinesis Streams](#) in the *Amazon Kinesis Streams Developer Guide* or [Creating an Amazon Kinesis Firehose Delivery Stream](#) in the *Amazon Kinesis Firehose Developer Guide*.
- The ARN of an AWS Identity and Access Management (IAM) role that authorizes Amazon Pinpoint to send data to the stream. For information about creating a role, see [IAM Role for Streaming Events to Amazon Kinesis](#) (p. 77).

AWS CLI

The following AWS CLI example uses the `put-event-stream` command. This command configures Amazon Pinpoint to send app and campaign events to an Amazon Kinesis stream:

```
aws pinpoint put-event-stream --application-id application-id --write-event-stream  
DestinationStreamArn=stream-arn,RoleArn=role-arn
```

AWS SDK for Java

The following Java example configures Amazon Pinpoint to send app and campaign events to an Amazon Kinesis stream:

```
public PutEventStreamResult createEventStream(AmazonPinpoint pinClient, String appId,
                                             String streamArn, String roleArn)
{
    WriteEventStream stream = new WriteEventStream()
        .withDestinationStreamArn(streamArn)
        .withRoleArn(roleArn);

    PutEventStreamRequest request = new PutEventStreamRequest()
        .withApplicationId(appId)
        .withWriteEventStream(stream);

    return pinClient.putEventStream(request);
}
```

This example constructs a `WriteEventStream` object that stores the ARNs of the Amazon Kinesis stream and the IAM role. The `writeEventStream` object is passed to a `PutEventStreamRequest` object to configure Amazon Pinpoint to stream events for a specific app. The `PutEventStreamRequest` object is passed to the `putEventStream` method of the Amazon Pinpoint client.

You can assign an Amazon Kinesis stream to multiple apps. Amazon Pinpoint will send event data from each app to the stream, enabling you to analyze the data as a collection. The following example method accepts a list of app IDs, and it uses the previous example method, `createEventStream`, to assign a stream to each app:

```
public List<PutEventStreamResult> createEventStreamFromAppList(
    AmazonPinpoint pinClient, List<String> appIDs, String streamArn, String roleArn) {
    return appIDs.stream()
        .map(appId -> createEventStream(pinClient, appId, streamArn, roleArn))
        .collect(Collectors.toList());
}
```

With Amazon Pinpoint, you can assign one stream to multiple apps, but you cannot assign multiple streams to one app.

Disabling Event Streaming

If you assigned an Amazon Kinesis stream to an app, you can disable event streaming for that app. Amazon Pinpoint stops streaming the events, but you can view analytics based on the events in the Amazon Pinpoint console.

AWS CLI

Use the `delete-event-stream` command:

```
aws pinpoint delete-event-stream --application-id application-id
```

AWS SDK for Java

Use the `deleteEventStream` method of the Amazon Pinpoint client:

```
pinClient.deleteEventStream(new DeleteEventStreamRequest().withApplicationId(appId));
```

Event Data

After you set up event streaming, Amazon Pinpoint sends each event reported by your application, and each campaign event created by Amazon Pinpoint, as a JSON data object to your Amazon Kinesis stream.

The event type is indicated by the `event_type` attribute in the event JSON object.

Mobile push events

If the mobile push channel is enabled, and if you have integrated a mobile app, Amazon Pinpoint streams events about app usage and push notification deliveries.

Example

The JSON object for a mobile push event contains the data shown in the following example.

```
{
  "event_type": "_session.stop",
  "event_timestamp": 1487973802507,
  "arrival_timestamp": 1487973803515,
  "event_version": "3.0",
  "application": {
    "app_id": "alb2c3d4e5f6g7h8i9j0k1l2m3n4o5p6",
    "cognito_identity_pool_id": "us-east-1:alb2c3d4-e5f6-g7h8-i9j0-k1l2m3n4o5p6",
    "package_name": "main.page",
    "sdk": {
      "name": "aws-sdk-mobile-analytics-js",
      "version": "0.9.1:2.4.8"
    },
    "title": "title",
    "version_name": "1.0",
    "version_code": "1"
  },
  "client": {
    "client_id": "m3n4o5p6-a1b2-c3d4-e5f6-g7h8i9j0k1l2",
    "cognito_id": "us-east-1:i9j0k1l2-m3n4-o5p6-a1b2-c3d4e5f6g7h8"
  },
  "device": {
    "locale": {
      "code": "en_US",
      "country": "US",
      "language": "en"
    },
    "make": "generic web browser",
    "model": "Unknown",
    "platform": {
      "name": "android",
      "version": "10.10"
    }
  },
  "session": {
    "session_id": "f549dea9-1090-945d-c3d1-e496780baac5",
    "start_timestamp": 1487973202531,
    "stop_timestamp": 1487973802507
  },
  "attributes": {},
}
```



```
"metrics": {}  
}
```

Standard Mobile Push Event Types

Amazon Pinpoint streams the following standard types for the mobile push channel:

- `_session.start`
- `_session.stop`
- `_session.pause`
- `_session.resume`
- `_monetization.purchase`
- `_campaign.send`

Email Events

If the email channel is enabled, Amazon Pinpoint streams events about email deliveries, complaints, opens, and more.

