
Amazon Pinpoint

Developer Guide



Amazon Pinpoint: Developer Guide

Copyright © 2019 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
Amazon Pinpoint Features	1
Define Audience Segments	1
Engage Your Audience with Messaging Campaigns	1
Send Transactional Messages	1
Analyze User Behavior	1
Regional Availability	2
Integrating with Your Application	3
SDK Support	3
Integrating the Mobile SDKs or JS Library	4
Integrating Mobile SDKs	4
Integrating AWS Amplify	4
Next Step	5
Registering Endpoints	5
Before You Begin	5
AWS Mobile SDKs	5
AWS Amplify	5
Next Steps	5
Reporting Events	6
Before You Begin	6
AWS Mobile SDKs	7
Web and React Native	5
Amazon Pinpoint Events API	7
Next Step	7
Handling Push Notifications	7
Setting Up Push Notifications	7
Handling Push Notifications	9
Defining Your Audience	10
Adding Endpoints	10
Examples	11
Related Information	15
Associating Users with Endpoints	15
Examples	15
Related Information	18
Adding a Batch of Endpoints	19
Examples	19
Related Information	23
Importing Endpoints	24
Before You Begin	24
Examples	24
Related Information	32
Deleting Endpoints	32
Examples	32
Accessing Audience Data	35
Looking Up Endpoints	35
Examples	35
Related Information	39
Exporting Endpoints	39
Before You Begin	39
Examples	40
Related Information	46
Listing Endpoint IDs	46
Creating Segments	50
Building Segments	50

Building Segments With the AWS SDK for Java	50
Importing Segments	51
Importing a Segment	51
Customizing Segments with AWS Lambda	54
Event Data	55
Creating a Lambda Function	56
Assigning a Lambda Function Policy	57
Assigning a Lambda Function to a Campaign	58
Creating Campaigns	60
Creating Standard Campaigns	60
Creating Campaigns With the AWS SDK for Java	60
Creating A/B Test Campaigns	61
Creating A/B Test Campaigns With the AWS SDK for Java	61
Sending Messages	63
Send Email	63
Choosing an Email-Sending Method	64
Using the API	64
Using the Email API	71
Using the SMTP Interface	80
Send SMS Messages	88
Send Voice Messages	94
Streaming Events	99
Setting up Event Streaming	99
AWS CLI	99
AWS SDK for Java	99
Disabling Event Streaming	100
AWS CLI	100
AWS SDK for Java	100
Event Data	100
App Events	101
Email Events	102
SMS Events	103
Event Attributes	104
Logging API Calls	106
Amazon Pinpoint Information in CloudTrail	106
Amazon Pinpoint API Actions That Can be Logged by CloudTrail	107
Amazon Pinpoint SMS Email Actions That Can be Logged by CloudTrail	108
Amazon Pinpoint SMS and Voice API Actions That Can be Logged by CloudTrail	109
Example: Amazon Pinpoint Log File Entries	110
Deleting Data	113
Deleting Endpoints	113
Deleting Segment and Endpoint Data Stored in Amazon S3	113
Delete All AWS Data by Closing Your AWS Account	113
Permissions	115
IAM Policies for Users	115
Example Policies	116
Amazon Pinpoint API Actions	117
Amazon Pinpoint SMS and Voice API Actions	128
IAM Role for Importing Endpoints or Segments	130
Trust Policy	130
Creating the IAM Role (AWS CLI)	130
IAM Role for Exporting Endpoints or Segments	131
Step 1: Create the IAM Policy	132
Step 2: Create the IAM Role	133
IAM Role for Streaming Events to Kinesis	134
Permissions Policies	135
Trust Policy	136

Creating the IAM Role (AWS CLI)	136
Limits	138
General Limits	138
Endpoint Limits	138
Endpoint Import Limits	139
Segment Limits	139
Campaign Limits	139
Mobile Push Limits	140
Email Limits	141
Email Sending Limits	141
Email Message Limits	141
Email Sender and Recipient Limits	141
SMS Limits	142
Voice Limits	143
Event Ingestion Limits	144
Requesting a Limit Increase	145
Creating Custom Channels with AWS Lambda	146
Event Data	146
Creating a Lambda Function	147
Example Lambda Function	147
Assigning a Lambda Function Policy	57
Example Function Policy	57
Granting Amazon Pinpoint Invocation Permission	150
Assigning a Lambda Function to a Campaign	150
Document History	152

What Is Amazon Pinpoint?

Amazon Pinpoint is an AWS service that you can use to engage with your customers across multiple messaging channels. You can use Amazon Pinpoint to send push notifications, emails, SMS text messages, or voice messages.

The information in this developer guide is intended for application developers. This guide contains information about using the features of Amazon Pinpoint programmatically. It also contains information of particular interest to mobile app developers, such as procedures for [integrating analytics and messaging features with your application \(p. 3\)](#).

There are several other documents that are companions to this document. The following documents provide reference information related to the Amazon Pinpoint APIs:

- [Amazon Pinpoint API Reference](#)
- [Amazon Pinpoint Email API](#)
- [Amazon Pinpoint SMS and Voice API](#)

If you're new to Amazon Pinpoint, you might find it helpful to review the [Amazon Pinpoint User Guide](#) before proceeding with this document.

Amazon Pinpoint Features

This section describes the major features of Amazon Pinpoint and the tasks that you can perform by using them.

Define Audience Segments

Reach the right audience for your messages by [defining audience segments \(p. 50\)](#). A segment designates which users receive the messages that are sent from a campaign. You can define dynamic segments based on data that's reported by your application, such as operating system or mobile device type. You can also import static segments that you define outside Amazon Pinpoint.

Engage Your Audience with Messaging Campaigns

Engage your audience by [creating a messaging campaign \(p. 60\)](#). A campaign sends tailored messages on a schedule that you define. You can create campaigns that send mobile push, email, or SMS messages.

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

Send Transactional Messages

Keep your customers informed by sending transactional mobile push and SMS messages—such as new account activation messages, order confirmations, and password reset notifications—to specific users. You can send transactional messages by using the Amazon Pinpoint REST API.

Analyze User Behavior

Gain insights about your audience and the effectiveness of your campaigns by using the analytics that Amazon Pinpoint provides. You can view trends about your users' level of engagement, purchase activity,

and demographics. You can also monitor your message traffic with metrics for messages that are 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 \(p. 99\)](#) to Amazon Kinesis.

Regional Availability

Amazon Pinpoint is available in several AWS Regions and it provides an endpoint for each of these Regions. For a list of supported Regions and endpoints, see [AWS Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

Integrating Amazon Pinpoint with Your Application

Integrate Amazon Pinpoint with your client code to understand and engage your users.

After you integrate, as users launch your application, it connects to the Amazon Pinpoint service to add or update *endpoints*. Endpoints represent the destinations that you can message—such as user devices, email addresses, or phone numbers.

Also, your application provides usage data, or *events*. View event data in the Amazon Pinpoint console to learn how many users you have, how often they use your application, when they use it, and more.

After your application supplies endpoints and events, you can use this information to tailor messaging campaigns for specific audiences, or *segments*. (You can also directly message simple lists of recipients without creating campaigns.)

Use the topics in this section to integrate Amazon Pinpoint with a mobile or web client. These topics include code examples and procedures to integrate with an Android, iOS, or JavaScript application.

Outside of your client, you can use [supported AWS SDKs \(p. 3\)](#) or the [Amazon Pinpoint API](#) to import endpoints, export event data, define customer segments, create and run campaigns, and more.

To start integrating your apps, see [the section called “Integrating the Mobile SDKs or JS Library” \(p. 4\)](#).

Topics

- [AWS SDK Support for Amazon Pinpoint \(p. 3\)](#)
- [Integrating the AWS Mobile SDKs or JavaScript Library with Your Application \(p. 4\)](#)
- [Registering Endpoints in Your Application \(p. 5\)](#)
- [Reporting Events in Your Application \(p. 6\)](#)
- [Handling Push Notifications \(p. 7\)](#)

AWS SDK Support for Amazon Pinpoint

One of the easiest ways to interact with the Amazon Pinpoint API is to use an AWS SDK. The following AWS SDKs include support for Amazon Pinpoint API operations:

- [AWS Mobile SDK for Android](#) version 2.3.5 or later
- [AWS Mobile SDK for iOS](#) version 2.4.14 or later
- [AWS Amplify JavaScript Library for JavaScript](#)
- [AWS Amplify JavaScript Library for React Native](#)
- [AWS SDK for JavaScript](#) version 2.7.10 or later
- [AWS SDK for Java](#) version 1.11.63 or later
- [AWS SDK for .NET](#) version 3.3.27.0 or later

- [AWS SDK for PHP](#) version 3.20.1
- [AWS SDK for Python \(Boto 3\)](#) version 1.4.2 or later
- [AWS SDK for Ruby](#) version 1.0.0.rc2 or later
- [AWS SDK for Go](#) version 1.5.13 or later
- [AWS SDK for C++](#) version 1.0.20151208.143 or later

You can also interact with the Amazon Pinpoint API by using version 1.11.24 or later of the [AWS Command Line Interface](#) (AWS CLI). The AWS CLI requires Python 2.6.5 or later, or Python 3.3 or later. For more information about installing and configuring the AWS CLI, see [Installing the AWS Command Line Interface](#) in the *AWS Command Line Interface User Guide*.

Note

The version numbers shown in this section are the first versions of each SDK or CLI that included support for the Amazon Pinpoint API. New resources or operations are occasionally added to the Amazon Pinpoint API. In order to use all of the features of Amazon Pinpoint through the API, ensure that you're using the latest version of the SDK or CLI.

Integrating the AWS Mobile SDKs or JavaScript Library with Your Application

To connect to the Amazon Pinpoint service from your web or mobile application, integrate the AWS Mobile SDKs or JavaScript library with your code. With these resources, you can use the native programming language for your platform to issue requests to the Amazon Pinpoint API. For example, you can add endpoints, apply custom endpoint attributes, report usage data, and more.

For Android or iOS apps, use the AWS Mobile SDKs. For web or mobile JavaScript applications, use the AWS Amplify JavaScript library for web and React Native.

Integrating the AWS Mobile SDKs for Android or iOS

Use AWS Amplify to integrate your app with AWS. For iOS apps, see [Getting Started](#) in the iOS SDK documentation. For Android apps, see [Getting Started](#) in the Android SDK documentation. These topics help you:

- Create a project with AWS Amplify.
- Connect your app to the AWS features and resources that AWS Amplify supports.

Integrating the AWS Amplify JavaScript Library

To integrate AWS Amplify with your JavaScript application, see [Get Started \(JavaScript\)](#) or [Get Started \(React Native\)](#) in the AWS Amplify JavaScript documentation. These topics help you:

- Use command line tools and AWS Amplify to create a project.
- Create backend AWS resources for your application.
- Connect your app to the backend resources.
- Integrate the AWS Amplify library with your application.

After you integrate AWS Amplify, return to this topic in the *Amazon Pinpoint Developer Guide* for the next step.

Next Step

You've integrated AWS Amplify with your application. Next, update your code to register your users' devices as endpoints. See [Registering Endpoints in Your Application \(p. 5\)](#).

Registering Endpoints in Your Application

When a user starts a session (for example, by launching your mobile app), your mobile or web application can automatically register (or update) an *endpoint* with Amazon Pinpoint. The endpoint represents the device that the user starts the session with. It includes attributes that describe the device, and it can also include custom attributes that you define. Endpoints can also represent other methods of communicating with customers, such as email addresses or mobile phone numbers.

After your application registers endpoints, you can segment your audience based on endpoint attributes. You can then engage these segments with tailored messaging campaigns. You can also use the **Analytics** page in the Amazon Pinpoint console to view charts about endpoint registration and activity, such as **New endpoints** and **Daily active endpoints**.

You can assign a single user ID to multiple endpoints. A user ID represents a single user, while each endpoint that is assigned the user ID represents one of the user's devices. After you assign user IDs to your endpoints, you can view charts about user activity in the console, such as **Daily active users** and **Monthly active users**.

Before You Begin

If you haven't already, integrate the AWS Mobile SDK for Android or iOS, or integrate the AWS Amplify JavaScript library with your application. See [Integrating the AWS Mobile SDKs or JavaScript Library with Your Application \(p. 4\)](#).

Registering Endpoints with the AWS Mobile SDKs for Android or iOS

You can use the AWS Mobile SDKs for Android or iOS to register and customize endpoints. For more information, and to view code examples, see the following documents:

- [Registering Endpoints in Your Application](#) in the Android SDK documentation.
- [Registering Endpoints in Your Application](#) in the iOS SDK documentation.

Registering Endpoints with the AWS Amplify JavaScript Library

You can use the AWS Amplify JavaScript library to register and update endpoints in your apps. For more information, and to view code examples, see [Update Endpoint](#) in the AWS Amplify JavaScript documentation.

Next Steps

You've updated your app to register endpoints. Now, as users launch your app, device information and custom attributes are provided to Amazon Pinpoint. You can use this information to define audience

segments. In the console, you can see metrics about endpoints and, if applicable, users who are assigned user IDs.

Next, complete the steps at [Reporting Events in Your Application \(p. 6\)](#) to update your app to report usage data.

Reporting Events in Your Application

In your mobile or web application, you can use AWS Mobile SDKs or the [Amazon Pinpoint Events API](#) to report usage data, or *events*, to Amazon Pinpoint. You can report events to capture information such as session times, users' purchasing behavior, sign-in attempts, or any custom event type that you need.

After your application reports events, you can view analytics in the Amazon Pinpoint console. The charts on the **Analytics** page provide metrics for many aspects of user behavior. For more information, see [Chart Reference for Amazon Pinpoint Analytics](#) in the *Amazon Pinpoint User Guide*.

To analyze and store your event data outside of Amazon Pinpoint, you can configure Amazon Pinpoint to stream the data to Amazon Kinesis. For more information, see [Streaming Amazon Pinpoint Events to Kinesis \(p. 99\)](#).

By using the AWS Mobile SDKs and the AWS Amplify JavaScript libraries, you can call the Amazon Pinpoint API to report the following types of events:

Session events

Indicate when and how often users open and close your app.

After your application reports session events, use the **Analytics** page in the Amazon Pinpoint console to view charts for **Sessions**, **Daily active endpoints**, **7-day retention rate**, and more.

Custom events

Are nonstandard events that you define by assigning a custom event type. You can add custom attributes and metrics to a custom event.

On the **Analytics** page in the console, the **Events** tab displays metrics for all custom events that are reported by your app.

Monetization events

Report the revenue that's generated by your application and the number of items that are purchased by users.

On the **Analytics** page, the **Revenue** tab displays charts for **Revenue**, **Paying users**, **Units sold**, and more.

Authentication events

Indicate how frequently users authenticate with your application.

On the **Analytics** page, the **Users** tab displays charts for **Sign-ins**, **Sign-ups**, and **Authentication failures**.

Before You Begin

If you haven't already, do the following:

- Integrate your app with AWS Amplify. See [Integrating the AWS Mobile SDKs or JavaScript Library with Your Application \(p. 4\)](#).

- Update your application to register endpoints. See [Registering Endpoints in Your Application \(p. 5\)](#).

Reporting Events with the AWS Mobile SDKs for Android or iOS

You can enable a mobile app to report events to Amazon Pinpoint by using the AWS Mobile SDKs for iOS and Android.

For more information about updating your app to record and submit events to Amazon Pinpoint, see the following pages in the AWS Amplify documentation:

- [Analytics](#) in the iOS SDK documentation
- [Analytics](#) in the Android SDK documentation

Reporting Events with the AWS Amplify JavaScript Library

You can enable JavaScript and React Native apps to report application usage events to Amazon Pinpoint by using the AWS Amplify JavaScript library. For more information about updating your app to submit events to Amazon Pinpoint, see [Analytics](#) in the AWS Amplify JavaScript documentation.

Reporting Events by Using the Amazon Pinpoint API

You can use the Amazon Pinpoint API or an AWS SDK to submit events to Amazon Pinpoint in bulk. For more information, see [Events](#) in the *Amazon Pinpoint API Reference*.

Next Step

You've updated your app to report events. Now when users interact with your app, it sends usage data to Amazon Pinpoint. You can view this data in the console, and you can stream it to Amazon Kinesis.

Next, update your app to handle push notifications that you send with Amazon Pinpoint. See [Handling Push Notifications \(p. 7\)](#).

Handling Push Notifications

The following topics describe how to modify your iOS or Android app so that it receives push notifications that you send by using Amazon Pinpoint.

Topics

- [Setting Up Push Notifications for Amazon Pinpoint \(p. 7\)](#)
- [Handling Push Notifications \(p. 9\)](#)

Setting Up Push Notifications for Amazon Pinpoint

In order to set up Amazon Pinpoint so that it can send push notifications to your apps, you first have to provide the credentials that authorize Amazon Pinpoint to send messages to your app. The credentials that you provide depend on which push notification system you use:

- 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. These credentials authorize Amazon Pinpoint to send messages to your app through Firebase Cloud Messaging.

After you obtain the credentials for a push notification channel, you have to create a project in Amazon Pinpoint and provide it with the credentials for the push notification service.

Topics

- [Setting Up iOS Push Notifications \(p. 8\)](#)
- [Setting Up Android Push Notifications \(p. 8\)](#)
- [Create a Project in Amazon Pinpoint \(p. 8\)](#)

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. You can find more information about completing these steps in [Setting Up APNs for Push Notifications](#) in the iOS SDK documentation.

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). You can use your FCM credentials to create an Android project and launch a sample app that can receive push notifications. You can find more information about completing these steps in [Setting Up FCM/GCM for Push Notifications](#) in the Android SDK documentation.

Create a Project in Amazon Pinpoint

In Amazon Pinpoint, a *project* is a collection of settings, data, campaigns, and segments that all share a common purpose. In the Amazon Pinpoint API, projects are also referred to as *applications*. This section uses the word "project" exclusively when referring to this concept.

To start sending push notifications in Amazon Pinpoint, you have to create a project. Next, you have to enable the push notification channels that you want to use by providing the appropriate credentials.

You can create new projects and set up push notification channels by using the Amazon Pinpoint management console. For more information, see [Setting Up Amazon Pinpoint Push Notification Channels](#) in the *Amazon Pinpoint User Guide*.

You can also create and set up projects by using the [Amazon Pinpoint API](#), by using an [AWS SDK](#), or by using the [AWS Command Line Interface](#). To create a project, use the `Apps` resource. To configure push notification channels, use the following resources:

- [APNs Channel](#) to send messages to users of iOS devices by using Apple Push Notification service.
- [ADM Channel](#) to send messages to users of Amazon Kindle Fire devices.
- [Baidu Channel](#) to send messages to Baidu users.
- [GCM Channel](#) to send messages to Android devices by using Firebase Cloud Messaging and Google Cloud Messaging.

Handling Push Notifications

After you obtain the credentials that are required to send push notifications, you can update your apps so that they're able to receive push notifications. For more information, see the following sections in the AWS Amplify documentation:

- **Firebase Cloud Messaging:** [Handling FCM/GCM Push Notifications](#)
- **Apple Push Notification service:** [Add Amazon Pinpoint Targeted and Campaign Push Messaging](#)
- **Amazon Device Messaging:** [Handling Amazon Device Messaging Push Notifications](#)
- **Baidu Push:** [Handling Baidu Push Notifications](#)

Defining Your Audience to Amazon Pinpoint

In Amazon Pinpoint, each member of your audience is represented by one or more endpoints. When you use Amazon Pinpoint to send a message, you direct the message to the endpoints that represent the members of your target audience. Each endpoint definition includes a message destination—such as a device token, email address, or phone number. It also includes data about your users and their devices. Before you analyze, segment, or engage your audience, the first step is to add endpoints to your Amazon Pinpoint project.

To add endpoints, you can:

- Integrate Amazon Pinpoint with your Android, iOS, or JavaScript client so that endpoints are added automatically when users visit your application.
- Use the Amazon Pinpoint API to add endpoints individually or in batches.
- Import endpoint definitions that are stored outside of Amazon Pinpoint.

After you add endpoints, you can:

- View analytics about your audience in the Amazon Pinpoint console.
- Learn about your audience by looking up or exporting endpoint data.
- Define audience segments based on endpoint attributes, such as demographic data or user interests.
- Engage your target audiences with tailored messaging campaigns.
- Send messages directly to lists of endpoints.

Use the topics in this section to add, update, and delete endpoints by using the Amazon Pinpoint API. If you want to add endpoints automatically from your Android, iOS, or JavaScript client, see [Registering Endpoints in Your Application \(p. 5\)](#) instead.

Topics

- [Adding Endpoints to Amazon Pinpoint \(p. 10\)](#)
- [Associating Users with Amazon Pinpoint Endpoints \(p. 15\)](#)
- [Adding a Batch of Endpoints to Amazon Pinpoint \(p. 19\)](#)
- [Importing Endpoints into Amazon Pinpoint \(p. 24\)](#)
- [Deleting Endpoints from Amazon Pinpoint \(p. 32\)](#)

Adding Endpoints to Amazon Pinpoint

An *endpoint* represents a destination that you can message—such as a mobile device, phone number, or email address. Before you can message a member of your audience, you must define one or more endpoints for that individual.

When you define an endpoint, you specify the *channel* and *address*. The channel is the type of platform that you use to message the endpoint. Examples of channels include a push notification service, SMS, or email. The address specifies where to message the endpoint, such as a device token, phone number, or email address.

To add more details about your audience, you can enrich your endpoints with custom and standard attributes. These attributes might include data about your users, their preferences, their devices, the versions of the client that they use, or their locations. When you add this type of data to your endpoints, you're able to:

- View charts about your audience in the Amazon Pinpoint console.
- Segment your audience based on endpoint attributes so that you can send your messages to the right target audience.
- Personalize your messages by incorporating message variables that are substituted with endpoint attribute values.

A mobile or JavaScript client application registers endpoints automatically if you integrate Amazon Pinpoint by using the AWS Mobile SDKs or the AWS Amplify JavaScript library. The client registers an endpoint for each new user, and it updates endpoints for returning users. To register endpoints from a mobile or JavaScript client, see [Registering Endpoints in Your Application \(p. 5\)](#).

Examples

The following examples show you how to add an endpoint to an Amazon Pinpoint project. The endpoint represents an audience member who lives in Seattle and uses an iPhone. This person can be messaged through the Apple Push Notification service (APNs). The endpoint's address is the device token that's provided by APNs.

AWS CLI

You can use Amazon Pinpoint by running commands with the AWS CLI.

Example Update Endpoint Command

To add or update an endpoint, use the `update-endpoint` command:

```
$ aws pinpoint update-endpoint \  
> --application-id application-id \  
> --endpoint-id endpoint-id \  
> --endpoint-request file://endpoint-request-file.json
```

Where:

- *application-id* is the ID of the Amazon Pinpoint project in which you're adding or updating an endpoint.
- *example-endpoint* is the ID that you're assigning to a new endpoint, or it's the ID of an existing endpoint that you're updating.
- *endpoint-request-file.json* is the file path to a local JSON file that contains the input for the `--endpoint-request` parameter.

Example Endpoint Request File

The example `update-endpoint` command uses a JSON file as the argument for the `--endpoint-request` parameter. This file contains an endpoint definition like the following:

```
{  
  "ChannelType": "APNS",  
  "Address": "1a2b3c4d5e6f7g8h9i0j1k2l3m4n5o6p7q8r9s0t1u2v3w4x5y6z7a8b9c0d1e2f",  
  "Attributes": {  
    "interests": [  

```



```
        "technology",
        "music",
        "travel"
    ]
},
"Metrics": {
    "technology_interest_level": 9.0,
    "music_interest_level": 6.0,
    "travel_interest_level": 4.0
},
"Demographic": {
    "AppVersion": "1.0",
    "Make": "apple",
    "Model": "iPhone",
    "ModelVersion": "8",
    "Platform": "ios",
    "PlatformVersion": "11.3.1",
    "Timezone": "America/Los_Angeles"
},
"Location": {
    "Country": "US",
    "City": "Seattle",
    "PostalCode": "98121",
    "Latitude": 47.61,
    "Longitude": -122.33
}
}
```

For the attributes that you can use to define an endpoint, see the [EndpointRequest](#) schema in the *Amazon Pinpoint API Reference*.

AWS SDK for Java

You can use the Amazon Pinpoint API in your Java applications by using the client that's provided by the AWS SDK for Java.

Example Code

To add an endpoint, initialize an [EndpointRequest](#) object, and pass it to the [updateEndpoint](#) method of the AmazonPinpoint client:

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.*;
import java.util.Arrays;

public class AddExampleEndpoint {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "AddExampleEndpoint - Adds an example endpoint to an Amazon Pinpoint application." +
            "Usage: AddExampleEndpoint <applicationId>" +
            "Where:\n" +
            " applicationId - The ID of the Amazon Pinpoint application to add the example " +
            "endpoint to.";

        if (args.length < 1) {
            System.out.println(USAGE);
            System.exit(1);
        }
    }
}
```

```
String applicationId = args[0];

// The device token assigned to the user's device by Apple Push Notification service
(APNs).
String deviceToken =
"1a2b3c4d5e6f7g8h9i0j1k2l3m4n5o6p7q8r9s0t1u2v3w4x5y6z7a8b9c0d1e2f";

// Initializes an endpoint definition with channel type and address.
EndpointRequest wangXiulansIphoneEndpoint = new EndpointRequest()
    .withChannelType(ChannelType.APNS)
    .withAddress(deviceToken);

// Adds custom attributes to the endpoint.
wangXiulansIphoneEndpoint.addAttributesEntry("interests", Arrays.asList(
    "technology",
    "music",
    "travel"));

// Adds custom metrics to the endpoint.
wangXiulansIphoneEndpoint.addMetricsEntry("technology_interest_level", 9.0);
wangXiulansIphoneEndpoint.addMetricsEntry("music_interest_level", 6.0);
wangXiulansIphoneEndpoint.addMetricsEntry("travel_interest_level", 4.0);

// Adds standard demographic attributes.
wangXiulansIphoneEndpoint.setDemographic(new EndpointDemographic()
    .withAppVersion("1.0")
    .withMake("apple")
    .withModel("iPhone")
    .withModelVersion("8")
    .withPlatform("ios")
    .withPlatformVersion("11.3.1")
    .withTimezone("America/Los_Angeles"));

// Adds standard location attributes.
wangXiulansIphoneEndpoint.setLocation(new EndpointLocation()
    .withCountry("US")
    .withCity("Seattle")
    .withPostalCode("98121")
    .withLatitude(47.61)
    .withLongitude(-122.33));

// Initializes the Amazon Pinpoint client.
AmazonPinpoint pinpointClient = AmazonPinpointClientBuilder.standard()
    .withRegion(Regions.US_EAST_1).build();

// Updates or creates the endpoint with Amazon Pinpoint.
UpdateEndpointResult result = pinpointClient.updateEndpoint(new
UpdateEndpointRequest()
    .withApplicationId(applicationId)
    .withEndpointId("example_endpoint")
    .withEndpointRequest(wangXiulansIphoneEndpoint));

System.out.format("Update endpoint result: %s\n",
result.getMessageBody().getMessage());

}
}
```

HTTP

You can use Amazon Pinpoint by making HTTP requests directly to the REST API.

Example PUT Endpoint Request

To add an endpoint, issue a **PUT** request to the [Endpoint](#) resource at the following URI:

/v1/apps/*application-id*/endpoints/*endpoint-id*

Where:

- *application-id* is the ID of the Amazon Pinpoint project in which you're adding or updating an endpoint.
- *endpoint-id* is the ID that you're assigning to a new endpoint, or it's the ID of an existing endpoint that you're updating.

In your request, include the required headers, and provide the [EndpointRequest](#) JSON as the body:

```
PUT /v1/apps/application_id/endpoints/example_endpoint HTTP/1.1
Host: pinpoint.us-east-1.amazonaws.com
X-Amz-Date: 20180415T182538Z
Content-Type: application/json
Accept: application/json
X-Amz-Date: 20180428T004705Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20180428/us-east-1/mobiletargeting/aws4_request, SignedHeaders=accept;content-length;content-type;host;x-amz-date, Signature=c25cbd6bf61bd3b3667c571ae764b9bf2d8af61b875caced95d1e68d91b4170
Cache-Control: no-cache

{
  "ChannelType": "APNS",
  "Address": "1a2b3c4d5e6f7g8h9i0j1k2l3m4n5o6p7q8r9s0t1u2v3w4x5y6z7a8b9c0d1e2f",
  "Attributes": {
    "interests": [
      "technology",
      "music",
      "travel"
    ]
  },
  "Metrics": {
    "technology_interest_level": 9.0,
    "music_interest_level": 6.0,
    "travel_interest_level": 4.0
  },
  "Demographic": {
    "AppVersion": "1.0",
    "Make": "apple",
    "Model": "iPhone",
    "ModelVersion": "8",
    "Platform": "ios",
    "PlatformVersion": "11.3.1",
    "Timezone": "America/Los_Angeles"
  },
  "Location": {
    "Country": "US",
    "City": "Seattle",
    "PostalCode": "98121",
    "Latitude": 47.61,
    "Longitude": -122.33
  }
}
```

If your request succeeds, you receive a response like the following:

```
{
  "RequestID": "67e572ed-41d5-11e8-9dc5-db288f3cbb72",
  "Message": "Accepted"
}
```

Related Information

For more information about the Endpoint resource in the Amazon Pinpoint API, including the supported HTTP methods and request parameters, see [Endpoint](#) in the *Amazon Pinpoint API Reference*.

For more information about personalizing messages with variables, see [Message Variables](#) in the *Amazon Pinpoint User Guide*.

For the limits that apply to endpoints, such as the number of attributes you can assign, see [the section called "Endpoint Limits" \(p. 138\)](#).

Associating Users with Amazon Pinpoint Endpoints

An endpoint can include attributes that define a *user*, which represents a person in your audience. For example, a user might represent someone who installed your mobile app, or someone who has an account on your website.

You define a user by specifying a unique user ID and, optionally, custom user attributes. If someone uses your app on multiple devices, or if that person can be messaged at multiple addresses, you can assign the same user ID to multiple endpoints. In this case, Amazon Pinpoint synchronizes user attributes across the endpoints. So, if you add a user attribute to one endpoint, Amazon Pinpoint adds that attribute to each endpoint that includes the same user ID.

You can add user attributes to track data that applies to an individual and doesn't vary based on which device the person is using. For example, you can add attributes for a person's name, age, or account status.

Tip

If your application uses Amazon Cognito user pools to handle user authentication, Amazon Cognito can add user IDs and attributes to your endpoints automatically. For the endpoint user ID value, Amazon Cognito assigns the `sub` value that's assigned to the user in the user pool. To add users with Amazon Cognito, see [Using Amazon Pinpoint Analytics with Amazon Cognito User Pools](#).

After you add user definitions to your endpoints, you have more options for how you segment your audience. You can define a segment based on user attributes, or you can define a segment by importing a list of user IDs. When you send a message to a segment that's based on users, the potential destinations include each endpoint that's associated with each user in the segment.

You also have more options for how you message your audience. You can use a campaign to message a segment of users, or you can send a message directly to a list of user IDs. To personalize your message, you can include message variables that are substituted with user attribute values.

Examples

The following examples show you how to add a user definition to an endpoint.

AWS CLI

You can use Amazon Pinpoint by running commands with the AWS CLI.

Example Update Endpoint Command

To add a user to an endpoint, use the `update-endpoint` command. For the `--endpoint-request` parameter, you can define a new endpoint, which can include a user. Or, to update an existing endpoint, you can provide just the attributes that you want to change. The following example adds a user to an existing endpoint by providing only the user attributes:

```
$ aws pinpoint update-endpoint \  
> --application-id application-id \  
> --endpoint-id endpoint-id \  
> --endpoint-request file://endpoint-request-file.json
```

Where:

- *application-id* is the ID of the Amazon Pinpoint project in which you're adding or updating an endpoint.
- *endpoint-id* is the ID that you're assigning to a new endpoint, or it's the ID of an existing endpoint that you're updating.
- *endpoint-request-file.json* is the file path to a local JSON file that contains the input for the `--endpoint-request` parameter.

Example Endpoint Request File

The example `update-endpoint` command uses a JSON file as the argument for the `--endpoint-request` parameter. This file contains a user definition like the following:

```
{  
  "User": {  
    "UserId": "example_user",  
    "UserAttributes": {  
      "name": [  
        "Wang",  
        "Xiulan"  
      ],  
      "gender": [  
        "female"  
      ],  
      "age": [  
        "39"  
      ]  
    }  
  }  
}
```

For the attributes that you can use to define a user, see the `User` object in the [EndpointRequest](#) schema in the *Amazon Pinpoint API Reference*.

AWS SDK for Java

You can use the Amazon Pinpoint API in your Java applications by using the client that's provided by the AWS SDK for Java.

Example Code

To add a user to an endpoint, initialize an [EndpointRequest](#) object, and pass it to the `updateEndpoint` method of the `AmazonPinpoint` client. You can use this object to define a new endpoint, which can include a user. Or, to update an existing endpoint, you can update just the properties that you want to change. The following example adds a user to an existing endpoint by adding an [EndpointUser](#) object to the `EndpointRequest` object:

```
import com.amazonaws.regions.Regions;  
import com.amazonaws.services.pinpoint.AmazonPinpoint;  
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;  
import com.amazonaws.services.pinpoint.model.EndpointRequest;  
import com.amazonaws.services.pinpoint.model.EndpointUser;
```

```
import com.amazonaws.services.pinpoint.model.UpdateEndpointRequest;
import com.amazonaws.services.pinpoint.model.UpdateEndpointResult;

import java.util.Arrays;
import java.util.Collections;

public class AddExampleUser {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "AddExampleUser - Adds a user definition to the specified Amazon
Pinpoint endpoint." +
            "Usage: AddExampleUser <endpointId> <applicationId>" +
            "Where:\n" +
            "  endpointId - The ID of the endpoint to add the user to." +
            "  applicationId - The ID of the Amazon Pinpoint application that
contains the endpoint.";

        if (args.length < 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String endpointId = args[0];
        String applicationId = args[1];

        // Creates a new user definition.
        EndpointUser wangXiulan = new EndpointUser().withUserId("example_user");

        // Assigns custom user attributes.
        wangXiulan.addUserAttributesEntry("name", Arrays.asList("Wang", "Xiulan"));
        wangXiulan.addUserAttributesEntry("gender",
Collections.singletonList("female"));
        wangXiulan.addUserAttributesEntry("age", Collections.singletonList("39"));

        // Adds the user definition to the EndpointRequest that is passed to the Amazon
Pinpoint client.
        EndpointRequest wangXiulansIphone = new EndpointRequest()
            .withUser(wangXiulan);

        // Initializes the Amazon Pinpoint client.
        AmazonPinpoint pinpointClient = AmazonPinpointClientBuilder.standard()
            .withRegion(Regions.US_EAST_1).build();

        // Updates the specified endpoint with Amazon Pinpoint.
        UpdateEndpointResult result = pinpointClient.updateEndpoint(new
UpdateEndpointRequest()
            .withEndpointRequest(wangXiulansIphone)
            .withApplicationId(applicationId)
            .withEndpointId(endpointId));

        System.out.format("Update endpoint result: %s\n",
result.getMessageBody().getMessage());

    }
}
```

HTTP

You can use Amazon Pinpoint by making HTTP requests directly to the REST API.

Example Put Endpoint Request with User Definition

To add a user to an endpoint, issue a `PUT` request to the [Endpoint](#) resource at the following URI:

`/v1/apps/application-id/endpoints/endpoint-id`

Where:

- *application-id* is the ID of the Amazon Pinpoint project in which you're adding or updating an endpoint.
- *endpoint-id* is the ID that you're assigning to a new endpoint, or it's the ID of an existing endpoint that you're updating.

In your request, include the required headers, and provide the [EndpointRequest](#) JSON as the body. The request body can define a new endpoint, which can include a user. Or, to update an existing endpoint, you can provide just the attributes that you want to change. The following example adds a user to an existing endpoint by providing only the user attributes:

```
PUT /v1/apps/application_id/endpoints/example_endpoint HTTP/1.1
Host: pinpoint.us-east-1.amazonaws.com
X-Amz-Date: 20180415T182538Z
Content-Type: application/json
Accept: application/json
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20180501/us-east-1/mobiletargeting/aws4_request, SignedHeaders=accept;content-length;content-type;host;x-amz-date, Signature=c25cbd6bf61bd3b3667c571ae764b9bf2d8af61b875caced95d1e68d91b4170
Cache-Control: no-cache

{
  "User": {
    "UserId": "example_user",
    "UserAttributes": {
      "name": [
        "Wang",
        "Xiulan"
      ],
      "gender": [
        "female"
      ],
      "age": [
        "39"
      ]
    }
  }
}
```

If the request succeeds, you receive a response like the following:

```
{
  "RequestID": "67e572ed-41d5-11e8-9dc5-db288f3cbb72",
  "Message": "Accepted"
}
```

Related Information

For more information about the Endpoint resource in the Amazon Pinpoint API, including the supported HTTP methods and request parameters, see [Endpoint](#) in the *Amazon Pinpoint API Reference*.

For more information about personalizing messages with variables, see [Message Variables](#) in the *Amazon Pinpoint User Guide*.

To define a segment by importing a list of user IDs, see [Importing Segments](#) in the *Amazon Pinpoint User Guide*.

To send a direct message to up to 100 user IDs, see [Users Messages](#) in the *Amazon Pinpoint API Reference*.

For the limits that apply to endpoints, including the number of user attributes you can assign, see [the section called "Endpoint Limits" \(p. 138\)](#).

Adding a Batch of Endpoints to Amazon Pinpoint

You can add or update multiple endpoints in a single operation by providing the endpoints in batches. Each batch request can include up to 100 endpoint definitions.

If you want to add or update more than 100 endpoints in a single operation, see [Importing Endpoints into Amazon Pinpoint \(p. 24\)](#) instead.

Examples

The following examples show you how to add two endpoints at once by including the endpoints in a batch request.

AWS CLI

You can use Amazon Pinpoint by running commands with the AWS CLI.

Example Update Endpoints Batch Command

To submit an endpoint batch request, use the `update-endpoints-batch` command:

```
$ aws pinpoint update-endpoints-batch \  
> --application-id application-id \  
> --endpoint-batch-request file://endpoint_batch_request_file.json
```

Where:

- *application-id* is the ID of the Amazon Pinpoint project in which you're adding or updating the endpoints.
- *endpoint_batch_request_file.json* is the file path to a local JSON file that contains the input for the `--endpoint-batch-request` parameter.

Example Endpoint Batch Request File

The example `update-endpoints-batch` command uses a JSON file as the argument for the `--endpoint-request` parameter. This file contains a batch of endpoint definitions like the following:

```
{  
  "Item": [  
    {  
      "ChannelType": "EMAIL",  
      "Address": "richard_roe@example.com",  
      "Attributes": {  
        "interests": [  
          "music",  
          "books"  
        ]  
      },  
      "Metrics": {  
        "music_interest_level": 3.0,  
      }  
    }  
  ]  
}
```



```
        "books_interest_level": 7.0
      },
      "Id": "example_endpoint_1",
      "User": {
        "UserId": "example_user_1",
        "UserAttributes": {
          "name": [
            "Richard",
            "Roe"
          ]
        }
      }
    },
    {
      "ChannelType": "SMS",
      "Address": "+16145550100",
      "Attributes": {
        "interests": [
          "cooking",
          "politics",
          "finance"
        ]
      },
      "Metrics": {
        "cooking_interest_level": 5.0,
        "politics_interest_level": 8.0,
        "finance_interest_level": 4.0
      },
      "Id": "example_endpoint_2",
      "User": {
        "UserId": "example_user_2",
        "UserAttributes": {
          "name": [
            "Mary",
            "Major"
          ]
        }
      }
    }
  ]
}
```

For the attributes that you can use to define a batch of endpoints, see the [EndpointBatchRequest](#) schema in the *Amazon Pinpoint API Reference*.

AWS SDK for Java

You can use the Amazon Pinpoint API in your Java applications by using the client that's provided by the AWS SDK for Java.

Example Code

To submit an endpoint batch request, initialize an [EndpointBatchRequest](#) object, and pass it to the `updateEndpointsBatch` method of the `AmazonPinpoint` client. The following example populates an `EndpointBatchRequest` object with two `EndpointBatchItem` objects:

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.ChannelType;
import com.amazonaws.services.pinpoint.model.EndpointBatchItem;
import com.amazonaws.services.pinpoint.model.EndpointBatchRequest;
```

```
import com.amazonaws.services.pinpoint.model.EndpointUser;
import com.amazonaws.services.pinpoint.model.UpdateEndpointsBatchRequest;
import com.amazonaws.services.pinpoint.model.UpdateEndpointsBatchResult;

import java.util.Arrays;

public class AddExampleEndpoints {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "AddExampleEndpoints - Adds example endpoints to an Amazon Pinpoint
application." +
            "Usage: AddExampleEndpoints <applicationId>" +
            "Where:\n" +
            "  applicationId - The ID of the Amazon Pinpoint application to add the
example endpoints to.";

        if (args.length < 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String applicationId = args[0];

        // Initializes an endpoint definition with channel type, address, and ID.
        EndpointBatchItem richardRoesEmailEndpoint = new EndpointBatchItem()
            .withChannelType(ChannelType.EMAIL)
            .withAddress("richard_roe@example.com")
            .withId("example_endpoint_1");

        // Adds custom attributes to the endpoint.
        richardRoesEmailEndpoint.addAttributesEntry("interests", Arrays.asList(
            "music",
            "books"));

        // Adds custom metrics to the endpoint.
        richardRoesEmailEndpoint.addMetricsEntry("music_interest_level", 3.0);
        richardRoesEmailEndpoint.addMetricsEntry("books_interest_level", 7.0);

        // Initializes a user definition with a user ID.
        EndpointUser richardRoe = new EndpointUser().withUserId("example_user_1");

        // Adds custom user attributes.
        richardRoe.addUserAttributesEntry("name", Arrays.asList("Richard", "Roe"));

        // Adds the user definition to the endpoint.
        richardRoesEmailEndpoint.setUser(richardRoe);

        // Initializes an endpoint definition with channel type, address, and ID.
        EndpointBatchItem maryMajorsSmsEndpoint = new EndpointBatchItem()
            .withChannelType(ChannelType.SMS)
            .withAddress("+16145550100")
            .withId("example_endpoint_2");

        // Adds custom attributes to the endpoint.
        maryMajorsSmsEndpoint.addAttributesEntry("interests", Arrays.asList(
            "cooking",
            "politics",
            "finance"));

        // Adds custom metrics to the endpoint.
        maryMajorsSmsEndpoint.addMetricsEntry("cooking_interest_level", 5.0);
        maryMajorsSmsEndpoint.addMetricsEntry("politics_interest_level", 8.0);
        maryMajorsSmsEndpoint.addMetricsEntry("finance_interest_level", 4.0);
    }
}
```

```
// Initializes a user definition with a user ID.
EndpointUser maryMajor = new EndpointUser().withUserId("example_user_2");

// Adds custom user attributes.
maryMajor.addUserAttributesEntry("name", Arrays.asList("Mary", "Major"));

// Adds the user definition to the endpoint.
maryMajorsSmsEndpoint.setUser(maryMajor);

// Adds multiple endpoint definitions to a single request object.
EndpointBatchRequest endpointList = new EndpointBatchRequest()
    .withItem(richardRoesEmailEndpoint)
    .withItem(maryMajorsSmsEndpoint);

// Initializes the Amazon Pinpoint client.
AmazonPinpoint pinpointClient = AmazonPinpointClientBuilder.standard()
    .withRegion(Regions.US_EAST_1).build();

// Updates or creates the endpoints with Amazon Pinpoint.
UpdateEndpointsBatchResult result = pinpointClient.updateEndpointsBatch(
    new UpdateEndpointsBatchRequest()
        .withApplicationId(applicationId)
        .withEndpointBatchRequest(endpointList));

System.out.format("Update endpoints batch result: %s\n",
    result.getMessageBody().getMessage());

}

}
```

HTTP

You can use Amazon Pinpoint by making HTTP requests directly to the REST API.

Example Put Endpoints Request

To submit an endpoint batch request, issue a PUT request to the [Endpoints](#) resource at the following URI:

```
/v1/apps/application-id/endpoints
```

Where *application-id* is the ID of the Amazon Pinpoint project in which you're adding or updating the endpoints.

In your request, include the required headers, and provide the [EndpointBatchRequest](#) JSON as the body:

```
PUT /v1/apps/application_id/endpoints HTTP/1.1
Host: pinpoint.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/json
X-Amz-Date: 20180501T184948Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20180501/us-east-1/mobiletargeting/aws4_request, SignedHeaders=accept;content-length;content-type;host;x-amz-date, Signature=c25cbd6bf61bd3b3667c571ae764b9bf2d8af61b875caced95d1e68d91b4170
Cache-Control: no-cache

{
  "Item": [
    {
      "ChannelType": "EMAIL",
```

```
"Address": "richard_roe@example.com",
"Attributes": {
  "interests": [
    "music",
    "books"
  ]
},
"Metrics": {
  "music_interest_level": 3.0,
  "books_interest_level": 7.0
},
"Id": "example_endpoint_1",
"User": {
  "UserId": "example_user_1",
  "UserAttributes": {
    "name": [
      "Richard",
      "Roe"
    ]
  }
}
},
{
  "ChannelType": "SMS",
  "Address": "+16145550100",
  "Attributes": {
    "interests": [
      "cooking",
      "politics",
      "finance"
    ]
  },
  "Metrics": {
    "cooking_interest_level": 5.0,
    "politics_interest_level": 8.0,
    "finance_interest_level": 4.0
  },
  "Id": "example_endpoint_2",
  "User": {
    "UserId": "example_user_2",
    "UserAttributes": {
      "name": [
        "Mary",
        "Major"
      ]
    }
  }
}
]
}
```

If your request succeeds, you receive a response like the following:

```
{
  "RequestID": "67e572ed-41d5-11e8-9dc5-db288f3cbb72",
  "Message": "Accepted"
}
```

Related Information

For more information about the Endpoint resource in the Amazon Pinpoint API, including the supported HTTP methods and request parameters, see [Endpoint](#) in the *Amazon Pinpoint API Reference*.

Importing Endpoints into Amazon Pinpoint

You can add or update endpoints in large numbers by importing them from an Amazon S3 bucket. Importing endpoints is useful if you have records about your audience outside of Amazon Pinpoint, and you want to add this information to an Amazon Pinpoint project. In this case, you would:

1. Create endpoint definitions that are based on your own audience data.
2. Save these endpoint definitions in one or more files, and upload the files to an Amazon S3 bucket.
3. Add the endpoints to your Amazon Pinpoint project by importing them from the bucket.

Each import job can transfer up to 1 GB of data. In a typical job, where each endpoint is 4 KB or less, you could import around 250,000 endpoints. You can run up to two concurrent import jobs per AWS account. If you need more bandwidth for your import jobs, you can submit a service limit increase request with AWS Support. For more information, see [Requesting a Limit Increase \(p. 145\)](#).

Before You Begin

Before you can import endpoints, you need the following resources in your AWS account:

- An Amazon S3 bucket. To create a bucket, see [Create a Bucket](#) in the *Amazon Simple Storage Service Getting Started Guide*.
- An AWS Identity and Access Management (IAM) role that grants Amazon Pinpoint read permissions for your Amazon S3 bucket. To create the role, see [IAM Role for Importing Endpoints or Segments \(p. 130\)](#).

Examples

The following examples demonstrate how to add endpoint definitions to your Amazon S3 bucket, and then import those endpoints into an Amazon Pinpoint project.

Files with Endpoint Definitions

The files that you add to your Amazon S3 bucket can contain endpoint definitions in CSV or newline-delimited JSON format. For the attributes that you can use to define your endpoints, see the [EndpointRequest](#) JSON schema in the *Amazon Pinpoint API Reference*.