Example

The JSON object for an email event contains the data shown in the following example.

```
{  
  "event_type": "_email.delivered",  
  "event_timestamp": 1487973802507,  
  "arrival_timestamp": 1487973803515,  
  "event_version": "3.0",  
  "application": {  
    "app_id": "a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6",  
    "cognito_identity_pool_id": "us-east-1:a1b2c3d4-e5f6-g7h8-i9j0-k1l2m3n4o5p6",  
    "package_name": "main.page",  
    "sdk": {  
      "name": "aws-sdk-mobile-analytics-js",  
      "version": "0.9.1:2.4.8"  
    },  
    "title": "title",  
    "version_name": "1.0",  
    "version_code": "1"  
  },  
  "client": {  
    "client_id": "m3n4o5p6-a1b2-c3d4-e5f6-g7h8i9j0k1l2",  
    "cognito_id": "us-east-1:i9j0k1l2-m3n4-o5p6-a1b2-c3d4e5f6g7h8"  
  },  
  "device": {  
    "locale": {  
      "code": "en_US",  
      "country": "US",  
      "language": "en"  
    },  
    "make": "generic web browser",  
    "model": "Unknown",  
    "platform": {  
      "name": "android",  
      "version": "10.10"  
    }  
  }  
}
```

```
"session": {
  "session_id": "f549dea9-1090-945d-c3d1-e496780baac5",
  "start_timestamp": 1487973202531,
  "stop_timestamp": 1487973802507
},
"attributes": {},
"metrics": {}
}
```

Standard Email Event Types

Amazon Pinpoint streams the following standard types for the email channel:

- `_email.send`
- `_email.delivered`
- `_email.rejected`
- `_email.hardbounce`
- `_email.softbounce`
- `_email.complaint`
- `_email.open`
- `_email.click`
- `_email.unsubscribe`

SMS Events

If the SMS channel is enabled, Amazon Pinpoint streams events about SMS deliveries.

Example

The JSON object for an SMS event contains the data shown in the following example.

```
{
  "account_id": "123412341234",
  "event_type": "_SMS.SUCCESS",
  "arrival_timestamp": 2345678,
  "timestamp": 13425345,
  "timestamp_created": "1495756908285",
  "application_key": "AppKey-688037015201",
  "unique_id": "uniqueId-68803701520114087975969",
  "attributes": {
    "message_id": "12234sdv",
    "sender_request_id": "abdfg",
    "number_of_message_parts": 1,
    "record_status": "SUCCESS",
    "message_type": "Transactional",
    "keyword": "test",
    "mcc_mnc": "456123",
    "iso_country_code": "US"
  },
  "metrics": {
    "price_in_millicents_usd": 0.0
  },
  "facets": {},
  "additional_properties": {}
}
```

Standard SMS Event Types

Amazon Pinpoint streams the following standard types for the SMS channel:

- `_sms.send`
- `_sms.success`
- `_sms.fail`
- `_sms.optout`

Event Attributes

The JSON object for an event contains the following attributes.

Event

Attribute	Description
<code>event_type</code>	The type of event reported by your app.
<code>event_timestamp</code>	The time at which the event is reported as Unix time in milliseconds. If your app reports events using the AWS Mobile SDK for Android or the AWS Mobile SDK for iOS, the time stamp is automatically generated.
<code>arrival_timestamp</code>	The time at which the event is received by Amazon Pinpoint as Unix time in milliseconds. Does not apply to campaign events.
<code>event_version</code>	The version of the event JSON schema. Check this version in your event-processing application so that you know when to update the application in response to a schema update.
<code>application</code>	Your Amazon Pinpoint app. See the <i>Application</i> table.
<code>client</code>	The app client installed on the device that reports the event. See the <i>Client</i> table.
<code>device</code>	The device that reports the event. See the <i>Device</i> table.
<code>session</code>	The app session on the device. Typically a session begins when a user opens your app.
<code>attributes</code>	Custom attributes that your app reports to Amazon Pinpoint as part of the event.
<code>metrics</code>	Custom metrics that your app reports to Amazon Pinpoint as part of the event.

Application

Attribute	Description
<code>app_id</code>	The unique ID of the Amazon Pinpoint app that reports the event.

Attribute	Description
<code>cognito_identity_pool_id</code>	The unique ID of the Amazon Cognito identity pool used by your app.
<code>package_name</code>	The name of your app package. For example, <code>com.example.my_app</code> .
<code>sdk</code>	The SDK used to report the event.
<code>title</code>	The title of your app.
<code>version_name</code>	The customer-facing app version, such as <code>v2.0</code> .
<code>version_code</code>	The internal code that represents your app version.

Client

Attribute	Description
<code>client_id</code>	The unique ID for the app client installed on the device. This ID is automatically generated by the AWS Mobile SDK for iOS and the AWS Mobile SDK for Android.
<code>cognito_id</code>	The unique ID assigned to the app client in the Amazon Cognito identity pool used by your app.