CSV

You can import endpoints that are defined in a CSV file, as in the following example:

```
ChannelType,Address,Location.Country, Demographic.Platform, Demographic.Make, User.UserId
SMS,2065550182,CAN,Android,LG,example-user-id-1
APNS,1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f,USA,iOS,Apple,example-user-id-2
EMAIL,john.stiles@example.com,USA,iOS,Apple,example-user-id-2
```

The first line is the header, which contains the endpoint attributes. Specify nested attributes by using dot notation, as in `Location.Country`.

The subsequent lines define the endpoints by providing values for each of the attributes in the header.

To include a comma, line break, or double quote in a value, enclose the value in double quotes, as in `"aaa,bbb"`. For more information about the CSV format, see [RFC 4180 Common Format and MIME Type for Comma-Separated Values \(CSV\) Files](#).

JSON

You can import endpoints that are defined in a newline-delimited JSON file, as in the following example:

```
{ "ChannelType": "SMS", "Address": "2065550182", "Location": { "Country": "CAN" }, "Demographic":  
{ "Platform": "Android", "Make": "LG" }, "User": { "UserId": "example-user-id-1" } }  
{ "ChannelType": "APNS", "Address": "1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f", "Location":  
{ "Country": "USA" }, "Demographic": { "Platform": "iOS", "Make": "Apple" }, "User":  
{ "UserId": "example-user-id-2" } }  
{ "ChannelType": "EMAIL", "Address": "john.stiles@example.com", "Location":  
{ "Country": "USA" }, "Demographic": { "Platform": "iOS", "Make": "Apple" }, "User":  
{ "UserId": "example-user-id-2" } }
```

In this format, each line is a complete JSON object that contains an individual endpoint definition.

Import Job Requests

The following examples show you how to add endpoint definitions to Amazon S3 by uploading a local file to a bucket. Then, the examples import the endpoint definitions into an Amazon Pinpoint project.

AWS CLI

You can use Amazon Pinpoint by running commands with the AWS CLI.

Example S3 CP Command

To upload a local file to an Amazon S3 bucket, use the Amazon S3 `cp` command:

```
$ aws s3 cp ./endpoints-file s3://bucket-name/prefix/
```

Where:

- *./endpoints-file* is the file path to a local file that contains the endpoint definitions.
- *bucket-name/prefix/* is the name of your Amazon S3 bucket and, optionally, a prefix that helps you organize the objects in your bucket hierarchically. For example, a useful prefix might be `pinpoint/imports/endpoints/`.

Example Create Import Job Command

To import endpoint definitions from an Amazon S3 bucket, use the `create-import-job` command:

```
$ aws pinpoint create-import-job \  
> --application-id application-id \  
> --import-job-request \  
> S3Url=s3://bucket-name/prefix/key,\  
> RoleArn=iam-import-role-arn,\  
> Format=format,\  
> RegisterEndpoints=true
```

Where:

- *application-id* is the ID of the Amazon Pinpoint project that you're importing endpoints for.
- *bucket-name/prefix/key* is the location in Amazon S3 that contains one or more objects to import. The location can end with the key for an individual object, or it can end with a prefix that qualifies multiple objects.

- *iam-import-role-arn* is the Amazon Resource Name (ARN) of an IAM role that grants Amazon Pinpoint read access to the bucket.
- *format* can be either JSON or CSV, depending on which format you used to define your endpoints. If the Amazon S3 location includes multiple objects of mixed formats, Amazon Pinpoint imports only the objects that match the specified format.

The response includes details about the import job:

```
{
  "ImportJobResponse": {
    "CreationDate": "2018-05-24T21:26:33.995Z",
    "Definition": {
      "DefineSegment": false,
      "ExternalId": "463709046829",
      "Format": "JSON",
      "RegisterEndpoints": true,
      "RoleArn": "iam-import-role-arn",
      "S3Url": "s3://bucket-name/prefix/key"
    },
    "Id": "d5ecad8e417d498389e1d5b9454d4e0c",
    "JobStatus": "CREATED",
    "Type": "IMPORT"
  }
}
```

The response provides the job ID with the `Id` attribute. You can use this ID to check the current status of the import job.

Example Get Import Job Command

To check the current status of an import job, use the `get-import-job` command:

```
$ aws pinpoint get-import-job \
> --application-id application-id \
> --job-id job-id
```

Where:

- *application-id* is the ID of the Amazon Pinpoint project that the import job was initiated for.
- *job-id* is the ID of the import job that you're checking.

The response to this command provides the current state of the import job:

```
{
  "ImportJobResponse": {
    "ApplicationId": "application-id",
    "CompletedPieces": 1,
    "CompletionDate": "2018-05-24T21:26:45.308Z",
    "CreationDate": "2018-05-24T21:26:33.995Z",
    "Definition": {
      "DefineSegment": false,
      "ExternalId": "463709046829",
      "Format": "JSON",
      "RegisterEndpoints": true,
      "RoleArn": "iam-import-role-arn",
      "S3Url": "s3://s3-bucket-name/prefix/endpoint-definitions.json"
    },
    "FailedPieces": 0,
    "Id": "job-id",
  }
}
```

```
        "JobStatus": "COMPLETED",
        "TotalFailures": 0,
        "TotalPieces": 1,
        "TotalProcessed": 3,
        "Type": "IMPORT"
    }
}
```

The response provides the job status with the `JobStatus` attribute.

AWS SDK for Java

You can use the Amazon Pinpoint API in your Java applications by using the client that's provided by the AWS SDK for Java.

Example Code

To upload a file with endpoint definitions to Amazon S3, use the `putObject` method of the `AmazonS3` client.

To import the endpoints into an Amazon Pinpoint project, initialize a `CreateImportJobRequest` object. Then, pass this object to the `createImportJob` method of the `AmazonPinpoint` client.

```
package com.amazonaws.examples.pinpoint;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.CreateImportJobRequest;
import com.amazonaws.services.pinpoint.model.CreateImportJobResult;
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.ImportJobRequest;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import java.io.File;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;
import java.util.concurrent.TimeUnit;

public class ImportEndpoints {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "ImportEndpoints - Adds endpoints to an Amazon Pinpoint application by: \n" +
            "1.) Uploading the endpoint definitions to an Amazon S3 bucket. \n" +
            "2.) Importing the endpoint definitions from the bucket to an Amazon Pinpoint "
            +
            "application.\n\n" +
            "Usage: ImportEndpoints <endpointsFileLocation> <s3BucketName>
            <iamImportRoleArn> " +
            "<applicationId>\n\n" +
            "Where:\n" +
            "  endpointsFileLocation - The relative location of the JSON file that contains
            the " +
            "  endpoint definitions.\n" +
            "  s3BucketName - The name of the Amazon S3 bucket to upload the JSON file to.
            If the " +
            "  bucket doesn't exist, a new bucket is created.\n" +
```



```
    " iamImportRoleArn - The ARN of an IAM role that grants Amazon Pinpoint read "
+
    "permissions to the S3 bucket.\n" +
    " applicationId - The ID of the Amazon Pinpoint application to add the
endpoints to.";

    if (args.length < 1) {
        System.out.println(USAGE);
        System.exit(1);
    }

    String endpointsFileLocation = args[0];
    String s3BucketName = args[1];
    String iamImportRoleArn = args[2];
    String applicationId = args[3];

    Path endpointsFilePath = Paths.get(endpointsFileLocation);
    File endpointsFile = new File(endpointsFilePath.toAbsolutePath().toString());
    uploadToS3(endpointsFile, s3BucketName);

    importToPinpoint(endpointsFile.getName(), s3BucketName, iamImportRoleArn,
applicationId);
}

private static void uploadToS3(File endpointsFile, String s3BucketName) {

    // Initializes Amazon S3 client.
    final AmazonS3 s3 = AmazonS3ClientBuilder.defaultClient();

    // Checks whether the specified bucket exists. If not, attempts to create one.
    if (!s3.doesBucketExistV2(s3BucketName)) {
        try {
            s3.createBucket(s3BucketName);
            System.out.format("Created S3 bucket %s.\n", s3BucketName);
        } catch (AmazonS3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Uploads the endpoints file to the bucket.
    String endpointsFileName = endpointsFile.getName();
    System.out.format("Uploading %s to S3 bucket %s . . .\n", endpointsFileName,
s3BucketName);
    try {
        s3.putObject(s3BucketName, "imports/" + endpointsFileName, endpointsFile);
        System.out.println("Finished uploading to S3.");
    } catch (AmazonServiceException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

private static void importToPinpoint(String endpointsFileName, String s3BucketName,
String iamImportRoleArn, String applicationId)
{

    // The S3 URL that Amazon Pinpoint requires to find the endpoints file.
    String s3Url = "s3://" + s3BucketName + "/imports/" + endpointsFileName;

    // Defines the import job that Amazon Pinpoint runs.
    ImportJobRequest importJobRequest = new ImportJobRequest()
        .withS3Url(s3Url)
        .withRegisterEndpoints(true)
        .withRoleArn(iamImportRoleArn)
```

```
        .withFormat(Format.JSON);
CreateImportJobRequest createImportJobRequest = new CreateImportJobRequest()
        .withApplicationId(applicationId)
        .withImportJobRequest(importJobRequest);

// Initializes the Amazon Pinpoint client.
AmazonPinpoint pinpointClient = AmazonPinpointClientBuilder.standard()
        .withRegion(Regions.US_EAST_1).build();

System.out.format("Importing endpoints in %s to Amazon Pinpoint application
%s . . .\n",
        endpointsFileName, applicationId);

try {

    // Runs the import job with Amazon Pinpoint.
    CreateImportJobResult importResult =
        pinpointClient.createImportJob(createImportJobRequest);

    String jobId = importResult.getImportJobResponse().getId();
    GetImportJobResult getImportJobResult = null;
    String jobStatus = null;

    // Checks the job status until the job completes or fails.
    do {
        getImportJobResult = pinpointClient.getImportJob(new
GetImportJobRequest()
                .withJobId(jobId)
                .withApplicationId(applicationId));
        jobStatus = getImportJobResult.getImportJobResponse().getJobStatus();
        System.out.format("Import job %s . . .\n", jobStatus.toLowerCase());
        TimeUnit.SECONDS.sleep(3);
    } while (!jobStatus.equals("COMPLETED") && !jobStatus.equals("FAILED"));

    if (jobStatus.equals("COMPLETED")) {
        System.out.println("Finished importing endpoints.");
    } else {
        System.err.println("Failed to import endpoints.");
        System.exit(1);
    }

    // Checks for entries that failed to import.
    // getFailures provides up to 100 of the first failed entries for the job,
if any exist.
    List<String> failedEndpoints =
getImportJobResult.getImportJobResponse().getFailures();
    if (failedEndpoints != null) {
        System.out.println("Failed to import the following entries:");
        for (String failedEndpoint : failedEndpoints) {
            System.out.println(failedEndpoint);
        }
    }

    } catch (AmazonServiceException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

HTTP

You can use Amazon Pinpoint by making HTTP requests directly to the REST API.

Example S3 PUT Object Request

To add your endpoint definitions to a bucket, use the Amazon S3 [PUT Object](#) operation, and provide the endpoint definitions as the body:

```
PUT /prefix/key HTTP/1.1
Content-Type: text/plain
Accept: application/json
Host: bucket-name.s3.amazonaws.com
X-Amz-Content-Sha256: c430dc094b0cec2905bc88d96314914d058534b14e2bc6107faa9daa12fdff2d
X-Amz-Date: 20180605T184132Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20180605/
us-east-1/s3/aws4_request, SignedHeaders=accept;cache-control;content-
length;content-type;host;postman-token;x-amz-content-sha256;x-amz-date,
Signature=c25cbd6bf61bd3b3667c571ae764b9bf2d8af61b875caced95d1e68d91b4170
Cache-Control: no-cache

{"ChannelType": "SMS", "Address": "2065550182", "Location": {"Country": "CAN"}, "Demographic":
{"Platform": "Android", "Make": "LG"}, "User": {"UserId": "example-user-id-1"}}
{"ChannelType": "APNS", "Address": "1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f", "Location":
{"Country": "USA"}, "Demographic": {"Platform": "iOS", "Make": "Apple"}, "User":
{"UserId": "example-user-id-2"}}
{"ChannelType": "EMAIL", "Address": "john.stiles@example.com", "Location":
{"Country": "USA"}, "Demographic": {"Platform": "iOS", "Make": "Apple"}, "User":
{"UserId": "example-user-id-2"}}
```

Where:

- */prefix/key* is the prefix and key name for the object that will contain the endpoint definitions after the upload. You can use the prefix to organize your objects hierarchically. For example, a useful prefix might be `pinpoint/imports/endpoints/`.
- *bucket-name* is the name of the Amazon S3 bucket that you're adding the endpoint definitions to.

Example POST Import Job Request

To import endpoint definitions from an Amazon S3 bucket, issue a POST request to the [Import Jobs](#) resource. In your request, include the required headers and provide the [ImportJobRequest](#) JSON as the body:

```
POST /v1/apps/application_id/jobs/import HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: pinpoint.us-east-1.amazonaws.com
X-Amz-Date: 20180605T214912Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20180605/
us-east-1/mobiletargeting/aws4_request, SignedHeaders=accept;cache-
control;content-length;content-type;host;postman-token;x-amz-date,
Signature=c25cbd6bf61bd3b3667c571ae764b9bf2d8af61b875caced95d1e68d91b4170
Cache-Control: no-cache

{
  "S3Url": "s3://bucket-name/prefix/key",
  "RoleArn": "iam-import-role-arn",
  "Format": "format",
  "RegisterEndpoints": true
}
```

Where:

- *application-id* is the ID of the Amazon Pinpoint project that you're importing endpoints for.

- *bucket-name/prefix/key* is the location in Amazon S3 that contains one or more objects to import. The location can end with the key for an individual object, or it can end with a prefix that qualifies multiple objects.
- *iam-import-role-arn* is the Amazon Resource Name (ARN) of an IAM role that grants Amazon Pinpoint read access to the bucket.
- *format* can be either JSON or CSV, depending on which format you used to define your endpoints. If the Amazon S3 location includes multiple files of mixed formats, Amazon Pinpoint imports only the files that match the specified format.

If your request succeeds, you receive a response like the following:

```
{
  "Id": "a995ce5d70fa44adb563b7d0e3f6c6f5",
  "JobStatus": "CREATED",
  "CreationDate": "2018-06-05T21:49:15.288Z",
  "Type": "IMPORT",
  "Definition": {
    "S3Url": "s3://bucket-name/prefix/key",
    "RoleArn": "iam-import-role-arn",
    "ExternalId": "external-id",
    "Format": "JSON",
    "RegisterEndpoints": true,
    "DefineSegment": false
  }
}
```

The response provides the job ID with the `Id` attribute. You can use this ID to check the current status of the import job.

Example GET Import Job Request

To check the current status of an import job, issue a GET request to the [Import Job](#) resource:

```
GET /v1/apps/application_id/jobs/import/job_id HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: pinpoint.us-east-1.amazonaws.com
X-Amz-Date: 20180605T220744Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20180605/
us-east-1/mobiletargeting/aws4_request, SignedHeaders=accept;cache-
control;content-type;host;postman-token;x-amz-date,
Signature=c25cbd6bf61bd3b3667c571ae764b9bf2d8af61b875caced95d1e68d91b4170
Cache-Control: no-cache
```

Where:

- *application_id* is the ID of the Amazon Pinpoint project for which the import job was initiated.
- *job_id* is the ID of the import job that you're checking.

If your request succeeds, you receive a response like the following:

```
{
  "ApplicationId": "application_id",
  "Id": "70a51b2cf442447492d2c8e50336a9e8",
  "JobStatus": "COMPLETED",
  "CompletedPieces": 1,
  "FailedPieces": 0,
  "TotalPieces": 1,
}
```

```
"CreationDate": "2018-06-05T22:04:49.213Z",
"CompletionDate": "2018-06-05T22:04:58.034Z",
"Type": "IMPORT",
"TotalFailures": 0,
"TotalProcessed": 3,
"Definition": {
  "S3Url": "s3://bucket-name/prefix/key.json",
  "RoleArn": "iam-import-role-arn",
  "ExternalId": "external-id",
  "Format": "JSON",
  "RegisterEndpoints": true,
  "DefineSegment": false
}
}
```

The response provides the job status with the `JobStatus` attribute.

Related Information

For more information about the Import Jobs resource in the Amazon Pinpoint API, including the supported HTTP methods and request parameters, see [Import Jobs](#) in the *Amazon Pinpoint API Reference*.

Deleting Endpoints from Amazon Pinpoint

You can delete endpoints when you no longer want to message a certain destination—such as when the destination becomes unreachable, or when a customer closes an account.

Examples

The following examples show you how to delete an endpoint.

AWS CLI

You can use Amazon Pinpoint by running commands with the AWS CLI.

Example Delete Endpoint Command

To delete an endpoint, use the `delete-endpoint` command:

```
$ aws pinpoint delete-endpoint \
> --application-id application-id \
> --endpoint-id endpoint-id
```

Where:

- *application-id* is the ID of the Amazon Pinpoint project that contains the endpoint.
- *endpoint-id* is the ID of the endpoint that you're deleting.

The response to this command is the JSON definition of the endpoint that you deleted.

AWS SDK for Java

You can use the Amazon Pinpoint API in your Java applications by using the client that's provided by the AWS SDK for Java.

Example Code

To delete an endpoint, use the `deleteEndpoint` method of the AmazonPinpoint client. Provide a `DeleteEndpointRequest` object as the method argument:

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.DeleteEndpointRequest;
import com.amazonaws.services.pinpoint.model.DeleteEndpointResult;

import java.util.Arrays;

public class DeleteEndpoints {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "DeleteEndpoints - Removes one or more endpoints from an " +
            "Amazon Pinpoint application.\n\n" +

            "Usage: DeleteEndpoints <applicationId> <endpointId1>
[endpointId2 ...]\n";

        if (args.length < 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String applicationId = args[0];
        String[] endpointIds = Arrays.copyOfRange(args, 1, args.length);

        // Initializes the Amazon Pinpoint client.
        AmazonPinpoint pinpointClient = AmazonPinpointClientBuilder.standard()
            .withRegion(Regions.US_EAST_1).build();

        try {
            // Deletes each of the specified endpoints with the Amazon Pinpoint client.
            for (String endpointId: endpointIds) {
                DeleteEndpointResult result =
                    pinpointClient.deleteEndpoint(new DeleteEndpointRequest()
                        .withEndpointId(endpointId)
                        .withApplicationId(applicationId));
                System.out.format("Deleted endpoint %s.\n",
                    result.getEndpointResponse().getId());
            }
        } catch (AmazonServiceException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

HTTP

You can use Amazon Pinpoint by making HTTP requests directly to the REST API.

Example DELETE Endpoint Request

To delete an endpoint, issue a DELETE request to the [Endpoint](#) resource:

```
DELETE /v1/apps/application-id/endpoints/endpoint-id HTTP/1.1
```

```
Host: pinpoint.us-east-1.amazonaws.com  
Content-Type: application/json  
Accept: application/json  
Cache-Control: no-cache
```

Where:

- *application-id* is the ID of the Amazon Pinpoint project that contains the endpoint.
- *endpoint-id* is the ID of the endpoint that you're deleting.

The response to this request is the JSON definition of the endpoint that you deleted.

Accessing Audience Data in Amazon Pinpoint

As you add endpoints to Amazon Pinpoint, it grows as a repository of audience data. This data consists of:

- The endpoints that you add or update by using the Amazon Pinpoint API.
- The endpoints that your client code adds or updates as users come to your application.

As your audience grows and changes, so does your endpoint data. To view the latest information that Amazon Pinpoint has about your audience, you can look up endpoints individually, or you can export all of the endpoints for an Amazon Pinpoint project. By viewing your endpoint data, you can learn about the audience characteristics that you record in your endpoints, such as:

- Your users' devices and platforms.
- Your users' time zones.
- The versions of your app that are installed on users' devices.
- Your users' locations, such as their cities or countries.
- Any custom attributes or metrics that you record.

The Amazon Pinpoint console also provides analytics for the demographics and custom attributes that are captured in your endpoints.

Before you can look up endpoints, you must add them to your Amazon Pinpoint project. To add endpoints, see [Defining Your Audience to Amazon Pinpoint \(p. 10\)](#).

Use the topics in this section to look up or export endpoints by using the Amazon Pinpoint API.

Topics

- [Looking Up Endpoints with Amazon Pinpoint \(p. 35\)](#)
- [Exporting Endpoints from Amazon Pinpoint \(p. 39\)](#)
- [Listing Endpoint IDs with Amazon Pinpoint \(p. 46\)](#)

Looking Up Endpoints with Amazon Pinpoint

You can look up the details for any individual endpoint that was added to an Amazon Pinpoint project. These details can include the destination address for your messages, the messaging channel, data about the user's device, data about the user's location, and any custom attributes that you record in your endpoints.

To look up an endpoint, you need the endpoint ID. If you don't know the ID, you can get the endpoint data by exporting instead. To export endpoints, see [the section called "Exporting Endpoints" \(p. 39\)](#).

Examples

The following examples show you how to look up an individual endpoint by specifying its ID.

AWS CLI

You can use Amazon Pinpoint by running commands with the AWS CLI.

Example Get Endpoint Command

To look up an endpoint, use the `get-endpoint` command:

```
$ aws pinpoint get-endpoint \  
> --application-id application-id \  
> --endpoint-id endpoint-id
```

Where:

- *application-id* is the ID of the Amazon Pinpoint project that contains the endpoint.
- *endpoint-id* is the ID of the endpoint that you're looking up.

The response to this command is the JSON definition of the endpoint, as in the following example:

```
{  
  "EndpointResponse": {  
    "Address": "1a2b3c4d5e6f7g8h9i0j1k2l3m4n5o6p7q8r9s0t1u2v3w4x5y6z7a8b9c0d1e2f",  
    "ApplicationId": "application-id",  
    "Attributes": {  
      "interests": [  
        "technology",  
        "music",  
        "travel"  
      ]  
    },  
    "ChannelType": "APNS",  
    "CohortId": "63",  
    "CreationDate": "2018-05-01T17:31:01.046Z",  
    "Demographic": {  
      "AppVersion": "1.0",  
      "Make": "apple",  
      "Model": "iPhone",  
      "ModelVersion": "8",  
      "Platform": "ios",  
      "PlatformVersion": "11.3.1",  
      "Timezone": "America/Los_Angeles"  
    },  
    "EffectiveDate": "2018-05-07T19:03:29.963Z",  
    "EndpointStatus": "ACTIVE",  
    "Id": "example_endpoint",  
    "Location": {  
      "City": "Seattle",  
      "Country": "US",  
      "Latitude": 47.6,  
      "Longitude": -122.3,  
      "PostalCode": "98121"  
    },  
    "Metrics": {  
      "music_interest_level": 6.0,  
      "travel_interest_level": 4.0,  
      "technology_interest_level": 9.0  
    },  
    "OptOut": "ALL",  
    "RequestId": "7f546cac-6858-11e8-adcd-2b5a07aab338",  
    "User": {  
      "UserAttributes": {
```

```
        "gender": [
            "female"
        ],
        "name": [
            "Wang",
            "Xiulan"
        ],
        "age": [
            "39"
        ]
    },
    "UserId": "example_user"
}
}
```

AWS SDK for Java

You can use the Amazon Pinpoint API in your Java applications by using the client that's provided by the AWS SDK for Java.

Example Code

To look up an endpoint, initialize a [GetEndpointRequest](#) object. Then, pass this object to the [getEndpoint](#) method of the AmazonPinpoint client:

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.EndpointResponse;
import com.amazonaws.services.pinpoint.model.GetEndpointRequest;
import com.amazonaws.services.pinpoint.model.GetEndpointResult;
import com.google.gson.FieldNamingPolicy;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class LookUpEndpoint {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "LookUpEndpoint - Prints the definition of the endpoint that has the\n" +
            "specified ID." +
            "Usage: LookUpEndpoint <applicationId> <endpointId>\n\n" +
            "Where:\n" +
            "  applicationId - The ID of the Amazon Pinpoint application that has\n" +
            "the " +
            "endpoint." +
            "  endpointId - The ID of the endpoint ";

        if (args.length < 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String applicationId = args[0];
        String endpointId = args[1];

        // Specifies the endpoint that the Amazon Pinpoint client looks up.
        GetEndpointRequest request = new GetEndpointRequest()
            .withEndpointId(endpointId)
```

```
        .withApplicationId(applicationId);

        // Initializes the Amazon Pinpoint client.
        AmazonPinpoint pinpointClient = AmazonPinpointClientBuilder.standard()
            .withRegion(Regions.US_EAST_1).build();

        // Uses the Amazon Pinpoint client to get the endpoint definition.
        GetEndpointResult result = pinpointClient.getEndpoint(request);
        EndpointResponse endpoint = result.getEndpointResponse();

        // Uses the Google Gson library to pretty print the endpoint JSON.
        Gson gson = new GsonBuilder()
            .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
            .setPrettyPrinting()
            .create();
        String endpointJson = gson.toJson(endpoint);

        System.out.println(endpointJson);
    }
}
```

To print the endpoint data in a readable format, this example uses the Google GSON library to convert the `EndpointResponse` object into a JSON string.

HTTP

You can use Amazon Pinpoint by making HTTP requests directly to the REST API.

Example GET Endpoint Request

To look up an endpoint, issue a GET request to the [Endpoint](#) resource:

```
GET /v1/apps/application_id/endpoints/endpoint_id HTTP/1.1
Host: pinpoint.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/json
Cache-Control: no-cache
```

Where:

- *application-id* is the ID of the Amazon Pinpoint project that contains the endpoint.
- *endpoint-id* is the ID of the endpoint that you're looking up.

The response to this request is the JSON definition of the endpoint, as in the following example:

```
{
  "ChannelType": "APNS",
  "Address": "1a2b3c4d5e6f7g8h9i0j1k2l3m4n5o6p7q8r9s0t1u2v3w4x5y6z7a8b9c0d1e2f",
  "EndpointStatus": "ACTIVE",
  "OptOut": "NONE",
  "RequestId": "b720cfa8-6924-11e8-aeda-0b22e0b0fa59",
  "Location": {
    "Latitude": 47.6,
    "Longitude": -122.3,
    "PostalCode": "98121",
    "City": "Seattle",
    "Country": "US"
  },
  "Demographic": {
```

```
    "Make": "apple",
    "Model": "iPhone",
    "ModelVersion": "8",
    "Timezone": "America/Los_Angeles",
    "AppVersion": "1.0",
    "Platform": "ios",
    "PlatformVersion": "11.3.1"
  },
  "EffectiveDate": "2018-06-06T00:58:19.865Z",
  "Attributes": {
    "interests": [
      "technology",
      "music",
      "travel"
    ]
  },
  "Metrics": {
    "music_interest_level": 6,
    "travel_interest_level": 4,
    "technology_interest_level": 9
  },
  "User": {},
  "ApplicationId": "application_id",
  "Id": "example_endpoint",
  "CohortId": "39",
  "CreationDate": "2018-06-06T00:58:19.865Z"
}
```

Related Information

For more information about the Endpoint resource in the Amazon Pinpoint API, see [Endpoint](#) in the *Amazon Pinpoint API Reference*.

Exporting Endpoints from Amazon Pinpoint

To get all of the information that Amazon Pinpoint has about your audience, you can export the endpoint definitions that belong to a project. When you export, Amazon Pinpoint places the endpoint definitions in an Amazon S3 bucket that you specify. Exporting endpoints is useful when you want to:

- View the latest data about new and existing endpoints that your client application registered with Amazon Pinpoint.
- Synchronize the endpoint data in Amazon Pinpoint with your own Customer Relationship Management (CRM) system.
- Create reports about or analyze your customer data.

Before You Begin

Before you can export endpoints, you need the following resources in your AWS account:

- An Amazon S3 bucket. To create a bucket, see [Create a Bucket](#) in the *Amazon Simple Storage Service Getting Started Guide*.
- An AWS Identity and Access Management (IAM) role that grants Amazon Pinpoint write permissions for your Amazon S3 bucket. To create the role, see [IAM Role for Exporting Endpoints or Segments](#) (p. 131).

Examples

The following examples demonstrate how to export endpoints from an Amazon Pinpoint project, and then download those endpoints from your Amazon S3 bucket.

AWS CLI

You can use Amazon Pinpoint by running commands with the AWS CLI.

Example Create Export Job Command

To export the endpoints in your Amazon Pinpoint project, use the `create-export-job` command:

```
$ aws pinpoint create-export-job \  
> --application-id application-id \  
> --export-job-request \  
> S3UrlPrefix=s3://bucket-name/prefix/,\  
> RoleArn=iam-export-role-arn
```

Where:

- `application-id` is the ID of the Amazon Pinpoint project that contains the endpoints.
- `bucket-name/prefix/` is the name of your Amazon S3 bucket and, optionally, a prefix that helps you organize the objects in your bucket hierarchically. For example, a useful prefix might be `pinpoint/exports/endpoints/`.
- `iam-export-role-arn` is the Amazon Resource Name (ARN) of an IAM role that grants Amazon Pinpoint write access to the bucket.

The response to this command provides details about the export job:

```
{  
  "ExportJobResponse": {  
    "CreationDate": "2018-06-04T22:04:20.585Z",  
    "Definition": {  
      "RoleArn": "iam-export-role-arn",  
      "S3UrlPrefix": "s3://s3-bucket-name/prefix/"  
    },  
    "Id": "7390e0de8e0b462380603c5a4df90bc4",  
    "JobStatus": "CREATED",  
    "Type": "EXPORT"  
  }  
}
```

The response provides the job ID with the `Id` attribute. You can use this ID to check the current status of the export job.

Example Get Export Job Command

To check the current status of an export job, use the `get-export-job` command:

```
$ aws pinpoint get-export-job \  
> --application-id application-id \  
> --job-id job-id
```

Where:

- `application-id` is the ID the Amazon Pinpoint project that you exported the endpoints from.

- `job-id` is the ID of the job that you're checking.

The response to this command provides the current state of the export job:

```
{
  "ExportJobResponse": {
    "ApplicationId": "application-id",
    "CompletedPieces": 1,
    "CompletionDate": "2018-05-08T22:16:48.228Z",
    "CreationDate": "2018-05-08T22:16:44.812Z",
    "Definition": {},
    "FailedPieces": 0,
    "Id": "6c99c463f14f49caa87fa27a5798bef9",
    "JobStatus": "COMPLETED",
    "TotalFailures": 0,
    "TotalPieces": 1,
    "TotalProcessed": 215,
    "Type": "EXPORT"
  }
}
```

The response provides the job status with the `JobStatus` attribute. When the job status value is `COMPLETED`, you can get your exported endpoints from your Amazon S3 bucket.

Example S3 CP Command

To download your exported endpoints, use the Amazon S3 `cp` command:

```
$ aws s3 cp s3://bucket-name/prefix/key.gz /local/directory/
```

Where:

- `bucket-name/prefix/key` is the location of the `.gz` file that Amazon Pinpoint added to your bucket when you exported your endpoints. This file contains the exported endpoint definitions.
- `/local/directory/` is the file path to the local directory that you want to download the endpoints to.

AWS SDK for Java

You can use the Amazon Pinpoint API in your Java applications by using the client that's provided by the AWS SDK for Java.

Example Code

To export endpoints from an Amazon Pinpoint project, initialize a `CreateExportJobRequest` object. Then, pass this object to the `createExportJob` method of the `AmazonPinpoint` client.

To download the exported endpoints from Amazon Pinpoint, use the `getObject` method of the `AmazonS3` client.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.*;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
```

```
import com.amazonaws.services.s3.model.S3Object;
import com.amazonaws.services.s3.model.S3ObjectInputStream;
import com.amazonaws.services.s3.model.S3ObjectSummary;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.Date;
import java.util.List;
import java.util.concurrent.TimeUnit;
import java.util.stream.Collectors;

public class ExportEndpoints {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "ExportEndpoints - Downloads endpoints from an Amazon Pinpoint
application by: \n" +
            "1.) Exporting the endpoint definitions to an Amazon S3 bucket. \n" +
            "2.) Downloading the endpoint definitions to the specified file path.\n
\n" +

            "Usage: ExportEndpoints <s3BucketName> <iamExportRoleArn>
<downloadDirectory> " +
            "<applicationId>\n\n" +

            "Where:\n" +
            " s3BucketName - The name of the Amazon S3 bucket to export the
endpoints files " +
            "to. If the bucket doesn't exist, a new bucket is created.\n" +
            " iamExportRoleArn - The ARN of an IAM role that grants Amazon
Pinpoint write " +
            "permissions to the S3 bucket.\n" +
            " downloadDirectory - The directory to download the endpoints files
to.\n" +
            " applicationId - The ID of the Amazon Pinpoint application that has
the " +
            "endpoints.";

        if (args.length < 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String s3BucketName = args[0];
        String iamExportRoleArn = args[1];
        String downloadDirectory = args[2];
        String applicationId = args[3];

        // Exports the endpoints to Amazon S3 and stores the keys of the new objects.
        List<String> objectKeys =
            exportEndpointsToS3(s3BucketName, iamExportRoleArn, applicationId);

        // Filters the keys to only those objects that have the endpoint definitions.
        // These objects have the .gz extension.
        List<String> endpointFileKeys = objectKeys
            .stream()
            .filter(o -> o.endsWith(".gz"))
            .collect(Collectors.toList());

        // Downloads the exported endpoints files to the specified directory.
        downloadFromS3(s3BucketName, endpointFileKeys, downloadDirectory);
    }
}
```

```
public static List<String> exportEndpointsToS3(String s3BucketName, String
iamExportRoleArn,
                                           String applicationId) {

    // The S3 path that Amazon Pinpoint exports the endpoints to.
    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd-
HH_mm:ss.SSS_z");
    String endpointsKeyPrefix = "exports/" + applicationId + "_" +
dateFormat.format(new Date
    ());
    String s3UrlPrefix = "s3://" + s3BucketName + "/" + endpointsKeyPrefix + "/";

    // Defines the export job that Amazon Pinpoint runs.
    ExportJobRequest exportJobRequest = new ExportJobRequest()
        .withS3UrlPrefix(s3UrlPrefix)
        .withRoleArn(iamExportRoleArn);
    CreateExportJobRequest createExportJobRequest = new CreateExportJobRequest()
        .withApplicationId(applicationId)
        .withExportJobRequest(exportJobRequest);

    // Initializes the Amazon Pinpoint client.
    AmazonPinpoint pinpointClient = AmazonPinpointClientBuilder.standard()
        .withRegion(Regions.US_EAST_1).build();

    System.out.format("Exporting endpoints from Amazon Pinpoint application %s to
Amazon S3 " +
        "bucket %s . . .\n", applicationId, s3BucketName);

    List<String> objectKeys = null;

    try {
        // Runs the export job with Amazon Pinpoint.
        CreateExportJobResult exportResult =
            pinpointClient.createExportJob(createExportJobRequest);

        // Prints the export job status to the console while the job runs.
        String jobId = exportResult.getExportJobResponse().getId();
        printExportJobStatus(pinpointClient, applicationId, jobId);

        // Initializes the Amazon S3 client.
        AmazonS3 s3Client = AmazonS3ClientBuilder.defaultClient();

        // Lists the objects created by Amazon Pinpoint.
        objectKeys = s3Client
            .listObjectsV2(s3BucketName, endpointsKeyPrefix)
            .getObjectSummaries()
            .stream()
            .map(S3ObjectSummary::getKey)
            .collect(Collectors.toList());

    } catch (AmazonServiceException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    return objectKeys;
}

private static void printExportJobStatus(AmazonPinpoint pinpointClient,
                                         String applicationId, String jobId) {

    GetExportJobResult getExportJobResult;
    String jobStatus;

    try {
```



```
// Checks the job status until the job completes or fails.
do {
    getExportJobResult = pinpointClient.getExportJob(new
GetExportJobRequest()
        .withJobId(jobId)
        .withApplicationId(applicationId));
    jobStatus = getExportJobResult.getExportJobResponse().getJobStatus();
    System.out.format("Export job %s . . .\n", jobStatus.toLowerCase());
    TimeUnit.SECONDS.sleep(3);
} while (!jobStatus.equals("COMPLETED") && !jobStatus.equals("FAILED"));

if (jobStatus.equals("COMPLETED")) {
    System.out.println("Finished exporting endpoints.");
} else {
    System.err.println("Failed to export endpoints.");
    System.exit(1);
}

// Checks for entries that failed to import.
// getFailures provides up to 100 of the first failed entries for the job,
if any exist.
List<String> failedEndpoints =
getExportJobResult.getExportJobResponse().getFailures();
if (failedEndpoints != null) {
    System.out.println("Failed to import the following entries:");
    for (String failedEndpoint : failedEndpoints) {
        System.out.println(failedEndpoint);
    }
}
} catch (AmazonServiceException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void downloadFromS3(String s3BucketName, List<String> objectKeys,
String downloadDirectory) {

    // Initializes the Amazon S3 client.
    AmazonS3 s3Client = AmazonS3ClientBuilder.defaultClient();

    try {
        // Downloads each object to the specified file path.
        for (String key : objectKeys) {
            S3Object object = s3Client.getObject(s3BucketName, key);
            String endpointsFileName = key.substring(key.lastIndexOf("/"));
            Path filePath = Paths.get(downloadDirectory + endpointsFileName);

            System.out.format("Downloading %s to %s . . .\n",
                filePath.getFileName(), filePath.getParent());

            writeObjectToFile(filePath, object);
        }
        System.out.println("Download finished.");
    } catch (AmazonServiceException | NullPointerException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

private static void writeObjectToFile(Path filePath, S3Object object) {

    // Writes the contents of the S3 object to a file.
    File endpointsFile = new File(filePath.toAbsolutePath().toString());
    try (FileOutputStream fos = new FileOutputStream(endpointsFile);
```

```
        S3ObjectInputStream s3is = object.getObjectContent() {
        byte[] read_buf = new byte[1024];
        int read_len = 0;
        while ((read_len = s3is.read(read_buf)) > 0) {
            fos.write(read_buf, 0, read_len);
        }
    } catch (IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

HTTP

You can use Amazon Pinpoint by making HTTP requests directly to the REST API.

Example POST Export Job Request

To export the endpoints in your Amazon Pinpoint project, issue a `POST` request to the [Export Jobs](#) resource:

```
POST /v1/apps/application_id/jobs/export HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: pinpoint.us-east-1.amazonaws.com
X-Amz-Date: 20180606T001238Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20180606/
us-east-1/mobiletargeting/aws4_request, SignedHeaders=accept;cache-
control;content-length;content-type;host;postman-token;x-amz-date,
Signature=c25cbd6bf61bd3b3667c571ae764b9bf2d8af61b875caced95d1e68d91b4170
Cache-Control: no-cache

{
  "S3UrlPrefix": "s3://bucket-name/prefix",
  "RoleArn": "iam-export-role-arn"
}
```

Where:

- *application-id* is the ID of the Amazon Pinpoint project that contains the endpoints.
- *bucket-name/prefix/* is the name of your Amazon S3 bucket and, optionally, a prefix that helps you organize the objects in your bucket hierarchically. For example, a useful prefix might be `pinpoint/exports/endpoints/`.
- *iam-export-role-arn* is the Amazon Resource Name (ARN) of an IAM role that grants Amazon Pinpoint write access to the bucket.

The response to this request provides details about the export job:

```
{
  "Id": "611bdc54c75244bfa51fe7001ddb2e36",
  "JobStatus": "CREATED",
  "CreationDate": "2018-06-06T00:12:43.271Z",
  "Type": "EXPORT",
  "Definition": {
    "S3UrlPrefix": "s3://bucket-name/prefix",
    "RoleArn": "iam-export-role-arn"
  }
}
```

The response provides the job ID with the `Id` attribute. You can use this ID to check the current status of the export job.

Example GET Export Job Request

To check the current status of an export job, issue a GET request to the [Export Job](#) resource:

```
GET /v1/apps/application_id/jobs/export/job_id HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: pinpoint.us-east-1.amazonaws.com
X-Amz-Date: 20180606T002443Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20180606/
us-east-1/mobiletargeting/aws4_request, SignedHeaders=accept;cache-
control;content-type;host;postman-token;x-amz-date,
Signature=c25cbd6bf61bd3b3667c571ae764b9bf2d8af61b875caced95d1e68d91b4170
Cache-Control: no-cache
```

Where:

- *application-id* is the ID the Amazon Pinpoint project that you exported the endpoints from.
- *job-id* is the ID of the job that you're checking.

The response to this request provides the current state of the export job:

```
{
  "ApplicationId": "application_id",
  "Id": "job_id",
  "JobStatus": "COMPLETED",
  "CompletedPieces": 1,
  "FailedPieces": 0,
  "TotalPieces": 1,
  "CreationDate": "2018-06-06T00:12:43.271Z",
  "CompletionDate": "2018-06-06T00:13:01.141Z",
  "Type": "EXPORT",
  "TotalFailures": 0,
  "TotalProcessed": 217,
  "Definition": {}
}
```

The response provides the job status with the `JobStatus` attribute. When the job status value is `COMPLETED`, you can get your exported endpoints from your Amazon S3 bucket.

Related Information

For more information about the Export Jobs resource in the Amazon Pinpoint API, including the supported HTTP methods and request parameters, see [Export Jobs](#) in the *Amazon Pinpoint API Reference*.

Listing Endpoint IDs with Amazon Pinpoint

To update or delete an endpoint, you need the endpoint ID. So, if you want to perform these operations on all of the endpoints in an Amazon Pinpoint project, the first step is to list all of the endpoint IDs that belong to that project. Then, you can iterate over these IDs to, for example, add an attribute globally or delete all of the endpoints in your project.

The following example uses the AWS SDK for Java and does the following:

1. Calls the example `exportEndpointsToS3` method from the example code in [Exporting Endpoints from Amazon Pinpoint \(p. 39\)](#). This method exports the endpoint definitions from an Amazon Pinpoint project. The endpoint definitions are added as gzip files to an Amazon S3 bucket.
2. Downloads the exported gzip files.
3. Reads the gzip files and obtains the endpoint ID from each endpoint's JSON definition.
4. Prints the endpoint IDs to the console.
5. Cleans up by deleting the files that Amazon Pinpoint added to Amazon S3.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.DeleteObjectsRequest;
import com.amazonaws.services.s3.model.S3Object;
import com.google.gson.FieldNamingPolicy;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.JsonObject;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;
import java.util.zip.GZIPInputStream;

public class ListEndpointIds {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "ListEndpointIds - Prints all of the endpoint IDs that belong to an Amazon\n" +
            "Pinpoint application. This program performs the following steps:\n\n" +
            "1) Exports the endpoints to an Amazon S3 bucket.\n" +
            "2) Downloads the exported endpoints files from Amazon S3.\n" +
            "3) Parses the endpoints files to obtain the endpoint IDs and prints them.\n" +
            "4) Cleans up by deleting the objects that Amazon Pinpoint created in the\n" +
            "S3 bucket.\n\n" +
            "Usage: ListEndpointIds <applicationId> <s3BucketName> <iamExportRoleArn>\n\n" +
            "Where:\n" +
            "  applicationId - The ID of the Amazon Pinpoint application that has the\n" +
            "  endpoint.\n" +
            "  s3BucketName - The name of the Amazon S3 bucket to export the JSON file\n" +
            "to. If the bucket doesn't exist, a new bucket is created.\n" +
            "  iamExportRoleArn - The ARN of an IAM role that grants Amazon Pinpoint\n" +
            "write permissions to the S3 bucket.";

        if (args.length < 1) {
            System.out.println(USAGE);
            System.exit(1);
        }
    }
}
```

```
String applicationId = args[0];
String s3BucketName = args[1];
String iamExportRoleArn = args[2];

// Exports the endpoints to Amazon S3 and stores the keys of the new objects.
List<String> objectKeys =
    ExportEndpoints.exportEndpointsToS3(s3BucketName, iamExportRoleArn,
applicationId);

// Filters the keys to only those objects that have the endpoint definitions.
// These objects have the .gz extension.
List<String> endpointFileKeys = objectKeys
    .stream()
    .filter(o -> o.endsWith(".gz"))
    .collect(Collectors.toList());

// Gets the endpoint IDs from the exported endpoints files.
List<String> endpointIds = getEndpointIds(s3BucketName, endpointFileKeys);

System.out.println("Endpoint IDs:");

for (String endpointId : endpointIds) {
    System.out.println("\t- " + endpointId);
}

// Deletes the objects that Amazon Pinpoint created in the S3 bucket.
deleteS3Objects(s3BucketName, objectKeys);
}

private static List<String> getEndpointIds(String s3bucketName, List<String>
endpointFileKeys) {

    List<String> endpointIds = new ArrayList<>();

    // Initializes the Amazon S3 client.
    AmazonS3 s3Client = AmazonS3ClientBuilder.defaultClient();

    // Gets the endpoint IDs from the exported endpoints files.
    try {
        for (String key : endpointFileKeys) {
            S3Object endpointFile = s3Client.getObject(s3bucketName, key);
            endpointIds.addAll(getEndpointIdsFromFile(endpointFile));
        }
    } catch (AmazonServiceException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    return endpointIds;
}

private static List<String> getEndpointIdsFromFile(S3Object endpointsFile) {

    List<String> endpointIdsFromFile = new ArrayList<>();

    // The Google Gson library is used to parse the exported endpoint JSON.
    Gson gson = new GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .create();

    // Reads each endpoint entry in the file and adds the ID to the list.
    try (GZIPInputStream gzipInputStream =
        new GZIPInputStream(endpointsFile.getObjectContent());
        BufferedReader reader =
        new BufferedReader(new InputStreamReader(
```

```
        gzipInputStream, "UTF-8"))) {
    String endpointString;
    while ((endpointString = reader.readLine()) != null) {
        JsonObject endpointJson = gson.fromJson(endpointString, JsonObject.class);
        endpointIdsFromFile.add(endpointJson
            .getAsJsonPrimitive("Id")
            .getAsString());
    }
} catch (IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

return endpointIdsFromFile;
}

private static void deleteS3Objects(String s3BucketName, List<String> keys) {

    AmazonS3 s3Client = AmazonS3ClientBuilder.defaultClient();

    String[] keysArray = keys.toArray(new String[keys.size()]);
    DeleteObjectsRequest request = new
DeleteObjectsRequest(s3BucketName).withKeys(keysArray);

    System.out.println("Deleting the following Amazon S3 objects created by Amazon
Pinpoint:");

    for (String key : keys) {
        System.out.println("\t- " + key);
    }

    try {
        s3Client.deleteObjects(request);
    } catch (AmazonServiceException e) {
        System.err.println(e.getErrorMessage());
        System.exit(1);
    }

    System.out.println("Finished deleting objects.");
}
}
```

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 \(p. 50\)](#)
- [Importing Segments \(p. 51\)](#)
- [Customizing Segments with AWS Lambda \(p. 54\)](#)

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 endpoints that belong to the segment. An *endpoint* is a single messaging destination, such as a mobile push device token, a mobile phone number, or an email address.

Importing segments is useful if you've already created segments of 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 Endpoints or Segments \(p. 130\)](#). For information about importing a segment by using the Amazon Pinpoint console, see [Importing Segments](#) in the *Amazon Pinpoint User Guide*.

Importing a Segment

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

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