Device

Attribute	Description
<code>locale</code>	The device locale.
<code>make</code>	The device make, such as <code>Apple</code> or <code>Samsung</code> .
<code>model</code>	The device model, such as <code>iPhone</code> .
<code>platform</code>	The device platform, such as <code>ios</code> or <code>android</code> .

Session

Attribute	Description
<code>session_id</code>	The unique ID for the app session.
<code>start_timestamp</code>	The time at which the session starts as Unix time in milliseconds.
<code>stop_timestamp</code>	The time at which the session stops as Unix time in milliseconds.

Permissions

To use Amazon Pinpoint, users in your AWS account require permissions that allow them to view usage data reported by your app, define user segments, create push notification campaigns, and more. Your app users require access to Amazon Pinpoint so that your app can register endpoints and report usage data. To grant access to Amazon Pinpoint features, create AWS Identity and Access Management (IAM) [policies that allow Amazon Pinpoint actions \(p. 65\)](#).

IAM is a service that helps you securely control access to AWS resources. IAM policies include statements that allow or deny specific actions that users can perform on specific resources. Amazon Pinpoint provides [a set of actions for IAM policies \(p. 67\)](#) that you can use to specify granular permissions for Amazon Pinpoint users. You can grant the appropriate level of access to Amazon Pinpoint without creating overly permissive policies that might expose important data or compromise your campaigns. For example, you can grant unrestricted access to an Amazon Pinpoint administrator, and grant read-only access to individuals in your organization who only need access to analytics.

For more information about IAM policies, see [Overview of IAM Policies](#) in the *IAM User Guide*.

When you add your app to Amazon Pinpoint by creating a project in AWS Mobile Hub, Mobile Hub automatically provisions [AWS resources for app user authentication \(p. 72\)](#). Mobile Hub creates an Amazon Cognito identity pool so that app users can authenticate with AWS. Mobile Hub also creates an IAM role that allows app users to register with Amazon Pinpoint and report usage data. You can customize these resources as needed for your app.

To import endpoint definitions, you must grant Amazon Pinpoint [read-only access to an Amazon S3 bucket \(p. 75\)](#).

Topics

- [IAM Policies for Amazon Pinpoint Users \(p. 65\)](#)
- [User Authentication in Amazon Pinpoint Apps \(p. 72\)](#)
- [AWS Mobile Hub Service Role \(p. 74\)](#)
- [IAM Role for Importing Segments \(p. 75\)](#)
- [IAM Role for Streaming Events to Amazon Kinesis \(p. 77\)](#)
- [IAM Role for Exporting Events to Amazon S3 \(p. 79\)](#)

IAM Policies for Amazon Pinpoint Users

You can add Amazon Pinpoint API actions to AWS Identity and Access Management (IAM) policies to allow or deny specific actions for Amazon Pinpoint users in your account. The Amazon Pinpoint API

actions in your policies control what users can do in the Amazon Pinpoint console. These actions also control which programmatic requests users can make with the AWS SDKs, the AWS CLI, or the Amazon Pinpoint REST API.

In a policy, you specify each action with the `mobiletargeting` namespace followed by a colon and the name of the action, such as `GetSegments`. Most actions correspond to a request to the Amazon Pinpoint REST API using a specific URI and HTTP method. For example, if you allow the `mobiletargeting:GetSegments` action in a user's policy, the user is allowed to make an HTTP GET request against the `/apps/{application-id}/segments` URI. This policy also allows the user to view the segments for an app in the console, and to retrieve the segments by using an AWS SDK or the AWS CLI.

Each action is performed on a specific Amazon Pinpoint resource, which you identify in a policy statement by its Amazon Resource Name (ARN). For example, the `mobiletargeting:GetSegments` action is performed on a specific app, which you identify with the ARN, `arn:aws:mobiletargeting:region:account-id:/apps/application-id`.

You can refer generically to all Amazon Pinpoint actions or resources by using wildcards ("`*`"). For example, to allow all actions for all resources, include the following in a policy statement:

```
"Effect": "Allow",  
"Action": "mobiletargeting:*",  
"Resource": "*"
```

Example Policies

The following examples demonstrate how you can manage Amazon Pinpoint access with IAM policies.

Amazon Pinpoint Administrator

The following administrator policy allows full access to Amazon Pinpoint actions and resources:

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "mobiletargeting:*",  
        "mobileanalytics:*"  
      ],  
      "Resource": "*"   
    }  
  ]  
}
```

In addition to the Amazon Pinpoint actions, this policy allows all Amazon Mobile Analytics actions with `mobileanalytics:*`. Amazon Pinpoint and Amazon Mobile Analytics share data about your apps, so you must include permissions for both services in policies for Amazon Pinpoint users.

Read-Only Access

The following policy allows read-only access for all apps in an account:

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  

```

```
"Action": [
  "mobiletargeting:GetEndpoint",
  "mobiletargeting:GetSegment*",
  "mobiletargeting:GetCampaign*",
  "mobiletargeting:GetImport*",
  "mobiletargeting:GetApnsChannel",
  "mobiletargeting:GetGcmChannel",
  "mobiletargeting:GetApplicationSettings",
  "mobiletargeting:GetEventStream"
],
"Effect": "Allow",
"Resource": "arn:aws:mobiletargeting:*:account-id:apps/*"
},
{
  "Action": "mobiletargeting:GetReports",
  "Effect": "Allow",
  "Resource": "arn:aws:mobiletargeting:*:account-id:reports"
},
{
  "Action": "mobileanalytics:ListApps",
  "Effect": "Allow",
  "Resource": "*"
}
]
```

API Actions for IAM Policies

You can add the following API actions to IAM policies to manage what Amazon Pinpoint users in your account are allowed to do.

mobiletargeting:GetEndpoint

Retrieve information about a specific endpoint.

- URI – `/apps/{application-id}/endpoints/{endpoint-id}`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/endpoints/endpoint-id`

mobiletargeting:UpdateEndpoint

Create an endpoint or update the information for an endpoint.

- URI – `/apps/{application-id}/endpoints/{endpoint-id}`
- Method – PUT
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/endpoints/endpoint-id`

mobiletargeting:UpdateEndpointsBatch

Create or update endpoints as a batch operation.

- URI – `/apps/{application-id}/endpoints`
- Method – PUT
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id`

mobiletargeting>CreateSegment

Create a segment that is based on endpoint data reported to Amazon Pinpoint by your app. To allow a user to create a segment by importing endpoint data from outside of Amazon Pinpoint, allow the `mobiletargeting>CreateImportJob` action.

- URI – `/apps/{application-id}/segments`
- Method – POST
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id`

mobiletargeting:DeleteSegment

Delete a specific segment.

- URI – `/apps/{application-id}/segments/{segment-id}`
- Method – DELETE
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/segments/segment-id`

mobiletargeting:GetSegment

Retrieve information about a specific segment.

- URI – `/apps/{application-id}/segments/{segment-id}`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/segments/segment-id`

mobiletargeting:GetSegments

Retrieve information about the segments for an app.

- URI – `/apps/{application-id}/segments`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id`

mobiletargeting:GetSegmentImportJobs

Retrieve information about jobs that create segments by importing endpoint definitions from Amazon S3.

- URI – `/apps/{application-id}/segments/{segment-id}/jobs/import`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/segments/segment-id`

mobiletargeting:GetSegmentVersion

Retrieve information about a specific segment version.

- URI – `/apps/{application-id}/segments/{segment-id}/versions/{version-id}`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/segments/segment-id`

mobiletargeting:GetSegmentVersions

Retrieve information about the current and prior versions of a segment.

- URI – `/apps/{application-id}/segments/{segment-id}/versions`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/segments/segment-id`

mobiletargeting:UpdateSegment

Update a specific segment.

- URI – `/apps/{application-id}/segments/{segment-id}`

- Method – PUT
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/segments/segment-id`

mobiletargeting:CreateCampaign

Create a campaign for an app.

- URI – `/apps/{application-id}/campaigns`
- Method – POST
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id`

mobiletargeting>DeleteCampaign

Delete a specific campaign.

- URI – `/apps/{application-id}/campaigns/{campaign-id}`
- Method – DELETE
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:/apps/application-id/campaigns/campaign-id`

mobiletargeting:GetCampaign

Retrieve information about a specific campaign.

- URI – `/apps/{application-id}/campaigns/{campaign-id}`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/campaigns/campaign-id`

mobiletargeting:GetCampaignActivities

Retrieve information about the activities performed by a campaign.

- URI – `/apps/{application-id}/campaigns/{campaign-id}/activities`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/campaigns/campaign-id`

mobiletargeting:GetCampaigns

Retrieve information about all campaigns for an app.

- URI – `/apps/{application-id}/campaigns`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id`

mobiletargeting:GetCampaignVersion

Retrieve information about a specific campaign version.

- URI – `/apps/{application-id}/campaigns/{campaign-id}/versions/{version-id}`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/campaigns/campaign-id`

mobiletargeting:GetCampaignVersions

Retrieve information about the current and prior versions of a campaign.

- URI – `/apps/{application-id}/campaigns/{campaign-id}/versions`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/campaigns/campaign-id`

mobiletargeting:UpdateCampaign

Update a specific campaign.

- URI – `/apps/{application-id}/campaigns/{campaign-id}`
- Method – PUT
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/campaigns/campaign-id`

mobiletargeting:CreateImportJob

Import endpoint definitions from Amazon S3 to create a segment.

- URI – `/apps/{application-id}/jobs/import`
- Method – POST
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id`

mobiletargeting:GetImportJob

Retrieve information about a specific import job.

- URI – `/apps/{application-id}/jobs/import/{job-id}`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/jobs/import/job-id`

mobiletargeting:GetImportJobs

Retrieve information about all import jobs for an app.