```
import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.*;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import java.io.File;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;
import java.util.concurrent.TimeUnit;

public class ImportSegment {

    public static void main(String[] args) {

        final String USAGE = "\n" +
            "ImportSegment - Creates a segment by: \n" +
            "1.) Uploading the endpoint definitions that belong to the segment to an Amazon S3 bucket. \n" +
            "2.) Importing the endpoint definitions from the bucket to an Amazon Pinpoint application." +
            "    Amazon Pinpoint creates a segment that has the specified name.\n\n" +
            "Usage: ImportSegment <endpointsFileLocation> <s3BucketName> <iamImportRoleArn> <segmentName> <applicationId>\n\n" +
            "Where:\n" +
            "    endpointsFileLocation - The relative location of the JSON file that contains the endpoint definitions.\n" +
            "    s3BucketName - The name of the Amazon S3 bucket to upload the JSON file to. If the bucket doesn't " +
            "exist, a new bucket is created.\n" +
            "    iamImportRoleArn - The ARN of an IAM role that grants Amazon Pinpoint read permissions so the S3 bucket.\n" +
            "    segmentName - The name for the segment that you are creating or updating." +
            "    applicationId - The ID of the Amazon Pinpoint application to add the endpoints to.";

        if (args.length < 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String endpointsFileLocation = args[0];
        String s3BucketName = args[1];
        String iamImportRoleArn = args[2];
        String segmentName = args[3];
        String applicationId = args[4];

        Path endpointsFilePath = Paths.get(endpointsFileLocation);
        File endpointsFile = new File(endpointsFilePath.toAbsolutePath().toString());
        uploadToS3(endpointsFile, s3BucketName);

        importSegment(endpointsFile.getName(), s3BucketName, iamImportRoleArn, segmentName, applicationId);
    }

    private static void uploadToS3(File endpointsFile, String s3BucketName) {

        // Initializes Amazon S3 client.
        final AmazonS3 s3 = AmazonS3ClientBuilder.defaultClient();

        // Checks whether the specified bucket exists. If not, attempts to create one.
        if (!s3.doesBucketExistV2(s3BucketName)) {
            try {
                s3.createBucket(s3BucketName);
            }
        }
    }
}
```

```
        System.out.format("Created S3 bucket %s.\n", s3BucketName);
    } catch (AmazonS3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Uploads the endpoints file to the bucket.
String endpointsFileName = endpointsFile.getName();
System.out.format("Uploading %s to S3 bucket %s . . .\n", endpointsFileName,
s3BucketName);
try {
    s3.putObject(s3BucketName, "imports/" + endpointsFileName, endpointsFile);
    System.out.println("Finished uploading to S3.");
} catch (AmazonServiceException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

private static void importSegment(String endpointsFileName, String s3BucketName, String
iamImportRoleArn,
                                String segmentName, String applicationId) {

    // The S3 URL that Amazon Pinpoint requires to find the endpoints file.
    String s3Url = "s3://" + s3BucketName + "/imports/" + endpointsFileName;

    // Defines the import job that Amazon Pinpoint runs.
    ImportJobRequest importJobRequest = new ImportJobRequest()
        .withS3Url(s3Url)
        .withFormat(Format.JSON)
        .withRoleArn(iamImportRoleArn)
        .withRegisterEndpoints(true)
        .withDefineSegment(true)
        .withSegmentName(segmentName);
    CreateImportJobRequest createImportJobRequest = new CreateImportJobRequest()
        .withApplicationId(applicationId)
        .withImportJobRequest(importJobRequest);

    // Initializes the Amazon Pinpoint client.
    AmazonPinpoint pinpointClient = AmazonPinpointClientBuilder.standard()
        .withRegion(Regions.US_EAST_1).build();

    System.out.format("Creating segment %s with the endpoints in %s . . .\n",
segmentName, endpointsFileName);

    try {

        // Runs the import job with Amazon Pinpoint.
        CreateImportJobResult importResult =
pinpointClient.createImportJob(createImportJobRequest);
        String jobId = importResult.getImportJobResponse().getId();

        // Checks the job status until the job completes or fails.
        GetImportJobResult getImportJobResult = null;
        String jobStatus = null;
        do {
            getImportJobResult = pinpointClient.getImportJob(new GetImportJobRequest()
                .withJobId(jobId)
                .withApplicationId(applicationId));
            jobStatus = getImportJobResult.getImportJobResponse().getJobStatus();
            System.out.format("Import job %s . . .\n", jobStatus.toLowerCase());
            if (jobStatus.equals("FAILED")) {
                System.err.println("Failed to import segment.");
                System.exit(1);
            }
        }
    }
}
```

```
        try {
            TimeUnit.SECONDS.sleep(3);
        } catch (InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    } while (!jobStatus.equals("COMPLETED"));

    System.out.println("Finished importing segment.");

    // Checks for entries that failed to import.
    List<String> failedEndpoints =
getImportJobResult.getImportJobResponse().getFailures();
    if (failedEndpoints != null) {
        System.out.println("Failed to import the following entries:");
        for (String failedEndpoint : failedEndpoints) {
            System.out.println(failedEndpoint);
        }
    }

} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}

}
}
```

Customizing Segments with AWS Lambda

This is prerelease documentation for a feature in public beta release. It is subject to change.

You can use AWS Lambda to tailor how an Amazon Pinpoint campaign engages your target audience. With AWS Lambda, you can modify the campaign's segment at the moment that Amazon Pinpoint delivers the campaign's message.

AWS Lambda is a compute service that you can use to run code without provisioning or managing servers. You package your code and upload it to Lambda as *Lambda functions*. Lambda runs a function when the function is invoked, which might be done manually by you or automatically in response to events.

For more information, see [Lambda Functions](#) in the *AWS Lambda Developer Guide*.

To assign a Lambda function to a campaign, you define the campaign's [CampaignHook](#) settings. These settings include the Lambda function name. They also include the `CampaignHook` mode, which sets whether Amazon Pinpoint receives a return value from the function.

A Lambda function that you assign to a campaign is referred to as an Amazon Pinpoint *extension*.

With the `CampaignHook` settings defined, Amazon Pinpoint automatically invokes the Lambda function when it runs the campaign, before it delivers the campaign's message. When Amazon Pinpoint invokes the function, it provides *event data* about the message delivery. This data includes the campaign's segment, which is the list of endpoints that Amazon Pinpoint sends the message to.

If the `CampaignHook` mode is set to `FILTER`, Amazon Pinpoint allows the function to modify and return the segment before sending the message. For example, the function might update the endpoint

definitions with attributes that contain data from a source that is external to Amazon Pinpoint. Or, the function might filter the segment by removing certain endpoints, based on conditions in your function code. After Amazon Pinpoint receives the modified segment from your function, it sends the message to each of the segment's endpoints using the campaign's delivery channel.

By processing your segments with AWS Lambda, you have more control over who you send messages to and what those messages contain. You can tailor your campaigns in real time, at the moment campaign messages are delivered. Filtering segments enables you to engage more narrowly defined subsets of your segments. Adding or updating endpoint attributes enables you to make new data available for message variables.

For more information about message variables, see [Message Variables](#) in the *Amazon Pinpoint User Guide*.

Note

You can also use the CampaignHook settings to assign a Lambda function that handles the message delivery. This type of function is useful for delivering messages through custom channels that Amazon Pinpoint doesn't support, such as social media platforms. For more information, see [Creating Custom Channels with AWS Lambda](#) (p. 146).

To modify campaign segments with AWS Lambda, first create a function that processes the event data sent by Amazon Pinpoint and returns a modified segment. Then, authorize Amazon Pinpoint to invoke the function by assigning a Lambda function policy. Finally, assign the function to one or more campaigns by defining CampaignHook settings.

Event Data

When Amazon Pinpoint invokes your Lambda function, it provides the following payload as the event data:

```
{
  "MessageConfiguration": {Message configuration}
  "ApplicationId": ApplicationId,
  "CampaignId": CampaignId,
  "TreatmentId": TreatmentId,
  "ActivityId": ActivityId,
  "ScheduledTime": Scheduled Time,
  "Endpoints": {
    EndpointId: {Endpoint definition}
    . . .
  }
}
```

AWS Lambda passes the event data to your function code. The event data provides the following attributes:

- **MessageConfiguration** – Has the same structure as the [DirectMessageConfiguration](#) in the Messages resource in the Amazon Pinpoint API.
- **ApplicationId** – The ID of the Amazon Pinpoint project that the campaign belongs to.
- **CampaignId** – The ID of the Amazon Pinpoint project that the function is invoked for.
- **TreatmentId** – The ID of a campaign variation that's used for A/B testing.
- **ActivityId** – The ID of the activity that's being performed by the campaign.
- **ScheduledTime** – The date and time, in ISO 8601 format, when the campaign's messages will be delivered.
- **Endpoints** – A map that associates endpoint IDs with endpoint definitions. Each event data payload contains up to 50 endpoints. If the campaign segment contains more than 50 endpoints, Amazon Pinpoint invokes the function repeatedly, with up to 50 endpoints at a time, until all endpoints have been processed.

Creating a Lambda Function

To create a Lambda function, see [Building Lambda Functions](#) in the *AWS Lambda Developer Guide*.

When you create your function, remember that the message delivery fails in the following conditions:

- The Lambda function takes longer than 15 seconds to return the modified segment.
- Amazon Pinpoint can't decode the function's return value.
- The function requires more than 3 attempts from Amazon Pinpoint to successfully invoke.

Amazon Pinpoint only accepts endpoint definitions in the function's return value. The function can't modify other elements in the event data.

Example Lambda Function

Your Lambda function processes the event data sent by Amazon Pinpoint, and it returns the modified endpoints, as shown by the following example handler, written in Node.js:

```
'use strict';

exports.handler = (event, context, callback) => {
  for (var key in event.Endpoints) {
    if (event.Endpoints.hasOwnProperty(key)) {
      var endpoint = event.Endpoints[key];
      var attr = endpoint.Attributes;
      if (!attr) {
        attr = {};
        endpoint.Attributes = attr;
      }
      attr["CreditScore"] = [ Math.floor(Math.random() * 200) + 650];
    }
  }
  console.log("Received event:", JSON.stringify(event, null, 2));
  callback(null, event.Endpoints);
};
```

Lambda passes the event data to the handler as the `event` parameter.

In this example, the handler iterates through each endpoint in the `event.Endpoints` object, and it adds a new attribute, `CreditScore`, to the endpoint. The value of the `CreditScore` attribute is simply a random number.

The `console.log()` statement logs the event in CloudWatch Logs.

The `callback()` statement returns the modified endpoints to Amazon Pinpoint. Normally, the `callback` parameter is optional in Node.js Lambda functions, but it is required in this context because the function must return the updated endpoints to Amazon Pinpoint.

Your function must return endpoints in the same format provided by the event data, which is a map that associates endpoint IDs with endpoint definitions, as in the following example:

```
{
  "eqmj8wpxszeqy/b3vch04sn41yw": {
    "ChannelType": "GCM",
    "Address": "4d5e6f1a2b3c4d5e6f7g8h9i0j1a2b3c",
    "EndpointStatus": "ACTIVE",
    "OptOut": "NONE",
    "Demographic": {
```

```
        "Make": "android"
    },
    "EffectiveDate": "2017-11-02T21:26:48.598Z",
    "User": {}
  },
  "idrexqqt8sbwfx0ouscod0yto": {
    "ChannelType": "APNS",
    "Address": "1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f",
    "EndpointStatus": "ACTIVE",
    "OptOut": "NONE",
    "Demographic": {
      "Make": "apple"
    },
  },
  "EffectiveDate": "2017-11-02T21:26:48.598Z",
  "User": {}
}
}
```

The example function modifies and returns the `event.Endpoints` object that it received in the event data.

Optionally, you can include the `TitleOverride` and `BodyOverride` attributes in the endpoint definitions that you return.

Note

When you use this solution to send messages, Amazon Pinpoint only honors the `TitleOverride` and `BodyOverride` attributes for endpoints where the value of the `ChannelType` attribute is one of the following: SMS, GCM, APNS, APNS_SANDBOX, APNS_VOIP, APNS_VOIP_SANDBOX, ADM, or BAIDU. Amazon Pinpoint **doesn't** honor these attributes for endpoints where the value of the `ChannelType` attribute is EMAIL.

Assigning a Lambda Function Policy

Before you can use your Lambda function to process your endpoints, you must authorize Amazon Pinpoint to invoke your Lambda function. To grant invocation permission, assign a *Lambda function policy* to the function. A Lambda function policy is a resource-based permissions policy that designates which entities can use your function and what actions those entities can take.

For more information, see [Using Resource-Based Policies for AWS Lambda \(Lambda Function Policies\)](#) in the *AWS Lambda Developer Guide*.

Example Function Policy

The following policy grants permission to the Amazon Pinpoint service principal to use the `lambda:InvokeFunction` action:

```
{
  "Sid": "sid",
  "Effect": "Allow",
  "Principal": {
    "Service": "pinpoint.us-east-1.amazonaws.com"
  },
  "Action": "lambda:InvokeFunction",
  "Resource": "{arn:aws:lambda:us-east-1:account-id:function:function-name}",
  "Condition": {
    "ArnLike": {
      "AWS:SourceArn": "arn:aws:mobiletargeting:us-east-1:account-id:/apps/application-id/campaigns/campaign-id"
    }
  }
}
```

```
}  
}
```

Your function policy requires a `Condition` block that includes an `AWS:SourceArn` key. This code states which Amazon Pinpoint campaign is allowed to invoke the function. In this example, the policy grants permission to only a single campaign ID. To write a more generic policy, use multi-character match wildcards (*). For example, you can use the following `Condition` block to allow any Amazon Pinpoint campaign in your AWS account to invoke the function:

```
"Condition": {  
  "ArnLike": {  
    "AWS:SourceArn": "arn:aws:mobiletargeting:us-east-1:account-id:/apps/*"  
  }  
}
```

Granting Amazon Pinpoint Invocation Permission

You can use the AWS Command Line Interface (AWS CLI) to add permissions to the Lambda function policy assigned to your Lambda function. To allow Amazon Pinpoint to invoke a function, use the Lambda `add-permission` command, as shown by the following example:

```
$ aws lambda add-permission \  
> --function-name function-name \  
> --statement-id sid \  
> --action lambda:InvokeFunction \  
> --principal pinpoint.us-east-1.amazonaws.com \  
> --source-arn arn:aws:mobiletargeting:us-east-1:account-id:/apps/application-id/  
campaigns/campaign-id
```

If you want to provide a campaign ID for the `--source-arn` parameter, you can look up your campaign IDs by using the Amazon Pinpoint `get-campaigns` command with the AWS CLI. This command requires an `--application-id` parameter. To look up your application IDs, sign in to the Amazon Pinpoint console at <https://console.aws.amazon.com/pinpoint/>, and go to the **Projects** page. The console shows an **ID** for each project, which is the project's application ID.

When you run the Lambda `add-permission` command, AWS Lambda returns the following output:

```
{  
  "Statement": "{\"Sid\":\"sid\",  
    \"Effect\":\"Allow\",  
    \"Principal\":{\"Service\":\"pinpoint.us-east-1.amazonaws.com\"},  
    \"Action\":\"lambda:InvokeFunction\",  
    \"Resource\":\"arn:aws:lambda:us-east-1:111122223333:function:function-name\",  
    \"Condition\":  
      {\"ArnLike\":  
        {\"AWS:SourceArn\":  
          \"arn:aws:mobiletargeting:us-east-1:111122223333:/apps/application-id/campaigns/  
campaign-id\"}}}"  
}
```

The `Statement` value is a JSON string version of the statement added to the Lambda function policy.

Assigning a Lambda Function to a Campaign

You can assign a Lambda function to an individual Amazon Pinpoint campaign. Or, you can set the Lambda function as the default used by all campaigns for a project, except for those campaigns to which you assign a function individually.

To assign a Lambda function to an individual campaign, use the Amazon Pinpoint API to create or update a `Campaign` object, and define its `CampaignHook` attribute. To set a Lambda function as the default for all campaigns in a project, create or update the `Settings` resource for that project, and define its `CampaignHook` object.

In both cases, set the following `CampaignHook` attributes:

- `LambdaFunctionName` – The name or ARN of the Lambda function that Amazon Pinpoint invokes before sending messages for the campaign.
- `Mode` – Set to `FILTER`. With this mode, Amazon Pinpoint invokes the function and waits for it to return the modified endpoints. After receiving them, Amazon Pinpoint sends the message. Amazon Pinpoint waits for up to 15 seconds before failing the message delivery.

With `CampaignHook` settings defined for a campaign, Amazon Pinpoint invokes the specified Lambda function before sending the campaign's messages. Amazon Pinpoint waits to receive the modified endpoints from the function. If Amazon Pinpoint receives the updated endpoints, it proceeds with the message delivery, using the updated endpoint data.

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
```

Sending Transactional Messages from Your Apps

You can use the Amazon Pinpoint API and the AWS SDKs to send *transactional messages* directly from your apps. Transactional messages are messages that you send to specific recipients, as opposed to messages that you send to segments. There are several reasons that you might want to send transactional messages rather than campaign-based messages. For example, you can send an order confirmation by email when a customer places an order. You could also send a one-time password by SMS or voice that a customer can use to complete the process of creating an account for your service.

This section includes example code in several programming languages that you can use to start sending transactional emails, SMS messages, and voice messages.

Topics in this section:

- [Send Transactional Email Messages \(p. 63\)](#)
- [Send SMS Messages \(p. 88\)](#)
- [Send Voice Messages \(p. 94\)](#)

Send Transactional Email Messages

This section provides complete code samples that you can use to send transactional email messages through Amazon Pinpoint. There are three methods that you can use to send transactional email messages:

- [By using the `SendMessage` operation in the Amazon Pinpoint API \(p. 64\)](#): You can use the `SendMessage` operation in the Amazon Pinpoint API to send messages in all of the channels that Amazon Pinpoint supports, including the push notification, SMS, voice, and email channels.

The advantage of using this operation is that the request syntax for sending messages is very similar across all channels. This makes it easier to repurpose your existing code. The `SendMessage` operation also lets you to substitute content in your email messages, and lets you send email to Amazon Pinpoint endpoint IDs rather than to specific email addresses.

- [By using the `SendEmail` operation in the Amazon Pinpoint Email API \(p. 80\)](#): You can use the `SendEmail` operation in the Amazon Pinpoint Email API to send email messages only.

The advantage of using this operation is that the Amazon Pinpoint Email API includes additional options that are specific to the email channel. For example, when you send an email by using the Amazon Pinpoint Email API, you can specify CC and BCC recipients. You can also specify a configuration set. You can use configuration sets to specify which of your dedicated IP addresses your emails should be sent from, or to specify how Amazon Pinpoint handles open and click events.

- [By using the Amazon Pinpoint SMTP interface \(p. 80\)](#): You can use the Amazon Pinpoint SMTP interface to send email messages only.

The advantage to using the SMTP interface is that you can use email-sending applications and libraries to send email. For example, if you use a ticketing system that sends emails to your customers, you can use the SMTP interface to configure the system to send emails from your domain. You can also use email-sending libraries in several programming languages to send email through the SMTP interface.

This section includes example code in several programming languages that you can use to start sending transactional emails.

Topics in this section:

- [Choosing an Email-Sending Method \(p. 64\)](#)
- [Send Email by Using the Amazon Pinpoint API \(p. 64\)](#)
- [Send Email by Using the Amazon Pinpoint Email API \(p. 71\)](#)
- [Send Email by Using the Amazon Pinpoint SMTP Interface \(p. 80\)](#)

Choosing an Email-Sending Method

The best method to use for sending transactional email depends on your use case. For example, if you need to send email by using a third-party application, or if there isn't an AWS SDK available for your programming language, you might have to use the SMTP interface. If you want to send messages in other channels that Amazon Pinpoint supports, and you want to use consistent code for making those requests, you should use the `SendMessage` operation in the Amazon Pinpoint API. If you need to be able to send messages to CC and BCC recipients, or use dedicated IP pools, or specify message tags in your emails, you should use the `SendEmail` operation in the Amazon Pinpoint Email API.

The following decision tree shows some of the factors that you should consider when you decide which method you use to send transactional email in Amazon Pinpoint.

Send Email by Using the Amazon Pinpoint API

This section contains complete code examples that you can use to send email through the Amazon Pinpoint API by using an AWS SDK.

C#

Use this example to send email by using the [AWS SDK for .NET](#). This example assumes that you've already installed and configured the AWS SDK for .NET. For more information, see [Getting Started with the AWS SDK for .NET](#) in the *AWS SDK for .NET Developer Guide*.

This example assumes that you're using a shared credentials file to specify the Access Key and Secret Access Key for an existing IAM user. For more information, see [Configuring AWS Credentials](#) in the *AWS SDK for .NET Developer Guide*.

This code example was tested using the AWS SDK for .NET version 3.3.29.13 and .NET Core runtime version 2.1.2.

```
using System;
using System.Collections.Generic;
using Amazon;
using Amazon.Pinpoint;
using Amazon.Pinpoint.Model;

namespace PinpointEmailSendMessageAPI
{
    class MainClass
    {
        // The AWS Region that you want to use to send the email. For a list of
        // AWS Regions where the Amazon Pinpoint API is available, see
        // https://docs.aws.amazon.com/pinpoint/latest/apireference/
        static string region = "us-west-2";
    }
}
```

```
// The "From" address. This address has to be verified in Amazon Pinpoint
// in the region you're using to send email.
static string senderAddress = "sender@example.com";

// The address on the "To" line. If your Amazon Pinpoint account is in
// the sandbox, this address also has to be verified.
static string toAddress = "recipient@example.com";

// The Amazon Pinpoint project/application ID to use when you send this
message.
// Make sure that the SMS channel is enabled for the project or application
// that you choose.
static string appId = "ce796be37f32f178af652b26eexample";

// The subject line of the email.
static string subject = "Amazon Pinpoint Email test";

// The body of the email for recipients whose email clients don't
// support HTML content.
static string textBody = @"Amazon Pinpoint Email Test (.NET)
-----
This email was sent using the Amazon Pinpoint API using the AWS SDK for .NET.";

// The body of the email for recipients whose email clients support
// HTML content.
static string htmlBody = @"<html>
<head></head>
<body>
  <h1>Amazon Pinpoint Email Test (AWS SDK for .NET)</h1>
  <p>This email was sent using the
    <a href='https://aws.amazon.com/pinpoint/'>Amazon Pinpoint</a> API
    using the <a href='https://aws.amazon.com/sdk-for-net/'>
      AWS SDK for .NET</a>.</p>
</body>
</html>";

// The character encoding the you want to use for the subject line and
// message body of the email.
static string charset = "UTF-8";

public static void Main(string[] args)
{
    using (var client = new
AmazonPinpointClient(RegionEndpoint.GetBySystemName(region)))
    {
        var sendRequest = new SendMessagesRequest
        {
            ApplicationId = appId,
            MessageRequest = new MessageRequest
            {
                Addresses = new Dictionary<string, AddressConfiguration>
                {
                    {
                        toAddress,
                        new AddressConfiguration
                        {
                            ChannelType = "EMAIL"
                        }
                    }
                },
                MessageConfiguration = new DirectMessageConfiguration
                {
                    EmailMessage = new EmailMessage
                    {
                        FromAddress = senderAddress,
```

```
        SimpleEmail = new SimpleEmail
        {
            HtmlPart = new SimpleEmailPart
            {
                Charset = charset,
                Data = htmlBody
            },
            TextPart = new SimpleEmailPart
            {
                Charset = charset,
                Data = textBody
            },
            Subject = new SimpleEmailPart
            {
                Charset = charset,
                Data = subject
            }
        }
    }
}
};
try
{
    Console.WriteLine("Sending message...");
    SendMessagesResponse response = client.SendMessages(sendRequest);
    Console.WriteLine("Message sent!");
}
catch (Exception ex)
{
    Console.WriteLine("The message wasn't sent. Error message: " +
ex.Message);
}
}
}
```

Java

Use this example to send email by using the [AWS SDK for Java](#). This example assumes that you've already installed and configured the AWS SDK for Java 2.x. For more information, see [Getting Started](#) in the *AWS SDK for Java 2.x Developer Guide*.

This example assumes that you're using a shared credentials file to specify the Access Key and Secret Access Key for an existing IAM user. For more information, see [Set up AWS Credentials and Region for Development](#) in the *AWS SDK for Java Developer Guide*.