- URI – `/apps/{application-id}/jobs/import`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id`

mobiletargeting>DeleteApnsChannel

Delete the APNs channel for an app.

- URI – `/apps/{application-id}/channels/apns`
- Method – DELETE
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/channels/apns`

mobiletargeting:GetApnsChannel

Retrieve information about the APNs channel for an app.

- URI – `/apps/{application-id}/channels/apns`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/channels/apns`

mobiletargeting:UpdateApnsChannel

Update the Apple Push Notification service (APNs) certificate and private key, which allow Amazon Pinpoint to send push notifications to your iOS app.

- URI – `/apps/{application-id}/channels/apns`
- Method – PUT
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/channels/apns`

mobiletargeting>DeleteGcmChannel

Delete the GCM channel for an app.

- URI – `/apps/{application-id}/channels/gcm`
- Method – DELETE
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/channels/gcm`

mobiletargeting:GetGcmChannel

Retrieve information about the GCM channel for an app.

- URI – `/apps/{application-id}/channels/gcm`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/channels/gcm`

mobiletargeting:UpdateGcmChannel

Update the Firebase Cloud Messaging (FCM) or Google Cloud Messaging (GCM) API key, which allows Amazon Pinpoint to send push notifications to your Android app.

- URI – `/apps/{application-id}/channels/gcm`
- Method – PUT
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/channels/gcm`

mobiletargeting:GetApplicationSettings

Retrieve the default settings for an app.

- URI – `/apps/{application-id}/settings`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id`

mobiletargeting:UpdateApplicationSettings

Update the default settings for an app.

- URI – `/apps/{application-id}/settings`
- Method – PUT
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id`

mobiletargeting>DeleteEventStream

Delete the event stream for an app.

- URI – `/apps/{application-id}/eventstream/`
- Method – DELETE
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/eventstream`

mobiletargeting:GetEventStream

Retrieve information about the event stream for an app.

- URI – `/apps/{application-id}/eventstream/`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/eventstream`

mobiletargeting:PutEventStream

Create or update an event stream for an app.

- URI – `/apps/{application-id}/eventstream/`
- Method – PUT

- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/application-id/eventstream`

mobiletargeting:GetReports

View analytics in the Amazon Pinpoint console.

- URI – Not applicable
- Method – Not applicable
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:reports`

User Authentication in Amazon Pinpoint Apps

To integrate with Amazon Pinpoint, your app must authenticate users to register endpoints and report usage data. When you add your app to Amazon Pinpoint by creating a project in AWS Mobile Hub, Mobile Hub automatically provisions the following AWS resources to help you implement user authentication:

Amazon Cognito identity pool

Amazon Cognito creates unique identities for your users and provides credentials that grant temporary access to the backend AWS resources for your app. An identity pool is a store of user identity data for your app users.

Amazon Cognito provides credentials for authenticated and unauthenticated users. Authenticated users include those who sign in to your app through a public identity provider, such as Facebook, Amazon, or Google. Unauthenticated users are those who do not sign in to your app, such as guest users.

You control your users' access to AWS resources with separate AWS Identity and Access Management (IAM) roles for authenticated and unauthenticated users. These roles must be assigned to the identity pool.

IAM role for unauthenticated users

Includes permissions policies that delegate limited access to AWS resources for unauthenticated users. You can customize the role as needed. By default, this role is assigned to the Amazon Cognito identity pool.

If your app requires users to authenticate with a public identity provider, you must create an IAM role for authenticated users and assign this role to the identity pool. To support Amazon Pinpoint, the permissions in your authenticated role must include the same permissions as those in the unauthenticated role created by Mobile Hub.

For more information about IAM roles, see [IAM Roles](#) in the *IAM User Guide*.

Your app code uses Amazon Cognito and IAM to authenticate users as follows:

1. Your app code constructs an Amazon Cognito credentials provider.
2. Your app code passes the provider as a parameter when it initializes the Amazon Pinpoint client.
3. The Amazon Pinpoint client uses the provider to get credentials for the user's identity in the identity pool. New users are assigned a new identity.
4. The user gains the permissions granted by the IAM roles that are associated with the identity pool.

For code examples that show how to construct the credentials provider and initialize the Amazon Pinpoint client, see [Initializing the Amazon Pinpoint Client \(iOS\)](#) (p. 34) and [Initializing the Amazon Pinpoint Client \(Android\)](#) (p. 41).

For more information about how Amazon Cognito supports user authentication, see [Amazon Cognito Identity: Using Federated Identities](#) in the *Amazon Cognito Developer Guide*.

Unauthenticated Role

The unauthenticated role created by Mobile Hub allows your app users to send data to Amazon Pinpoint. The role name includes "unauth_MOBILEHUB"; for example, in the IAM console, you will see a role with a name similar to `MySampleApp_unauth_MOBILEHUB_1234567890`.

IAM roles delegate permissions with two types of policies:

- Permissions policy – Grants the user of the role permission to take the specified actions on the specified resources.
- Trust policy – Specifies which entities are allowed to assume the role and gain its permissions.