This code example was tested using the AWS SDK for Java version 2.3.1 and OpenJDK version 11.0.1.

```
package com.amazonaws.samples;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.AddressConfiguration;
import com.amazonaws.services.pinpoint.model.ChannelType;
import com.amazonaws.services.pinpoint.model.DirectMessageConfiguration;
import com.amazonaws.services.pinpoint.model.EmailMessage;
import com.amazonaws.services.pinpoint.model.MessageRequest;
```

```
import com.amazonaws.services.pinpoint.model.SendMessagesRequest;
import com.amazonaws.services.pinpoint.model.SimpleEmail;
import com.amazonaws.services.pinpoint.model.SimpleEmailPart;

public class SendMessages {

    // The AWS Region that you want to use to send the message. For a list of
    // AWS Regions where the Amazon Pinpoint API is available, see
    // https://docs.aws.amazon.com/pinpoint/latest/apireference/
    public static String region = "us-west-2";

    // The "From" address. This address has to be verified in Amazon
    // Pinpoint in the region you're using to send email.
    public static String senderAddress = "sender@example.com";

    // The address on the "To" line. If your Amazon Pinpoint account is in
    // the sandbox, this address also has to be verified.
    public static String toAddress = "recipient@example.com";

    // The Amazon Pinpoint project/application ID to use when you send this message.
    // Make sure that the SMS channel is enabled for the project or application
    // that you choose.
    public static String appId = "ce796be37f32f178af652b26eexample";

    // The subject line of the email.
    public static String subject = "Amazon Pinpoint test";

    // The email body for recipients with non-HTML email clients.
    static final String textBody = "Amazon Pinpoint Test (SDK for Java 2.x)\r\n"
        + "-----\r\n"
        + "This email was sent using the Amazon Pinpoint "
        + "API using the AWS SDK for Java 2.x.";

    // The body of the email for recipients whose email clients support
    // HTML content.
    static final String htmlBody = "<h1>Amazon Pinpoint test (AWS SDK for Java 2.x)</
h1>"
        + "<p>This email was sent through the <a href='https://aws.amazon.com/
pinpoint/'>"
        + "Amazon Pinpoint</a> Email API using the "
        + "<a href='https://aws.amazon.com/sdk-for-java/'>AWS SDK for Java 2.x</
a>";

    // The character encoding the you want to use for the subject line and
    // message body of the email.
    public static String charset = "UTF-8";

    public static void main(String[] args) throws IOException {

        try {
            Map<String,AddressConfiguration> addressMap =
                new HashMap<String,AddressConfiguration>();

            addressMap.put(senderAddress, new AddressConfiguration()
                .withChannelType(ChannelType.EMAIL));

            AmazonPinpoint client = AmazonPinpointClientBuilder.standard()
                .withRegion(region).build();

            SendMessagesRequest request = (new SendMessagesRequest()
                .withApplicationId(appId)
                .withMessageRequest(new MessageRequest()
                    .withAddresses(addressMap)
                    .withMessageConfiguration(new DirectMessageConfiguration()
                        .withEmailMessage(new EmailMessage()
                            .withSimpleEmail(new SimpleEmail()
```



```
        .withHtmlPart(new SimpleEmailPart()
            .withCharset(charset)
            .withData(htmlBody)
        )
        .withTextPart(new SimpleEmailPart()
            .withCharset(charset)
            .withData(textBody)
        )
        .withSubject(new SimpleEmailPart()
            .withCharset(charset)
            .withData(subject)
        )
    )
    )
    )
    );
    System.out.println("Sending message...");
    client.sendMessage(request);
    System.out.println("Message sent!");
} catch (Exception ex) {
    System.out.println("The message wasn't sent. Error message: "
        + ex.getMessage());
}
}
```

JavaScript (Node.js)

Use this example to send email by using the [AWS SDK for JavaScript in Node.js](#). This example assumes that you've already installed and configured the SDK for JavaScript in Node.js. For more information, see [Getting Started](#) in the *AWS SDK for JavaScript in Node.js Developer Guide*.

This example assumes that you're using a shared credentials file to specify the Access Key and Secret Access Key for an existing IAM user. For more information, see [Setting Credentials](#) in the *AWS SDK for JavaScript in Node.js Developer Guide*.

This code example was tested using the SDK for JavaScript in Node.js version 2.388.0 and Node.js version 11.7.0.

```
'use strict';

const AWS = require('aws-sdk');

// The AWS Region that you want to use to send the email. For a list of
// AWS Regions where the Amazon Pinpoint API is available, see
// https://docs.aws.amazon.com/pinpoint/latest/apireference/
const aws_region = "us-west-2"

// The "From" address. This address has to be verified in Amazon Pinpoint
// in the region that you use to send email.
const senderAddress = "sender@example.com";

// The address on the "To" line. If your Amazon Pinpoint account is in
// the sandbox, this address also has to be verified.
var toAddress = "recipient@example.com";

// The Amazon Pinpoint project/application ID to use when you send this message.
// Make sure that the SMS channel is enabled for the project or application
// that you choose.
const appId = "ce796be37f32f178af652b26eexample";
```

```
// The subject line of the email.
var subject = "Amazon Pinpoint (AWS SDK for JavaScript in Node.js)";

// The email body for recipients with non-HTML email clients.
var body_text = `Amazon Pinpoint Test (SDK for JavaScript in Node.js)
-----
This email was sent with Amazon Pinpoint using the AWS SDK for JavaScript in Node.js.
For more information, see https://aws.amazon.com/sdk-for-node-js/`;

// The body of the email for recipients whose email clients support HTML content.
var body_html = `
<head></head>
<body>
  <h1>Amazon Pinpoint Test (SDK for JavaScript in Node.js)</h1>
  <p>This email was sent with
    <a href='https://aws.amazon.com/pinpoint/'>the Amazon Pinpoint API</a> using the
    <a href='https://aws.amazon.com/sdk-for-node-js/'>
      AWS SDK for JavaScript in Node.js</a>.</p>
</body>
</html>`;

// The character encoding the you want to use for the subject line and
// message body of the email.
var charset = "UTF-8";

// Specify that you're using a shared credentials file.
var credentials = new AWS.SharedIniFileCredentials({profile: 'default'});
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({region:aws_region});

//Create a new Pinpoint object.
var pinpoint = new AWS.Pinpoint();

// Specify the parameters to pass to the API.
var params = {
  ApplicationId: appId,
  MessageRequest: {
    Addresses: {
      [toAddress]:{
        ChannelType: 'EMAIL'
      }
    },
  },
  MessageConfiguration: {
    EmailMessage: {
      FromAddress: senderAddress,
      SimpleEmail: {
        Subject: {
          Charset: charset,
          Data: subject
        },
      },
      HtmlPart: {
        Charset: charset,
        Data: body_html
      },
      TextPart: {
        Charset: charset,
        Data: body_text
      }
    }
  }
}
};
```

```
//Try to send the email.
pinpoint.sendMessage(params, function(err, data) {
  // If something goes wrong, print an error message.
  if(err) {
    console.log(err.message);
  } else {
    console.log("Email sent! Message ID: ", data['MessageResponse']['Result']
[toAddress]['MessageId']);
  }
});
```

Python

Use this example to send email by using the [AWS SDK for Python \(Boto 3\)](#). This example assumes that you've already installed and configured the SDK for Python (Boto 3). For more information, see [Quickstart](#) in the *AWS SDK for Python (Boto 3) API Reference*.

This example assumes that you're using a shared credentials file to specify the Access Key and Secret Access Key for an existing IAM user. For more information, see [Credentials](#) in the *AWS SDK for Python (Boto 3) API Reference*.

This code example was tested using the SDK for Python (Boto 3) version 1.9.62 and Python version 3.6.7.

```
import boto3
from botocore.exceptions import ClientError

# The AWS Region that you want to use to send the email. For a list of
# AWS Regions where the Amazon Pinpoint API is available, see
# https://docs.aws.amazon.com/pinpoint/latest/apireference/
AWS_REGION = "us-west-2"

# The "From" address. This address has to be verified in
# Amazon Pinpoint in the region you're using to send email.
SENDER = "Mary Major <sender@example.com>"

# The addresses on the "To" line. If your Amazon Pinpoint account is in
# the sandbox, these addresses also have to be verified.
TOADDRESS = "recipient@example.com"

# The Amazon Pinpoint project/application ID to use when you send this message.
# Make sure that the email channel is enabled for the project or application
# that you choose.
APPID = "ce796be37f32f178af652b26eexample"

# The subject line of the email.
SUBJECT = "Amazon Pinpoint Test (SDK for Python (Boto 3))"

# The body of the email for recipients whose email clients don't support HTML
# content.
BODY_TEXT = """Amazon Pinpoint Test (SDK for Python)
-----
This email was sent with Amazon Pinpoint using the AWS SDK for Python (Boto 3).
For more information, see https:#aws.amazon.com/sdk-for-python/
"""

# The body of the email for recipients whose email clients can display HTML
# content.
BODY_HTML = """<html>
<head></head>
<body>
```

```
<h1>Amazon Pinpoint Test (SDK for Python)</h1>
<p>This email was sent with
  <a href='https://aws.amazon.com/pinpoint/'>Amazon Pinpoint</a> using the
  <a href='https://aws.amazon.com/sdk-for-python/'>
    AWS SDK for Python (Boto 3)</a>.</p>
</body>
</html>

"""

# The character encoding that you want to use for the subject line and message
# body of the email.
CHARSET = "UTF-8"

# Create a new client and specify a region.
client = boto3.client('pinpoint', region_name=AWS_REGION)
try:
    response = client.send_messages(
        ApplicationId=APPID,
        MessageRequest={
            'Addresses': {
                TOADDRESS: {
                    'ChannelType': 'EMAIL'
                }
            },
            'MessageConfiguration': {
                'EmailMessage': {
                    'FromAddress': SENDER,
                    'SimpleEmail': {
                        'Subject': {
                            'Charset': CHARSET,
                            'Data': SUBJECT
                        },
                        'HtmlPart': {
                            'Charset': CHARSET,
                            'Data': BODY_HTML
                        },
                        'TextPart': {
                            'Charset': CHARSET,
                            'Data': BODY_TEXT
                        }
                    }
                }
            }
        }
    )
except ClientError as e:
    print(e.response['Error']['Message'])
else:
    print("Message sent! Message ID: "
          + response['MessageResponse']['Result'][TOADDRESS]['MessageId'])
```

Send Email by Using the Amazon Pinpoint Email API

This section contains complete code examples that you can use to send email through the Amazon Pinpoint Email API by using an AWS SDK.

C#

Use this example to send email by using the [AWS SDK for .NET](#). This example assumes that you've already installed and configured the AWS SDK for .NET. For more information, see [Getting Started with the AWS SDK for .NET](#) in the *AWS SDK for .NET Developer Guide*.

This example assumes that you're using a shared credentials file to specify the Access Key and Secret Access Key for an existing IAM user. For more information, see [Configuring AWS Credentials](#) in the *AWS SDK for .NET Developer Guide*.

This code example was tested using the AWS SDK for .NET version 3.3.29.13 and .NET Core runtime version 2.1.2.

```
using System;
using System.Collections.Generic;
using Amazon;
using Amazon.PinpointEmail;
using Amazon.PinpointEmail.Model;

namespace PinpointEmailSDK
{
    class MainClass
    {
        // The AWS Region that you want to use to send the email. For a list of
        // AWS Regions where the Amazon Pinpoint Email API is available, see
        // https://docs.aws.amazon.com/pinpoint-email/latest/APIReference
        static string region = "us-west-2";

        // The "From" address. This address has to be verified in Amazon
        // Pinpoint in the region you're using to send email.
        static string senderAddress = "sender@example.com";

        // The address on the "To" line. If your Amazon Pinpoint account is in
        // the sandbox, this address also has to be verified.
        static string toAddress = "recipient@example.com";

        // CC and BCC addresses. If your account is in the sandbox, these
        // addresses have to be verified.
        static string ccAddress = "cc-recipient@example.com";
        static string bccAddress = "bcc-recipient@example.com";

        // The configuration set that you want to use to send the email.
        static string configSet = "ConfigSet";

        // The subject line of the email.
        static string subject = "Amazon Pinpoint Email API test";

        // The body of the email for recipients whose email clients don't
        // support HTML content.
        static string textBody = "Amazon Pinpoint Email Test (.NET)\r\n"
            + "This email was sent using the Amazon Pinpoint Email"
            + "API using the AWS SDK for .NET.";

        // The body of the email for recipients whose email clients support
        // HTML content.
        static string htmlBody = @"<html>
<head></head>
<body>
  <h1>Amazon Pinpoint Email API Test (AWS SDK for .NET)</h1>
  <p>This email was sent using the
  <a href='https://aws.amazon.com/pinpoint/'>Amazon Pinpoint</a> Email API
  using the <a href='https://aws.amazon.com/sdk-for-net/'>
  AWS SDK for .NET</a>.</p>
</body>
</html>";

        // The message tags that you want to apply to the email.
        static MessageTag tag0 = new MessageTag
        {
            Name = "key0",
```

```
        Value = "value0"
    };
    static MessageTag tag1 = new MessageTag
    {
        Name = "key1",
        Value = "value1"
    };

    // The character encoding the you want to use for the subject line and
    // message body of the email.
    static string charset = "UTF-8";

    public static void Main(string[] args)
    {
        using (var client = new
AmazonPinpointEmailClient(RegionEndpoint.GetBySystemName(region)))
        {
            // Create a request to send the message. The request contains
            // all of the message attributes and content that were defined
            // earlier.
            var sendRequest = new SendEmailRequest
            {
                FromEmailAddress = senderAddress,

                // An object that contains all of the destinations that you
                // want to send the message to. You can send a message to up
                // to 50 recipients in a single call to the API.
                Destination = new Destination
                {
                    // You can include multiple To, CC, and BCC addresses.
                    ToAddresses = new List<string>
                    {
                        toAddress,
                        //additional recipients here
                    },
                    CcAddresses = new List<string>
                    {
                        ccAddress,
                        //additional recipients here
                    },
                    BccAddresses = new List<string>
                    {
                        bccAddress,
                        //additional recipients here
                    }
                },
            },

            // The body of the email message.
            Content = new EmailContent
            {
                // Create a new Simple email message. If you need to
                // include attachments, you should send a RawMessage
                // instead.
                Simple = new Message
                {
                    Subject = new Content
                    {
                        Charset = charset,
                        Data = subject
                    },
                    Body = new Body
                    {
                        // The text-only body of the message.
                        Text = new Content
                        {
                            Charset = charset,
```

```
        Data = textBody
    },
    // The HTML body of the message.
    Html = new Content
    {
        Charset = charset,
        Data = htmlBody
    }
    }
},
// The configuration set that you want to use when you send
// this message.
ConfigurationSetName = configSet,
// A list of tags that you want to apply to this message.
EmailTags = new List<MessageTag>
{
    tag0,
    tag1
}
};
// Send the email, and provide a confirmation message if it's
// sent successfully.
try
{
    Console.WriteLine("Sending email using the Amazon Pinpoint Email
API...");
    var response = client.SendEmail(sendRequest);
    Console.WriteLine("Email sent!");
}
// If the message can't be sent, return the error message that's
// provided by the Amazon Pinpoint Email API.
catch (Exception ex)
{
    Console.WriteLine("The email wasn't sent. ");
    Console.WriteLine("Error message: " + ex.Message);
}
}
}
}
```

Java

Use this example to send email by using the [AWS SDK for Java](#). This example assumes that you've already installed and configured the AWS SDK for Java 2.x. For more information, see [Getting Started](#) in the *AWS SDK for Java 2.x Developer Guide*.

This example assumes that you're using a shared credentials file to specify the Access Key and Secret Access Key for an existing IAM user. For more information, see [Set up AWS Credentials and Region for Development](#) in the *AWS SDK for Java Developer Guide*.

```
package com.amazonaws.samples;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collection;
```

```
import com.amazonaws.services.pinpointemail.AmazonPinpointEmail;
import com.amazonaws.services.pinpointemail.AmazonPinpointEmailClientBuilder;
import com.amazonaws.services.pinpointemail.model.Body;
import com.amazonaws.services.pinpointemail.model.Content;
import com.amazonaws.services.pinpointemail.model.Destination;
import com.amazonaws.services.pinpointemail.model.EmailContent;
import com.amazonaws.services.pinpointemail.model.Message;
import com.amazonaws.services.pinpointemail.model.MessageTag;
import com.amazonaws.services.pinpointemail.model.SendEmailRequest;

public class SendMessage {

    // The AWS Region that you want to use to send the email. For a list of
    // AWS Regions where the Amazon Pinpoint Email API is available, see
    // https://docs.aws.amazon.com/pinpoint-email/latest/APIReference
    static final String region = "us-west-2";

    // The "From" address. This address has to be verified in Amazon
    // Pinpoint in the region you're using to send email.
    static final String senderAddress = "sender@example.com";

    // The address on the "To" line. If your Amazon Pinpoint account is in
    // the sandbox, this address also has to be verified.
    static final String toAddress = "recipient@example.com";

    // CC and BCC addresses. If your account is in the sandbox, these
    // addresses have to be verified.
    // In this example, there are several CC recipients. Each address is separated
    // with a comma. We convert this string to an ArrayList later in the example.
    static final String ccAddress = "cc_recipient0@example.com,
cc_recipient1@example.com";
    static final String bccAddress = "bcc_recipient@example.com";

    // The configuration set that you want to use to send the email.
    static final String configurationSet = "ConfigSet";

    // The subject line of the email.
    static final String subject = "Amazon Pinpoint Email API test";

    // The email body for recipients with non-HTML email clients.
    static final String textBody = "Amazon Pinpoint Email Test (Java)\r\n"
        + "This email was sent using the Amazon Pinpoint "
        + "Email API using the AWS SDK for Java.";

    // The body of the email for recipients whose email clients support
    // HTML content.
    static final String htmlBody = "<h1>Amazon Pinpoint test (AWS SDK for Java)</h1>"
        + "<p>This email was sent through the <a href='https://aws.amazon.com/
pinpoint/'>"
        + "Amazon Pinpoint</a> Email API using the "
        + "<a href='https://aws.amazon.com/sdk-for-java/'>AWS SDK for Java</a>";

    // The message tags that you want to apply to the email.
    static final String tagKey0 = "key0";
    static final String tagValue0 = "value0";
    static final String tagKey1 = "key1";
    static final String tagValue1 = "value1";

    // The character encoding the you want to use for the subject line and
    // message body of the email.
    static final String charset = "UTF-8";

    public static void main(String[] args) throws IOException {

        // Convert comma-separated lists of To, CC, and BCC Addresses into collections.
        // This works even if the string only contains a single email address.
```



```
Collection<String> toAddresses =
    new ArrayList<String>(Arrays.asList(toAddress.split("\\s*,\\s*")));
Collection<String> ccAddresses =
    new ArrayList<String>(Arrays.asList(ccAddress.split("\\s*,\\s*")));
Collection<String> bccAddresses =
    new ArrayList<String>(Arrays.asList(bccAddress.split("\\s*,\\s*")));

try {
    // Create a new email client
    AmazonPinpointEmail client = AmazonPinpointEmailClientBuilder.standard()
        .withRegion(region).build();

    // Combine all of the components of the email to create a request.
    SendEmailRequest request = new SendEmailRequest()
        .withFromEmailAddress(senderAddress)
        .withConfigurationSetName(configurationSet)
        .withDestination(new Destination()
            .withToAddresses(toAddresses)
            .withCcAddresses(ccAddresses)
            .withBccAddresses(bccAddresses)
        )
        .withContent(new EmailContent()
            .withSimple(new Message()
                .withSubject(new Content()
                    .withCharset(charset)
                    .withData(subject)
                )
            )
            .withBody(new Body()
                .withHtml(new Content()
                    .withCharset(charset)
                    .withData(htmlBody)
                )
                .withText(new Content()
                    .withCharset(charset)
                    .withData(textBody)
                )
            )
        )
        .withEmailTags(new MessageTag()
            .withName(tagKey0)
            .withValue(tagValue0)
        )
        .withEmailTags(new MessageTag()
            .withName(tagKey1)
            .withValue(tagValue1)
        );
    client.sendEmail(request);
    System.out.println("Email sent!");
    System.out.println(request);
} catch (Exception ex) {
    System.out.println("The email wasn't sent. Error message: "
        + ex.getMessage());
}
}
```

JavaScript (Node.js)

Use this example to send email by using the [AWS SDK for JavaScript in Node.js](#). This example assumes that you've already installed and configured the SDK for JavaScript in Node.js. For more information, see [Getting Started](#) in the *AWS SDK for JavaScript in Node.js Developer Guide*.

This example assumes that you're using a shared credentials file to specify the Access Key and Secret Access Key for an existing IAM user. For more information, see [Setting Credentials](#) in the *AWS SDK for JavaScript in Node.js Developer Guide*.

This code example was tested using the SDK for JavaScript in Node.js version 2.388.0 and Node.js version 11.7.0.

```
'use strict';

var AWS = require('aws-sdk');

// The AWS Region that you want to use to send the email. For a list of
// AWS Regions where the Amazon Pinpoint Email API is available, see
// https://docs.aws.amazon.com//pinpoint-email/latest/APIReference
var aws_region = "us-west-2";

// The "From" address. This address has to be verified in Amazon Pinpoint
// in the region that you use to send email.
var senderAddress = "Mary Major <sender@example.com>";

// The address on the "To" line. If your Amazon Pinpoint account is in
// the sandbox, this address also has to be verified.
// Note: All recipient addresses in this example are in arrays, which makes it
// easier to specify multiple recipients. Alternatively, you can make these
// variables strings, and then modify the To/Cc/BccAddresses attributes in the
// params variable so that it passes an array for each recipient type.
var toAddresses = [ "recipient@example.com" ];

// CC and BCC addresses. If your account is in the sandbox, these
// addresses have to be verified.
var ccAddresses = [ "cc_recipient1@example.com", "cc_recipient2@example.com" ];
var bccAddresses = [ "bcc_recipient@example.com" ];

// The configuration set that you want to use to send the email.
var configuration_set = "ConfigSet";

// The subject line of the email.
var subject = "Amazon Pinpoint Test (AWS SDK for JavaScript in Node.js)";

// The email body for recipients with non-HTML email clients.
var body_text = `Amazon Pinpoint Test (SDK for JavaScript in Node.js)
-----
This email was sent with Amazon Pinpoint using the AWS SDK for JavaScript in Node.js.
For more information, see https://aws.amazon.com/sdk-for-node-js/`;

// The body of the email for recipients whose email clients support HTML content.
var body_html = `
<head></head>
<body>
  <h1>Amazon Pinpoint Test (SDK for JavaScript in Node.js)</h1>
  <p>This email was sent with
    <a href='https://aws.amazon.com//pinpoint/'>the Amazon Pinpoint Email API</a> using
    the
    <a href='https://aws.amazon.com//sdk-for-node-js/'>
      AWS SDK for JavaScript in Node.js</a>.</p>
</body>
</html>`;

// The message tags that you want to apply to the email.
var tag0 = { 'Name':'key0', 'Value':'value0' };
var tag1 = { 'Name':'key1', 'Value':'value1' };

// The character encoding the you want to use for the subject line and
// message body of the email.
```

```
var charset = "UTF-8";

// Specify that you're using a shared credentials file, and specify which
// profile to use in the shared credentials file.
var credentials = new AWS.SharedIniFileCredentials({profile: 'default'});
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({region:aws_region});

//Create a new PinpointEmail object.
var pinpointEmail = new AWS.PinpointEmail();

// Specify the parameters to pass to the API.
var params = {
  FromEmailAddress: senderAddress,
  Destination: {
    ToAddresses: toAddresses,
    CcAddresses: ccAddresses,
    BccAddresses: bccAddresses
  },
  Content: {
    Simple: {
      Body: {
        Html: {
          Data: body_html,
          Charset: charset
        },
        Text: {
          Data: body_text,
          Charset: charset
        }
      },
      Subject: {
        Data: subject,
        Charset: charset
      }
    }
  },
  ConfigurationSetName: configuration_set,
  EmailTags: [
    tag0,
    tag1
  ]
};

//Try to send the email.
pinpointEmail.sendEmail(params, function(err, data) {
  // If something goes wrong, print an error message.
  if(err) {
    console.log(err.message);
  } else {
    console.log("Email sent! Message ID: ", data.MessageId);
  }
});
```

Python

Use this example to send email by using the [AWS SDK for Python \(Boto 3\)](#). This example assumes that you've already installed and configured the SDK for Python (Boto 3). For more information, see [Quickstart](#) in the *AWS SDK for Python (Boto 3) API Reference*.