Permissions Policies

The unauthenticated role includes two permissions policies. The following permissions policy allows your app users to register with Amazon Pinpoint and report app usage events. Mobile Hub assigns the policy a name that includes "mobileanalytics_MOBILEHUB".

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "mobileanalytics:PutEvents"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "mobiletargeting:UpdateEndpoint"
      ],
      "Resource": [
        "arn:aws:mobiletargeting:*:*:apps/*"
      ]
    }
  ]
}
```

After your app is integrated with Amazon Pinpoint, your app registers an endpoint with Amazon Pinpoint when a new user starts an app session. Your app sends updated endpoint data to Amazon Pinpoint each time the user starts a new session.

The following permissions policy allows your app users to establish an identity with the Amazon Cognito identity pool for your app. Mobile Hub assigns the policy a name that includes "signin_MOBILEHUB".

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
        "cognito-identity:GetId"
      ],
      "Resource": [
        "arn:aws:cognito-identity:*:*:identityPool/us-
east-1:1a2b3c4d-5e6f-7g8h-9i0j-1k2l3m4n5o6p"
      ]
    }
  ]
}
```

Trust Policy

To allow Amazon Cognito to assume the role for unauthenticated users in your identity pool, Mobile Hub adds the following trust policy to the role:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.amazonaws.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com:aud": "us-
east-1:1a2b3c4d-5e6f-7g8h-9i0j-1k2l3m4n5o6p"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "unauthenticated"
        }
      }
    }
  ]
}
```

For an example of a trust policy assigned to an authenticated role, see [Role-Based Access Control](#) in the *Amazon Cognito Developer Guide*.

AWS Mobile Hub Service Role

AWS Mobile Hub creates an AWS Identity and Access Management (IAM) role in your AWS account when you agree to a one-time request in the Mobile Hub console to manage AWS resources and services for you. This role, called `MobileHub_Service_Role`, allows Mobile Hub to create and modify your AWS resources and services for your Mobile Hub project.

For more information about the Mobile Hub service role, see [Mobile Hub Service Role and Policies Used on Your Behalf](#) in the *AWS Mobile Hub Developer Guide*.

To add an app to Amazon Pinpoint, you create a Mobile Hub project and configure it to include the User Engagement feature. To support this feature, Mobile Hub adds the following permissions to the Mobile Hub service role:

```
{
  "Effect": "Allow",
  "Action": [
```

```
"mobiletargeting:UpdateApnsChannel",
"mobiletargeting:UpdateApnsSandboxChannel",
"mobiletargeting:UpdateGcmChannel",
"mobiletargeting>DeleteApnsChannel",
"mobiletargeting>DeleteApnsSandboxChannel",
"mobiletargeting>DeleteGcmChannel"
],
"Resource": [
  "arn:aws:mobiletargeting:*:*:apps/*/channels/*"
]
}
```

These permissions allow Mobile Hub to manage the channels that Amazon Pinpoint uses to deliver messages to the push notification services for iOS and Android. Mobile Hub creates or updates a channel when you provide your credentials for Apple Push Notification service, Firebase Cloud Messaging, or Google Cloud Messaging. You provide your credentials by using the Mobile Hub console or Amazon Pinpoint console.

IAM Role for Importing Segments

With Amazon Pinpoint, you define a user segment by importing endpoint definitions from an Amazon S3 bucket in your AWS account. Before you import, you must delegate the required permissions to Amazon Pinpoint. Create an AWS Identity and Access Management (IAM) role and attach the following policies to the role:

- The `AmazonS3ReadOnlyAccess` AWS managed policy. This policy is created and managed by AWS, and it grants read-only access to your Amazon S3 bucket.
- A *trust policy* that allows Amazon Pinpoint to assume the role.

For more information about IAM roles, see [IAM Roles](#) in the *IAM User Guide*.

After you create the role, you can use Amazon Pinpoint to import segments. For an example of how to import a segment by using the AWS SDK for Java, see [Importing Segments \(p. 52\)](#). For information about creating the Amazon S3 bucket, creating endpoint files, and importing a segment by using the console, see [Importing Segments](#) in the *Amazon Pinpoint User Guide*.

Trust Policy

To allow Amazon Pinpoint to assume the IAM role and perform the actions allowed by the `AmazonS3ReadOnlyAccess` policy, attach the following trust policy to the role:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "pinpoint.us-east-1.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "account-id"
        }
      }
    }
  ]
}
```

```
}
```

For `sts:ExternalId`, replace `account-id` with your 12-digit AWS account ID. The external ID restricts who can assume the role, and you provide the ID to Amazon Pinpoint when you import the endpoint definitions for your segment.

Creating the IAM Role (AWS CLI)

Complete the following steps to create the IAM role by using the AWS Command Line Interface (AWS CLI).

If you have not installed the AWS CLI, see [Getting Set Up with the AWS Command Line Interface](#) in the *AWS Command Line Interface User Guide*.

To create the role by using the IAM console, see [Creating an IAM Role for Importing](#) in the *Amazon Pinpoint User Guide*.

To create the IAM role by using the AWS CLI

1. Create a JSON file that contains the trust policy for your role, and save the file locally. You can copy the trust policy provided in this topic.
2. Use the `create-role` command to create the role and attach the trust policy:

```
aws iam create-role --role-name PinpointSegmentImport --assume-role-policy-document  
file://PinpointImportTrustPolicy.json
```

Following the `file://` prefix, specify the path to the JSON file that contains the trust policy.

When you run this command, the AWS CLI prints the following output in your terminal:

```
{  
  "Role": {  
    "AssumeRolePolicyDocument": {  
      "Version": "2012-10-17",  
      "Statement": [  
        {  
          "Action": "sts:AssumeRole",  
          "Effect": "Allow",  
          "Condition": {  
            "StringEquals": {  
              "sts:ExternalId": "myExternalId"  
            }  
          },  
          "Principal": {  
            "Service": "pinpoint.us-east-1.amazonaws.com"  
          }  
        }  
      ]  
    },  
    "RoleId": "AIDACKCEVSQ6C2EXAMPLE",  
    "CreateDate": "2016-12-20T00:44:37.406Z",  
    "RoleName": "PinpointSegmentImport",  
    "Path": "/",  
    "Arn": "arn:aws:iam::111122223333:role/PinpointSegmentImport"  
  }  
}
```

3. Use the `attach-role-policy` command to attach the `AmazonS3ReadOnlyAccess` AWS managed policy to the role:


```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess  
--role-name PinpointSegmentImport
```

IAM Role for Streaming Events to Amazon Kinesis

Amazon Pinpoint can automatically send app usage data, or *event data*, from your app to an Amazon Kinesis stream or Amazon Kinesis Firehose delivery stream in your AWS account. Before Amazon Pinpoint can begin streaming the event data, you must delegate the required permissions to Amazon Pinpoint.

If you use the console to set up event streaming, Amazon Pinpoint automatically creates an AWS Identity and Access Management (IAM) role with the required permissions. For more information, see [Streaming Amazon Pinpoint Events to Amazon Kinesis](#) in the *Amazon Pinpoint User Guide*.

If you want to create the role manually, attach the following policies to the role:

- A permissions policy that allows Amazon Pinpoint to send records to your stream.
- A trust policy that allows Amazon Pinpoint to assume the role.

For more information about IAM roles, see [IAM Roles](#) in the *IAM User Guide*.

After you create the role, you can configure Amazon Pinpoint to automatically send events to your stream. For more information, see [Streaming Amazon Pinpoint Events to Amazon Kinesis \(p. 58\)](#).

Permissions Policies

To allow Amazon Pinpoint to send event data to your stream, attach one of the following policies to the role.

Amazon Kinesis Streams

The following policy allows Amazon Pinpoint to send event data to an Amazon Kinesis stream.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Action": [  
        "kinesis:PutRecords",  
        "kinesis:DescribeStream"  
      ],  
      "Effect": "Allow",  
      "Resource": [  
        "arn:aws:kinesis:region:account-id:stream/stream-name"  
      ]  
    }  
  ]  
}
```

Amazon Kinesis Firehose

The following policy allows Amazon Pinpoint to send event data to a Firehose delivery stream.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Action": [  
        "kinesis:PutRecords",  
        "kinesis:DescribeStream"  
      ],  
      "Effect": "Allow",  
      "Resource": [  
        "arn:aws:kinesis:region:account-id:stream/stream-name"  
      ]  
    }  
  ]  
}
```

```
"Effect": "Allow",
"Action": [
  "firehose:PutRecordBatch",
  "firehose:DescribeDeliveryStream"
],
"Resource": [
  "arn:aws:firehose:region:account-id:deliverystream/delivery-stream-name"
]
}
```

Trust Policy

To allow Amazon Pinpoint to assume the IAM role and perform the actions allowed by the permissions policy, attach the following trust policy to the role:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "pinpoint.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Creating the IAM Role (AWS CLI)

Complete the following steps to create the IAM role by using the AWS Command Line Interface (AWS CLI).

If you have not installed the AWS CLI, see [Getting Set Up with the AWS Command Line Interface](#) in the *AWS Command Line Interface User Guide*.

To create the role by using the IAM console, see [Setting up Event Streaming](#) in the *Amazon Pinpoint User Guide*.

To create the IAM role by using the AWS CLI

1. Create a JSON file that contains the trust policy for your role, and save the file locally. You can copy the trust policy provided in this topic.
2. Use the `create-role` command to create the role and attach the trust policy:

```
aws iam create-role --role-name PinpointEventStreamRole --assume-role-policy-document
file://PinpointEventStreamTrustPolicy.json
```

Following the `file://` prefix, specify the path to the JSON file that contains the trust policy.