This example assumes that you're using a shared credentials file to specify the Access Key and Secret Access Key for an existing IAM user. For more information, see [Credentials](#) in the *AWS SDK for Python (Boto 3) API Reference*.

This code example was tested using the SDK for Python (Boto 3) version 1.9.62 and Python version 3.6.7.

```
import boto3
from botocore.exceptions import ClientError

# The AWS Region that you want to use to send the email. For a list of
# AWS Regions where the Amazon Pinpoint Email API is available, see
# https://docs.aws.amazon.com/pinpoint-email/latest/APIReference
AWS_REGION = "us-west-2"

# The "From" address. This address has to be verified in
# Amazon Pinpoint in the region you're using to send email.
SENDER = "Mary Major <sender@example.com>"

# The addresses on the "To" line. If your Amazon Pinpoint account is in
# the sandbox, these addresses also have to be verified.
TOADDRESSES = ["recipient@example.com"]

# CC and BCC addresses. If your account is in the sandbox, these
# addresses have to be verified.
CCADDRESSES = ["cc_recipient1@example.com", "cc_recipient2@example.com"]
BCCADDRESSES = ["bcc_recipient@example.com"]

# The configuration set that you want to use to send the email.
CONFIGURATION_SET = "ConfigSet"

# The subject line of the email.
SUBJECT = "Amazon Pinpoint Test (SDK for Python)"

# The body of the email for recipients whose email clients don't support HTML
# content.
BODY_TEXT = """Amazon Pinpoint Test (SDK for Python)
-----
This email was sent with Amazon Pinpoint using the AWS SDK for Python.
For more information, see https:#aws.amazon.com/sdk-for-python/
"""

# The body of the email for recipients whose email clients can display HTML
# content.
BODY_HTML = """<html>
<head></head>
<body>
  <h1>Amazon Pinpoint Test (SDK for Python)</h1>
  <p>This email was sent with
  <a href='https:#aws.amazon.com/pinpoint/'>Amazon Pinpoint</a> using the
  <a href='https:#aws.amazon.com/sdk-for-python/'>
    AWS SDK for Python</a>.</p>
</body>
</html>
"""

# The message tags that you want to apply to the email.
TAG0 = {'Name': 'key0', 'Value': 'value0'}
TAG1 = {'Name': 'key1', 'Value': 'value1'}

# The character encoding that you want to use for the subject line and message
# body of the email.
CHARSET = "UTF-8"
```

```
# Create a new Pinpoint resource and specify a region.
client = boto3.client('pinpoint-email', region_name=AWS_REGION)

# Send the email.
try:
    # Create a request to send the email. The request contains all of the
    # message attributes and content that were defined earlier.
    response = client.send_email(

        FromEmailAddress=SENDER,

        # An object that contains all of the email addresses that you want to
        # send the message to. You can send a message to up to 50 recipients in
        # a single call to the API.
        Destination={
            'ToAddresses': TOADDRESSES,
            'CcAddresses': CCADDRESSES,
            'BccAddresses': BCCADDRESSES
        },
        # The body of the email message.
        Content={
            # Create a new Simple message. If you need to include attachments,
            # you should send a RawMessage instead.
            'Simple': {
                'Subject': {
                    'Charset': CHARSET,
                    'Data': SUBJECT,
                },
                'Body': {
                    'Html': {
                        'Charset': CHARSET,
                        'Data': BODY_HTML
                    },
                    'Text': {
                        'Charset': CHARSET,
                        'Data': BODY_TEXT,
                    }
                }
            }
        },
        # The configuration set that you want to use when you send this message.
        ConfigurationSetName=CONFIGURATION_SET,
        EmailTags=[
            TAG0,
            TAG1
        ]
    )
except ClientError as e:
    print("The message wasn't sent. Error message: \"" + e.response['Error']['Message']
          + "\"")
else:
    print("Email sent!")
    print("Message ID: " + response['MessageId'])
```

Send Email by Using the Amazon Pinpoint SMTP Interface

This section contains complete code examples that you can use to send email from your apps by using the Amazon Pinpoint SMTP interface. These examples use standard email-sending libraries whenever possible.

These examples assume that you've already created an Amazon Pinpoint SMTP user name and password. For more information, see [Obtaining SMTP Credentials](#) in the *Amazon Pinpoint User Guide*.

C#

Use this example to send email by using classes in the .NET [System.Net.Mail](#) namespace.

```
using System;
using System.Net;
using System.Net.Mail;

namespace PinpointEmailSMTP
{
    class MainClass
    {
        // If you're using Amazon Pinpoint in a region other than US West (Oregon),
        // replace email-smtp.us-west-2.amazonaws.com with the Amazon Pinpoint SMTP
        // endpoint in the appropriate AWS Region.
        static string smtpEndpoint = "email-smtp.us-west-2.amazonaws.com";

        // The port to use when connecting to the SMTP server.
        static int port = 587;

        // Replace sender@example.com with your "From" address.
        // This address must be verified with Amazon Pinpoint.
        static string senderName = "Mary Major";
        static string senderAddress = "sender@example.com";

        // Replace recipient@example.com with a "To" address. If your account
        // is still in the sandbox, this address must be verified.
        static string toAddress = "recipient@example.com";

        // CC and BCC addresses. If your account is in the sandbox, these
        // addresses have to be verified.
        static string ccAddress = "cc-recipient@example.com";
        static string bccAddress = "bcc-recipient@example.com";

        // Replace smtp_username with your Amazon Pinpoint SMTP user name.
        static string smtpUsername = "AKIAIOSFODNN7EXAMPLE";

        // Replace smtp_password with your Amazon Pinpoint SMTP user name.
        static string smtpPassword = "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY";

        // (Optional) the name of a configuration set to use for this message.
        static string configurationSet = "ConfigSet";

        // The subject line of the email
        static string subject =
            "Amazon Pinpoint test (SMTP interface accessed using C#)";

        // The body of the email for recipients whose email clients don't
        // support HTML content.
        static AlternateView textBody = AlternateView.
            CreateAlternateViewFromString("Amazon Pinpoint Email Test (.NET)\r\n"
                + "This email was sent using the Amazon Pinpoint SMTP "
                + "interface.", null, "text/plain");

        // The body of the email for recipients whose email clients support
        // HTML content.
        static AlternateView htmlBody = AlternateView.
            CreateAlternateViewFromString("<html><head></head><body>"
                + "<h1>Amazon Pinpoint SMTP Interface Test</h1><p>This "
                + "email was sent using the "
                + "<a href='https://aws.amazon.com/pinpoint/'>Amazon Pinpoint"
```

```
        + "</a> SMTP interface.</p></body></html>", null, "text/html");

// The message tags that you want to apply to the email.
static string tag0 = "key0=value0";
static string tag1 = "key1=value1";

public static void Main(string[] args)
{
    // Create a new MailMessage object
    MailMessage message = new MailMessage();

    // Add sender and recipient email addresses to the message
    message.From = new MailAddress(senderAddress, senderName);
    message.To.Add(new MailAddress(toAddress));
    message.CC.Add(new MailAddress(ccAddress));
    message.Bcc.Add(new MailAddress(bccAddress));

    // Add the subject line, text body, and HTML body to the message
    message.Subject = subject;
    message.AlternateViews.Add(textBody);
    message.AlternateViews.Add(htmlBody);

    // Add optional headers for configuration set and message tags to the
message
    message.Headers.Add("X-SES-CONFIGURATION-SET", configurationSet);
    message.Headers.Add("X-SES-MESSAGE-TAGS", tag0);
    message.Headers.Add("X-SES-MESSAGE-TAGS", tag1);

    using (var client = new System.Net.Mail.SmtpClient(smtpEndpoint, port))
    {
        // Create a Credentials object for connecting to the SMTP server
        client.Credentials =
            new NetworkCredential(smtpUsername, smtpPassword);

        client.EnableSsl = true;

        // Send the message
        try
        {
            Console.WriteLine("Attempting to send email...");
            client.Send(message);
            Console.WriteLine("Email sent!");
        }
        // Show an error message if the message can't be sent
        catch (Exception ex)
        {
            Console.WriteLine("The email wasn't sent.");
            Console.WriteLine("Error message: " + ex.Message);
        }
    }
}
}
```

Java

Use this example to send email by using the [JavaMail API](#).

```
import java.util.Properties;

import javax.mail.Message;
import javax.mail.Session;
import javax.mail.Transport;
```

```
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

public class SendEmail {

    // If you're using Amazon Pinpoint in a region other than US West (Oregon),
    // replace email-smtp.us-west-2.amazonaws.com with the Amazon Pinpoint SMTP
    // endpoint in the appropriate AWS Region.
    static final String smtpEndpoint = "email-smtp.us-west-2.amazonaws.com";

    // The port to use when connecting to the SMTP server.
    static final int port = 587;

    // Replace sender@example.com with your "From" address.
    // This address must be verified with Amazon Pinpoint.
    static final String senderName= "Mary Major";
    static final String senderAddress = "sender@example.com";

    // Replace recipient@example.com with a "To" address. If your account
    // is still in the sandbox, this address must be verified. To specify
    // multiple addresses, separate each address with a comma.
    static final String toAddresses = "recipient@example.com";

    // CC and BCC addresses. If your account is in the sandbox, these
    // addresses have to be verified. To specify multiple addresses, separate
    // each address with a comma.
    static final String ccAddresses = "cc-recipient0@example.com,cc-
recipient1@example.com";
    static final String bccAddresses = "bcc-recipient@example.com";

    // Replace smtp_username with your Amazon Pinpoint SMTP user name.
    static final String smtpUsername = "AKIAIOSFODNN7EXAMPLE";

    // Replace smtp_password with your Amazon Pinpoint SMTP password.
    static final String smtpPassword = "wJalrXUtnFEMI/K7MDENG/bPxrFcYEXAMPLEKEY";

    // (Optional) the name of a configuration set to use for this message.
    static final String configurationSet = "ConfigSet";

    // The subject line of the email
    static final String subject = "Amazon Pinpoint test (SMTP interface accessed using
Java)";

    // The body of the email for recipients whose email clients don't
    // support HTML content.
    static final String htmlBody = String.join(
        System.getProperty("line.separator"),
        "<h1>Amazon Pinpoint SMTP Email Test</h1>",
        "<p>This email was sent with Amazon Pinpoint using the ",
        "<a href='https://github.com/javaee/javamail'>Javamail Package</a>",
        " for <a href='https://www.java.com'>Java</a>."
    );

    // The message tags that you want to apply to the email.
    static final String tag0 = "key0=value0";
    static final String tag1 = "key1=value1";

    public static void main(String[] args) throws Exception {

        // Create a Properties object to contain connection configuration information.
        Properties props = System.getProperties();
        props.put("mail.transport.protocol", "smtp");
        props.put("mail.smtp.port", port);
        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.smtp.auth", "true");
    }
}
```



```
// Create a Session object to represent a mail session with the specified
properties.
Session session = Session.getDefaultInstance(props);

// Create a message with the specified information.
MimeMessage msg = new MimeMessage(session);
msg.setFrom(new InternetAddress(senderAddress, senderName));
msg.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(toAddresses));
msg.setRecipients(Message.RecipientType.CC,
InternetAddress.parse(ccAddresses));
msg.setRecipients(Message.RecipientType.BCC,
InternetAddress.parse(bccAddresses));

msg.setSubject(subject);
msg.setContent(htmlBody, "text/html");

// Add headers for configuration set and message tags to the message.
msg.setHeader("X-SES-CONFIGURATION-SET", configurationSet);
msg.setHeader("X-SES-MESSAGE-TAGS", tag0);
msg.setHeader("X-SES-MESSAGE-TAGS", tag1);

// Create a transport.
Transport transport = session.getTransport();

// Send the message.
try {
    System.out.println("Sending...");

    // Connect to Amazon Pinpoint using the SMTP username and password you
specified above.
    transport.connect(smtpEndpoint, smtpUsername, smtpPassword);

    // Send the email.
    transport.sendMessage(msg, msg.getAllRecipients());
    System.out.println("Email sent!");
}
catch (Exception ex) {
    System.out.println("The email wasn't sent. Error message: "
+ ex.getMessage());
}
finally {
    // Close the connection to the SMTP server.
    transport.close();
}
}
```

JavaScript (Node.js)

Use this example to send email by using the [Nodemailer](#) module for Node.js.

```
"use strict";
const nodemailer = require("nodemailer");

// If you're using Amazon Pinpoint in a region other than US West (Oregon),
// replace email-smtp.us-west-2.amazonaws.com with the Amazon Pinpoint SMTP
// endpoint in the appropriate AWS Region.
const smtpEndpoint = "email-smtp.us-west-2.amazonaws.com";

// The port to use when connecting to the SMTP server.
const port = 587;
```

Amazon Pinpoint Developer Guide Using the SMTP Interface

```
// Replace sender@example.com with your "From" address.
// This address must be verified with Amazon Pinpoint.
const senderAddress = "Mary Major <sender@example.com>";

// Replace recipient@example.com with a "To" address. If your account
// is still in the sandbox, this address must be verified. To specify
// multiple addresses, separate each address with a comma.
var toAddresses = "recipient@example.com";

// CC and BCC addresses. If your account is in the sandbox, these
// addresses have to be verified. To specify multiple addresses, separate
// each address with a comma.
var ccAddresses = "cc-recipient0@example.com,cc-recipient1@example.com";
var bccAddresses = "bcc-recipient@example.com";

// Replace smtp_username with your Amazon Pinpoint SMTP user name.
const smtpUsername = "AKIAIOSFODNN7EXAMPLE";

// Replace smtp_password with your Amazon Pinpoint SMTP password.
const smtpPassword = "wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY";

// (Optional) the name of a configuration set to use for this message.
var configurationSet = "ConfigSet";

// The subject line of the email
var subject = "Amazon Pinpoint test (Nodemailer)";

// The email body for recipients with non-HTML email clients.
var body_text = `Amazon Pinpoint Test (Nodemailer)
-----
This email was sent through the Amazon Pinpoint SMTP interface using Nodemailer.
`;

// The body of the email for recipients whose email clients support HTML content.
var body_html = `
<head></head>
<body>
  <h1>Amazon Pinpoint Test (Nodemailer)</h1>
  <p>This email was sent with <a href='https://aws.amazon.com/pinpoint/'>Amazon
  Pinpoint</a>
    using <a href='https://nodemailer.com'>Nodemailer</a> for Node.js.</p>
</body>
</html>`;

// The message tags that you want to apply to the email.
var tag0 = "key0=value0";
var tag1 = "key1=value1";

async function main(){

  // Create the SMTP transport.
  let transporter = nodemailer.createTransport({
    host: smtpEndpoint,
    port: port,
    secure: false, // true for 465, false for other ports
    auth: {
      user: smtpUsername,
      pass: smtpPassword
    }
  });

  // Specify the fields in the email.
  let mailOptions = {
    from: senderAddress,
    to: toAddresses,
    subject: subject,
```

```
    cc: ccAddresses,
    bcc: bccAddresses,
    text: body_text,
    html: body_html,
    // Custom headers for configuration set and message tags.
    headers: {
      'X-SES-CONFIGURATION-SET': configurationSet,
      'X-SES-MESSAGE-TAGS': tag0,
      'X-SES-MESSAGE-TAGS': tag1
    }
  };

  // Send the email.
  let info = await transporter.sendMail(mailOptions)

  console.log("Message sent! Message ID: ", info.messageId);
}

main().catch(console.error);
```

Python

Use this example to send email by using the Python [email](#) and [smtplib](#) libraries.

```
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

# If you're using Amazon Pinpoint in a region other than US West (Oregon),
# replace email-smtp.us-west-2.amazonaws.com with the Amazon Pinpoint SMTP
# endpoint in the appropriate AWS Region.
HOST = "email-smtp.us-west-2.amazonaws.com"

# The port to use when connecting to the SMTP server.
PORT = 587

# Replace sender@example.com with your "From" address.
# This address must be verified.
SENDER = 'Mary Major <sender@example.com>'

# Replace recipient@example.com with a "To" address. If your account
# is still in the sandbox, this address has to be verified.
RECIPIENT = 'recipient@example.com'

# CC and BCC addresses. If your account is in the sandbox, these
# addresses have to be verified.
CCRECIPIENT = "cc_recipient@example.com"
BCCRECIPIENT = "bcc_recipient@example.com"

# Replace smtp_username with your Amazon Pinpoint SMTP user name.
USERNAME_SMTP = "AKIAIOSFODNN7EXAMPLE"

# Replace smtp_password with your Amazon Pinpoint SMTP password.
PASSWORD_SMTP = "wJalrXUtnFEMI/K7MDENG/bPxrFcicYEXAMPLEKEY"

# (Optional) the name of a configuration set to use for this message.
# If you comment out this line, you also need to remove or comment out
# the "X-Pinpoint-CONFIGURATION-SET:" header below.
CONFIGURATION_SET = "ConfigSet"

# The subject line of the email.
SUBJECT = 'Amazon Pinpoint Test (Python smtplib)'
```

```
# The email body for recipients with non-HTML email clients.
BODY_TEXT = ("Amazon Pinpoint Test\r\n"
             "This email was sent through the Amazon Pinpoint SMTP "
             "Interface using the Python smtplib package."
            )

# Create a MIME part for the text body.
textPart = MIMEText(BODY_TEXT, 'plain')

# The HTML body of the email.
BODY_HTML = """<html>
<head></head>
<body>
  <h1>Amazon Pinpoint SMTP Email Test</h1>
  <p>This email was sent with Amazon Pinpoint using the
  <a href='https://www.python.org/'>Python</a>
  <a href='https://docs.python.org/3/library/smtplib.html'>
  smtplib</a> library.</p>
</body>
</html>
"""

# Create a MIME part for the HTML body.
htmlPart = MIMEText(BODY_HTML, 'html')

# The message tags that you want to apply to the email.
TAG0 = "key0=value0"
TAG1 = "key1=value1"

# Create message container. The correct MIME type is multipart/alternative.
msg = MIMEMultipart('alternative')

# Add sender and recipient addresses to the message
msg['From'] = SENDER
msg['To'] = RECIPIENT
msg['Cc'] = CCRECIPIENT
msg['Bcc'] = BCCRECIPIENT

# Add the subject line, text body, and HTML body to the message.
msg['Subject'] = SUBJECT
msg.attach(textPart)
msg.attach(htmlPart)

# Add headers for configuration set and message tags to the message.
msg.add_header('X-SES-CONFIGURATION-SET', CONFIGURATION_SET)
msg.add_header('X-SES-MESSAGE-TAGS', TAG0)
msg.add_header('X-SES-MESSAGE-TAGS', TAG1)

# Open a new connection to the SMTP server and begin the SMTP conversation.
try:
    with smtplib.SMTP(HOST, PORT) as server:
        server.ehlo()
        server.starttls()
        #stmplib docs recommend calling ehlo() before and after starttls()
        server.ehlo()
        server.login(USERNAME_SMTP, PASSWORD_SMTP)
        #Uncomment the next line to send SMTP server responses to stdout
        #server.set_debuglevel(1)
        server.sendmail(SENDER, RECIPIENT, msg.as_string())
except Exception as e:
    print ("Error: ", e)
else:
    print ("Email sent!")
```

Send SMS Messages

You can use the Amazon Pinpoint API to send SMS messages (text messages) to specific phone numbers or endpoint IDs. This section contains complete code examples that you can use to send SMS messages through the Amazon Pinpoint API by using an AWS SDK.

C#

Use this example to send an SMS message by using the [AWS SDK for .NET](#). This example assumes that you've already installed and configured the AWS SDK for .NET. For more information, see [Getting Started](#) in the AWS SDK for .NET Developer Guide.

This example assumes that you're using a shared credentials file to specify the Access Key and Secret Access Key for an existing IAM user. For more information, see [Configuring AWS Credentials](#) in the *AWS SDK for .NET Developer Guide*.

```
using System;
using System.Collections.Generic;
using Amazon;
using Amazon.Pinpoint;
using Amazon.Pinpoint.Model;

namespace SendMessage
{
    class MainClass
    {
        // The AWS Region that you want to use to send the message. For a list of
        // AWS Regions where the Amazon Pinpoint API is available, see
        // https://docs.aws.amazon.com/pinpoint/latest/apireference/
        private static readonly string region = "us-east-1";

        // The phone number or short code to send the message from. The phone number
        // or short code that you specify has to be associated with your Amazon
        Pinpoint
        // account. For best results, specify long codes in E.164 format.
        private static readonly string originationNumber = "+12065550199";

        // The recipient's phone number. For best results, you should specify the
        // phone number in E.164 format.
        private static readonly string destinationNumber = "+14255550142";

        // The content of the SMS message.
        Pinpoint
        private static readonly string message = "This message was sent through Amazon
            + "using the AWS SDK for .NET. Reply STOP to opt out.";

        // The Pinpoint project/application ID to use when you send this message.
        // Make sure that the SMS channel is enabled for the project or application
        // that you choose.
        private static readonly string appId = "ce796be37f32f178af652b26eexample";

        // The type of SMS message that you want to send. If you plan to send
        // time-sensitive content, specify TRANSACTIONAL. If you plan to send
        // marketing-related content, specify PROMOTIONAL.
        private static readonly string messageType = "TRANSACTIONAL";

        // The registered keyword associated with the originating short code.
        private static readonly string registeredKeyword = "myKeyword";

        // The sender ID to use when sending the message. Support for sender ID
        // varies by country or region. For more information, see
```

```
// https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
private static readonly string senderId = "mySenderId";

public static void Main(string[] args)
{
    using (AmazonPinpointClient client = new
AmazonPinpointClient(RegionEndpoint.GetBySystemName(region)))
    {
        SendMessagesRequest sendRequest = new SendMessagesRequest
        {
            ApplicationId = appId,
            MessageRequest = new MessageRequest
            {
                Addresses = new Dictionary<string, AddressConfiguration>
                {
                    {
                        destinationNumber,
                        new AddressConfiguration
                        {
                            ChannelType = "SMS"
                        }
                    }
                },
                MessageConfiguration = new DirectMessageConfiguration
                {
                    SMSMessage = new SMSMessage
                    {
                        Body = message,
                        MessageType = messageType,
                        OriginationNumber = originationNumber,
                        SenderId = senderId,
                        Keyword = registeredKeyword
                    }
                }
            }
        };
        try
        {
            Console.WriteLine("Sending message...");
            SendMessagesResponse response = client.SendMessages(sendRequest);
            Console.WriteLine("Message sent!");
        }
        catch (Exception ex)
        {
            Console.WriteLine("The message wasn't sent. Error message: " +
ex.Message);
        }
    }
}
}
```

Java

Use this example to send an SMS message by using the [AWS SDK for Java](#). This example assumes that you've already installed and configured the SDK for Java. For more information, see [Getting Started](#) in the *AWS SDK for Java Developer Guide*.