When you run this command, the AWS CLI prints the following output in your terminal:

```
{
  "Role": {
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
```

```
{
  "Action": "sts:AssumeRole",
  "Effect": "Allow",
  "Principal": {
    "Service": "pinpoint.amazonaws.com"
  }
}
],
"RoleId": "AIDACKCEVSQ6C2EXAMPLE",
"CreateDate": "2017-02-28T18:02:48.220Z",
"RoleName": "PinpointEventStreamRole",
"Path": "/",
"Arn": "arn:aws:iam::111122223333:role/PinpointEventStreamRole"
}
```

3. Create a JSON file that contains the permissions policy for your role, and save the file locally. You can copy one of the policies provided in the [Permissions Policies \(p. 77\)](#) section.
4. Use the `put-role-policy` command to attach the permissions policy to the role:

```
aws iam put-role-policy --role-name PinpointEventStreamRole --
policy-name PinpointEventStreamPermissionsPolicy --policy-document
file://PinpointEventStreamPermissionsPolicy.json
```

Following the `file://` prefix, specify the path to the JSON file that contains the permissions policy.

IAM Role for Exporting Events to Amazon S3

Amazon Pinpoint can automatically export analytics data, or *event data*, from your app to an Amazon S3 bucket in your AWS account. Before you export the event data, you must delegate the required permissions to Amazon Pinpoint. Create an AWS Identity and Access Management (IAM) role and attach the following policies to the role:

- A permissions policy that allows Amazon Pinpoint to add event data to your Amazon S3 bucket.
- A trust policy that allows Amazon Pinpoint to assume the role.

For more information about IAM roles, see [IAM Roles](#) in the *IAM User Guide*.

After you create the role, you can export event data by setting up event streaming in the Amazon Pinpoint console. For more information, see [Exporting Amazon Pinpoint Events to Amazon S3](#) in the *Amazon Pinpoint User Guide*.

Permissions Policy

To allow Amazon Pinpoint to add event data to your Amazon S3 bucket, attach the following permissions policy to the role:

```
{
  "Statement": [
    {
      "Resource": "arn:aws:s3:::your-bucket-name/*",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl",
        "s3:PutObjectVersionAcl"
      ]
    }
  ]
}
```

```
    "Effect": "Allow"
  }
],
"Version": "2012-10-17"
}
```

Replace `your-bucket-name` with the name of the Amazon S3 bucket to which you want to export the event data.

Trust Policy

To allow Amazon Pinpoint to assume the IAM role and perform the actions allowed by the permissions policy, attach the following trust policy to the role:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "mobileanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Creating the IAM Role (AWS CLI)

Complete the following steps to create the IAM role by using the AWS Command Line Interface (AWS CLI).

If you have not installed the AWS CLI, see [Getting Set Up with the AWS Command Line Interface](#) in the *AWS Command Line Interface User Guide*.

To create the role by using the IAM console, see [Setting up Automatic Exports to Amazon S3](#) in the *Amazon Pinpoint User Guide*.

To create the IAM role by using the AWS CLI

1. Create a JSON file that contains the trust policy for your role, and save the file locally. You can copy the trust policy provided in this topic.
2. Use the `create-role` command to create the role and attach the trust policy:

```
aws iam create-role --role-name PinpointAnalyticsExport --assume-role-policy-document
file://PinpointExportTrustPolicy.json
```

Following the `file://` prefix, specify the path to the JSON file that contains the trust policy.

When you run this command, the AWS CLI prints the following output in your terminal:

```
{
  "Role": {
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
```

```
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::111122223333:root"
        }
    ]
},
"RoleId": "AIDACKCEVSQ6C2EXAMPLE",
"CreateDate": "2016-12-20T18:02:48.220Z",
"RoleName": "PinpointAutoExport",
"Path": "/",
"Arn": "arn:aws:iam::111122223333:role/PinpointAutoExport"
}
```

3. Create a JSON file that contains the permissions policy for your role, and save the file locally. You can copy the permissions policy provided in this topic.
4. Use the `put-role-policy` command to attach the permissions policy to the role:

```
aws iam put-role-policy --role-name PinpointAutoExport --
policy-name PinpointExportPermissionsPolicy --policy-document
file://PinpointExportPermissionsPolicy.json
```

Following the `file://` prefix, specify the path to the JSON file that contains the permissions policy.

Document History for Amazon Pinpoint

The following table describes the documentation for this release of Amazon Pinpoint.

- **Latest documentation update:** June 08, 2017

Change	Description	Date
Updated sample events	The example events (p. 60) include events that Amazon Pinpoint streams for email and SMS activity.	June 08, 2017
Android session management	Manage sessions (p. 44) in Android apps by using a class provided by the AWS Mobile Hub sample app.	April 20, 2017
Updated monetization event samples	The sample code is updated for reporting monetization events with the Mobile SDK for iOS (p. 37) and Mobile SDK for Android (p. 45) .	March 31, 2017
Event streams	You can configure Amazon Pinpoint to send your app and campaign events to an Amazon Kinesis stream (p. 58) .	March 24, 2017
Permissions	See Permissions (p. 65) for information about granting access to Amazon Pinpoint for AWS users in your account and users of your mobile app.	January 12, 2017
Amazon Pinpoint general availability	This release introduces Amazon Pinpoint.	December 1, 2016