This example assumes that you're using a shared credentials file to specify the Access Key and Secret Access Key for an existing IAM user. For more information, see [Set up AWS Credentials and Region for Development](#) in the *AWS SDK for Java Developer Guide*.

```
package com.amazonaws.samples;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

import com.amazonaws.services.pinpoint.AmazonPinpoint;
import com.amazonaws.services.pinpoint.AmazonPinpointClientBuilder;
import com.amazonaws.services.pinpoint.model.AddressConfiguration;
import com.amazonaws.services.pinpoint.model.ChannelType;
import com.amazonaws.services.pinpoint.model.DirectMessageConfiguration;
import com.amazonaws.services.pinpoint.model.MessageRequest;
import com.amazonaws.services.pinpoint.model.SMSMessage;
import com.amazonaws.services.pinpoint.model.SendMessagesRequest;

public class SendMessage {

    // The AWS Region that you want to use to send the message. For a list of
    // AWS Regions where the Amazon Pinpoint API is available, see
    // https://docs.aws.amazon.com/pinpoint/latest/apireference/
    public static String region = "us-east-1";

    // The phone number or short code to send the message from. The phone number
    // or short code that you specify has to be associated with your Amazon Pinpoint
    // account. For best results, specify long codes in E.164 format.
    public static String originationNumber = "+12065550199";

    // The recipient's phone number. For best results, you should specify the
    // phone number in E.164 format.
    public static String destinationNumber = "+14255550142";

    // The content of the SMS message.
    public static String message = "This message was sent through Amazon Pinpoint "
        + "using the AWS SDK for Java. Reply STOP to "
        + "opt out.";

    // The Pinpoint project/application ID to use when you send this message.
    // Make sure that the SMS channel is enabled for the project or application
    // that you choose.
    public static String appId = "ce796be37f32f178af652b26eexample";

    // The type of SMS message that you want to send. If you plan to send
    // time-sensitive content, specify TRANSACTIONAL. If you plan to send
    // marketing-related content, specify PROMOTIONAL.
    public static String messageType = "TRANSACTIONAL";

    // The registered keyword associated with the originating short code.
    public static String registeredKeyword = "myKeyword";

    // The sender ID to use when sending the message. Support for sender ID
    // varies by country or region. For more information, see
    // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
    // countries.html
    public static String senderId = "MySenderId";

    public static void main(String[] args) throws IOException {

        try {
            Map<String,AddressConfiguration> addressMap =
                new HashMap<String,AddressConfiguration>();

            addressMap.put(destinationNumber, new AddressConfiguration()
                .withChannelType(ChannelType.SMS));

            AmazonPinpoint client = AmazonPinpointClientBuilder.standard()
```

```
        .withRegion(region).build();

    SendMessagesRequest request = new SendMessagesRequest()
        .withApplicationId(appId)
        .withMessageRequest(new MessageRequest()
            .withAddresses(addressMap)
            .withMessageConfiguration(new DirectMessageConfiguration()
                .withSMSMessage(new SMSMessage()
                    .withBody(message)
                    .withMessageType(messageType)
                    .withOriginationNumber(originationNumber)
                    .withSenderId(senderId)
                    .withKeyword(registeredKeyword)
                )
            )
        );
    System.out.println("Sending message...");
    client.sendMessages(request);
    System.out.println("Message sent!");
} catch (Exception ex) {
    System.out.println("The message wasn't sent. Error message: "
        + ex.getMessage());
}
}
```

JavaScript (Node.js)

Use this example to send an SMS message by using the [AWS SDK for JavaScript in Node.js](#). This example assumes that you've already installed and configured the SDK for JavaScript in Node.js. For more information, see [Getting Started](#) in the *AWS SDK for JavaScript in Node.js Developer Guide*.

This example assumes that you're using a shared credentials file to specify the Access Key and Secret Access Key for an existing IAM user. For more information, see [Setting Credentials](#) in the *AWS SDK for JavaScript in Node.js Developer Guide*.

```
'use strict';

var AWS = require('aws-sdk');

// The AWS Region that you want to use to send the message. For a list of
// AWS Regions where the Amazon Pinpoint API is available, see
// https://docs.aws.amazon.com/pinpoint/latest/apireference/.
var aws_region = "us-east-1";

// The phone number or short code to send the message from. The phone number
// or short code that you specify has to be associated with your Amazon Pinpoint
// account. For best results, specify long codes in E.164 format.
var originationNumber = "+12065550199";

// The recipient's phone number. For best results, you should specify the
// phone number in E.164 format.
var destinationNumber = "+14255550142";

// The content of the SMS message.
var message = "This message was sent through Amazon Pinpoint "
    + "using the AWS SDK for JavaScript in Node.js. Reply STOP to "
    + "opt out.";

// The Amazon Pinpoint project/application ID to use when you send this message.
// Make sure that the SMS channel is enabled for the project or application
// that you choose.
```



```
var applicationId = "ce796be37f32f178af652b26eexample";

// The type of SMS message that you want to send. If you plan to send
// time-sensitive content, specify TRANSACTIONAL. If you plan to send
// marketing-related content, specify PROMOTIONAL.
var messageType = "TRANSACTIONAL";

// The registered keyword associated with the originating short code.
var registeredKeyword = "myKeyword";

// The sender ID to use when sending the message. Support for sender ID
// varies by country or region. For more information, see
// https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-countries.html
var senderId = "MySenderId";

// Specify that you're using a shared credentials file, and optionally specify
// the profile that you want to use.
var credentials = new AWS.SharedIniFileCredentials({profile: 'default'});
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({region:aws_region});

//Create a new Pinpoint object.
var pinpoint = new AWS.Pinpoint();

// Specify the parameters to pass to the API.
var params = {
  ApplicationId: applicationId,
  MessageRequest: {
    Addresses: {
      [destinationNumber]: {
        ChannelType: 'SMS'
      }
    },
    MessageConfiguration: {
      SMSMessage: {
        Body: message,
        Keyword: registeredKeyword,
        MessageType: messageType,
        OriginationNumber: originationNumber,
        SenderId: senderId,
      }
    }
  }
};

//Try to send the message.
pinpoint.sendMessage(params, function(err, data) {
  // If something goes wrong, print an error message.
  if(err) {
    console.log(err.message);
  } // Otherwise, show the unique ID for the message.
  } else {
    console.log("Message sent! "
      + data['MessageResponse']['Result']['destinationNumber']['StatusMessage']);
  }
});
```

Python

Use this example to send an SMS message by using the [AWS SDK for Python \(Boto 3\)](#). This example assumes that you've already installed and configured the SDK for Python. For more information, see [Quickstart](#) in the AWS SDK for Python (Boto 3) Getting Started.

This example assumes that you're using a shared credentials file to specify the Access Key and Secret Access Key for an existing IAM user. For more information, see [Credentials](#) in the *AWS SDK for Python (Boto 3) API Reference*.

```
import boto3
from botocore.exceptions import ClientError

# The AWS Region that you want to use to send the message. For a list of
# AWS Regions where the Amazon Pinpoint API is available, see
# https://docs.aws.amazon.com/pinpoint/latest/apireference/
region = "us-east-1"

# The phone number or short code to send the message from. The phone number
# or short code that you specify has to be associated with your Amazon Pinpoint
# account. For best results, specify long codes in E.164 format.
originationNumber = "+12065550199"

# The recipient's phone number. For best results, you should specify the
# phone number in E.164 format.
destinationNumber = "+14255550142"

# The content of the SMS message.
message = ("This is a sample message sent from Amazon Pinpoint by using the "
          "AWS SDK for Python (Boto 3).")

# The Amazon Pinpoint project/application ID to use when you send this message.
# Make sure that the SMS channel is enabled for the project or application
# that you choose.
applicationId = "ce796be37f32f178af652b26eexample"

# The type of SMS message that you want to send. If you plan to send
# time-sensitive content, specify TRANSACTIONAL. If you plan to send
# marketing-related content, specify PROMOTIONAL.
messageType = "TRANSACTIONAL"

# The registered keyword associated with the originating short code.
registeredKeyword = "myKeyword"

# The sender ID to use when sending the message. Support for sender ID
# varies by country or region. For more information, see
# https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-countries.html
senderId = "MySenderId"

# Create a new client and specify a region.
client = boto3.client('pinpoint', region_name=region)
try:
    response = client.send_messages(
        ApplicationId=applicationId,
        MessageRequest={
            'Addresses': {
                destinationNumber: {
                    'ChannelType': 'SMS'
                }
            },
            'MessageConfiguration': {
                'SMSMessage': {
                    'Body': message,
                    'Keyword': registeredKeyword,
                    'MessageType': messageType,
                    'OriginationNumber': originationNumber,
                    'SenderId': senderId
                }
            }
        }
    )
```

```
)  
  
except ClientError as e:  
    print(e.response['Error']['Message'])  
else:  
    print("Message sent! Message ID: "  
          + response['MessageResponse']['Result']['destinationNumber']['MessageId'])
```

Send Voice Messages

You can use the Amazon Pinpoint API to send voice messages to specific phone numbers. This section contains complete code examples that you can use to send voice messages through the Amazon Pinpoint SMS and Voice API by using an AWS SDK.

Java

Use this example to send a voice message by using the [AWS SDK for Java](#). This example assumes that you've already installed and configured the SDK for Java. For more information, see [Getting Started](#) in the *AWS SDK for Java Developer Guide*.

This example assumes that you're using a shared credentials file to specify the Access Key and Secret Access Key for an existing IAM user. For more information, see [Set up AWS Credentials and Region for Development](#) in the *AWS SDK for Java Developer Guide*.

```
package com.amazonaws.samples;  
  
import java.io.IOException;  
  
import com.amazonaws.services.pinpointsmsvoice.AmazonPinpointSMSVoice;  
import com.amazonaws.services.pinpointsmsvoice.AmazonPinpointSMSVoiceClientBuilder;  
import com.amazonaws.services.pinpointsmsvoice.model.SSMLMessageType;  
import com.amazonaws.services.pinpointsmsvoice.model.SendVoiceMessageRequest;  
import com.amazonaws.services.pinpointsmsvoice.model.VoiceMessageContent;  
  
public class SendMessage {  
  
    // The AWS Region that you want to use to send the voice message. For a list of  
    // AWS Regions where the Amazon Pinpoint SMS and Voice API is available, see  
    // https://docs.aws.amazon.com/pinpoint-sms-voice/latest/APIReference/  
    static final String region = "us-east-1";  
  
    // The phone number that the message is sent from. The phone number that you  
    // specify has to be associated with your Amazon Pinpoint account. For best  
    // results, you should specify the phone number in E.164 format.  
    static final String originationNumber = "+12065550110";  
  
    // The recipient's phone number. For best results, you should specify the  
    // phone number in E.164 format.  
    static final String destinationNumber = "+12065550142";  
  
    // The Amazon Polly voice that you want to use to send the message. For a list  
    // of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html  
    static final String voiceName = "Matthew";  
  
    // The language to use when sending the message. For a list of supported  
    // languages, see https://docs.aws.amazon.com/polly/latest/dg/  
    SupportedLanguage.html  
    static final String languageCode = "en-US";
```

```
// The content of the message. This example uses SSML to customize and control
// certain aspects of the message, such as by adding pauses and changing
// phonation. The message can't contain any line breaks.
static final String ssmlMessage = "<speak>This is a test message sent from "
    + "<emphasis>Amazon Pinpoint</emphasis> "
    + "using the <break strength='weak'>AWS "
    + "SDK for Java. "
    + "<amazon:effect phonation='soft'>Thank "
    + "you for listening.</amazon:effect></speak>";

// The phone number that you want to appear on the recipient's device. The
// phone number that you specify has to be associated with your Amazon Pinpoint
// account.
static final String callerId = "+12065550199";

// The configuration set that you want to use to send the message.
static final String configurationSet = "ConfigSet";

public static void main(String[] args) throws IOException {
    try {
        AmazonPinpointSMSVoice client =
AmazonPinpointSMSVoiceClientBuilder.standard()
            .withRegion(region).build();
        SendVoiceMessageRequest request = new SendVoiceMessageRequest()
            .withCallerId(callerId)
            .withDestinationPhoneNumber(destinationNumber)
            .withOriginationPhoneNumber(originationNumber)
            .withConfigurationSetName(configurationSet)
            .withContent(new VoiceMessageContent()
                .withSSMLMessage(new SSMLMessageType()
                    .withLanguageCode(languageCode)
                    .withVoiceId(voiceName)
                    .withText(ssmlMessage)
                )
            );
        client.sendVoiceMessage(request);
        System.out.println("The message was sent successfully.");
    } catch (Exception ex) {
        System.out.println("The message wasn't sent. Error message: " +
ex.getMessage()
        );
    }
}
}
```

JavaScript (Node.js)

Use this example to send a voice message by using the AWS SDK for JavaScript in Node.js. This example assumes that you've already installed and configured the SDK for JavaScript in Node.js.

This example assumes that you're using a shared credentials file to specify the Access Key and Secret Access Key for an existing IAM user. For more information, see [Setting Credentials](#) in the *AWS SDK for JavaScript in Node.js Developer Guide*.

```
'use strict'

var AWS = require('aws-sdk');

// The AWS Region that you want to use to send the voice message. For a list of
// AWS Regions where the Amazon Pinpoint SMS and Voice API is available, see
// https://docs.aws.amazon.com/pinpoint-sms-voice/latest/APIReference/
var aws_region = "us-east-1";
```

```
// The phone number that the message is sent from. The phone number that you
// specify has to be associated with your Amazon Pinpoint account. For best results,
// you
// should specify the phone number in E.164 format.
var originationNumber = "+12065550110";

// The recipient's phone number. For best results, you should specify the phone
// number in E.164 format.
var destinationNumber = "+12065550142";

// The language to use when sending the message. For a list of supported
// languages, see https://docs.aws.amazon.com/polly/latest/dg/SupportedLanguage.html
var languageCode = "en-US";

// The Amazon Polly voice that you want to use to send the message. For a list
// of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html
var voiceId = "Matthew";

// The content of the message. This example uses SSML to customize and control
// certain aspects of the message, such as the volume or the speech rate.
// The message can't contain any line breaks.
var ssmlMessage = "<speak>"
  + "This is a test message sent from <emphasis>Amazon Pinpoint</emphasis> "
  + "using the <break strength='weak'>AWS SDK for JavaScript in Node.js. "
  + "<amazon:effect phonation='soft'>Thank you for listening."
  + "</amazon:effect>"
  + "</speak>";

// The phone number that you want to appear on the recipient's device. The phone
// number that you specify has to be associated with your Amazon Pinpoint account.
var callerId = "+12065550199";

// The configuration set that you want to use to send the message.
var configurationSet = "ConfigSet";

// Specify that you're using a shared credentials file, and optionally specify
// the profile that you want to use.
var credentials = new AWS.SharedIniFileCredentials({profile: 'default'});
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({region:aws_region});

//Create a new Pinpoint object.
var pinpointSMSvoice = new AWS.PinpointSMSVoice();

var params = {
  CallerId: callerId,
  ConfigurationSetName: configurationSet,
  Content: {
    SSMLMessage: {
      LanguageCode: languageCode,
      Text: ssmlMessage,
      VoiceId: voiceId
    }
  },
  DestinationPhoneNumber: destinationNumber,
  OriginationPhoneNumber: originationNumber
};

//Try to send the message.
pinpointSMSvoice.sendVoiceMessage(params, function(err, data) {
  // If something goes wrong, print an error message.
  if(err) {
    console.log(err.message);
  }
  // Otherwise, show the unique ID for the message.
});
```

```
    } else {  
        console.log("Message sent! Message ID: " + data['MessageId']);  
    }  
});
```

Python

Use this example to send a voice message by using the AWS SDK for Python (Boto 3). This example assumes that you've already installed and configured the SDK for Python (Boto 3).

This example assumes that you're using a shared credentials file to specify the Access Key and Secret Access Key for an existing IAM user. For more information, see [Credentials](#) in the *AWS SDK for Python (Boto 3) API Reference*.

```
import boto3  
from botocore.exceptions import ClientError  
  
# The AWS Region that you want to use to send the voice message. For a list of  
# AWS Regions where the Amazon Pinpoint SMS and Voice API is available, see  
# https://docs.aws.amazon.com/pinpoint-sms-voice/latest/APIReference/  
region = "us-east-1"  
  
# The phone number that the message is sent from. The phone number that you  
# specify has to be associated with your Amazon Pinpoint account. For best results, you  
# should specify the phone number in E.164 format.  
originationNumber = "+12065550110"  
  
# The recipient's phone number. For best results, you should specify the phone  
# number in E.164 format.  
destinationNumber = "+12065550142"  
  
# The language to use when sending the message. For a list of supported  
# languages, see https://docs.aws.amazon.com/polly/latest/dg/SupportedLanguage.html  
languageCode = "en-US"  
  
# The Amazon Polly voice that you want to use to send the message. For a list  
# of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html  
voiceId = "Matthew"  
  
# The content of the message. This example uses SSML to customize and control  
# certain aspects of the message, such as the volume or the speech rate.  
# The message can't contain any line breaks.  
ssmlMessage = ("<speak>"  
               "This is a test message sent from <emphasis>Amazon Pinpoint</emphasis> "  
               "using the <break strength='weak' />AWS SDK for Python. "  
               "<amazon:effect phonation='soft'>Thank you for listening."  
               "</amazon:effect>"  
               "</speak>")  
  
# The phone number that you want to appear on the recipient's device. The phone  
# number that you specify has to be associated with your Amazon Pinpoint account.  
callerId = "+12065550199"  
  
# The configuration set that you want to use to send the message.  
configurationSet = "ConfigSet"  
  
# Create a new SMS and Voice client and specify an AWS Region.  
client = boto3.client('sms-voice', region_name=region)  
  
try:  
    response = client.send_voice_message(  
        DestinationPhoneNumber = destinationNumber,
```

```
    OriginationPhoneNumber = originationNumber,
    CallerId = callerId,
    ConfigurationSetName = configurationSet,
    Content={
      'SSMLMessage':{
        'LanguageCode': languageCode,
        'VoiceId': voiceId,
        'Text': ssmlMessage
      }
    }
  )
# Display an error message if something goes wrong.
except ClientError as e:
    print(e.response['Error']['Message'])
# If the message is sent successfully, show the message ID.
else:
    print("Message sent!"),
    print("Message ID: " + response['MessageId'])
```

Streaming Amazon Pinpoint Events to Kinesis

The 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 Data Firehose or Amazon Kinesis Data 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 Kinesis stream or Kinesis Data 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 Your Application](#) (p. 3).
- The ARN of an Kinesis stream or Kinesis Data Firehose delivery stream in your AWS account. For information about creating these resources, see [Amazon Kinesis Data Streams](#) in the *Amazon Kinesis Data Streams Developer Guide* or [Creating an Amazon Kinesis Data Firehose Delivery Stream](#) in the *Amazon Kinesis Data 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 Kinesis](#) (p. 134).

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 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 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 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 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 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 Kinesis stream.

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

App Events

After you integrate your app with Amazon Pinpoint and you run one or more campaigns, Amazon Pinpoint streams events about user activity and campaign message deliveries.

Example

The JSON object for an app 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": "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 App Event Types

Amazon Pinpoint streams the following standard types for app events:

- `_campaign.send`
- `_monetization.purchase`

- `_session.start`
- `_session.stop`
- `_session.pause`
- `_session.resume`
- `_userauth.sign_in`
- `_userauth.sign_up`
- `_userauth.auth_fail`

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": "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 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.
<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> .

Attribute	Description
<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.

Logging Amazon Pinpoint API Calls with AWS CloudTrail

Amazon Pinpoint is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Amazon Pinpoint. CloudTrail captures API calls for Amazon Pinpoint as events. The calls captured include calls from the Amazon Pinpoint console and code calls to the Amazon Pinpoint API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon Pinpoint. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to Amazon Pinpoint, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, including how to configure and enable it, see the [AWS CloudTrail User Guide](#).

Amazon Pinpoint Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When supported event activity occurs in Amazon Pinpoint, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see [Viewing Events with CloudTrail Event History](#).

For an ongoing record of events in your AWS account, including events for Amazon Pinpoint, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- [Overview for Creating a Trail](#)
- [CloudTrail Supported Services and Integrations](#)
- [Configuring Amazon SNS Notifications for CloudTrail](#)
- [Receiving CloudTrail Log Files from Multiple Regions](#) and [Receiving CloudTrail Log Files from Multiple Accounts](#)

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the [CloudTrail userIdentity Element](#).

You can create a trail and store your log files in your Amazon S3 bucket for as long as you want, and define Amazon S3 lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted with Amazon S3 server-side encryption (SSE).

To be notified of log file delivery, configure CloudTrail to publish Amazon SNS notifications when new log files are delivered. For more information, see [Configuring Amazon SNS Notifications for CloudTrail](#).

You can also aggregate Amazon Pinpoint log files from multiple AWS Regions and multiple AWS accounts into a single Amazon S3 bucket.

For more information, see [Receiving CloudTrail Log Files from Multiple Regions](#) and [Receiving CloudTrail Log Files from Multiple Accounts](#).

You can use CloudTrail to log actions for the following Amazon Pinpoint APIs:

- [Amazon Pinpoint API \(p. 107\)](#)
- [Amazon Pinpoint Email API \(p. 108\)](#)
- [Amazon Pinpoint SMS and Voice API \(p. 109\)](#)

Amazon Pinpoint API Actions That Can be Logged by CloudTrail

Amazon Pinpoint supports logging the following actions as events in CloudTrail log files:

- [CreateApp](#)
- [CreateCampaign](#)
- [CreateImportJob](#)
- [CreateSegment](#)
- [DeleteAdmChannel](#)
- [DeleteApnsChannel](#)
- [DeleteApnsSandboxChannel](#)
- [DeleteApnsVoipChannel](#)
- [DeleteApnsVoipSandboxChannel](#)
- [DeleteApp](#)
- [DeleteBaiduChannel](#)
- [DeleteCampaign](#)
- [DeleteEmailChannel](#)
- [DeleteEventStream](#)
- [DeleteGcmChannel](#)
- [DeleteSegment](#)
- [DeleteSmsChannel](#)
- [GetAdmChannel](#)
- [GetApnsChannel](#)
- [GetApnsSandboxChannel](#)
- [GetApnsVoipChannel](#)
- [GetApnsVoipSandboxChannel](#)
- [GetApp](#)
- [GetApplicationSettings](#)
- [GetApps](#)
- [GetBaiduChannel](#)
- [GetCampaign](#)

- [GetCampaignActivities](#)
- [GetCampaignVersion](#)
- [GetCampaignVersions](#)
- [GetCampaigns](#)
- [GetEmailChannel](#)
- [GetEventStream](#)
- [GetGcmChannel](#)
- [GetImportJob](#)
- [GetImportJobs](#)
- [GetSegment](#)
- [GetSegmentImportJobs](#)
- [GetSegmentVersion](#)
- [GetSegmentVersions](#)
- [GetSegments](#)
- [GetSmsChannel](#)
- [PutEventStream](#)
- [UpdateAdmChannel](#)
- [UpdateApnsChannel](#)
- [UpdateApnsSandboxChannel](#)
- [UpdateApnsVoipChannel](#)
- [UpdateApnsVoipSandboxChannel](#)
- [UpdateApplicationSettings](#)
- [UpdateBaiduChannel](#)
- [UpdateCampaign](#)
- [UpdateEmailChannel](#)
- [UpdateGcmChannel](#)
- [UpdateSegment](#)
- [UpdateSmsChannel](#)

Note

The following Amazon Pinpoint actions **aren't** logged in CloudTrail:

- [GetEndpoint](#)
- [SendMessage](#)
- [SendUsersMessages](#)
- [UpdateEndpoint](#)
- [UpdateEndpointsBatch](#)

Amazon Pinpoint SMS Email Actions That Can be Logged by CloudTrail

The Amazon Pinpoint Email API supports logging the following actions as events in CloudTrail log files:

- [CreateConfigurationSet](#)
- [CreateConfigurationSetEventDestination](#)

- [CreateDedicatedIpPool](#)
- [CreateEmailIdentity](#)
- [DeleteConfigurationSet](#)
- [DeleteConfigurationSetEventDestination](#)
- [DeleteDedicatedIpPool](#)
- [DeleteEmailIdentity](#)
- [GetAccount](#)
- [GetConfigurationSet](#)
- [GetConfigurationSetEventDestinations](#)
- [GetDedicatedIp](#)
- [GetDedicatedIps](#)
- [GetEmailIdentity](#)
- [ListConfigurationSets](#)
- [ListDedicatedIpPools](#)
- [ListEmailIdentities](#)
- [PutAccountDedicatedIpWarmupAttributes](#)
- [PutAccountSendingAttributes](#)
- [PutConfigurationSetDeliveryOptions](#)
- [PutConfigurationSetReputationOptions](#)
- [PutConfigurationSetSendingOptions](#)
- [PutConfigurationSetTrackingOptions](#)
- [PutDedicatedIpInPool](#)
- [PutDedicatedIpWarmupAttributes](#)
- [PutEmailIdentityDkimAttributes](#)
- [PutEmailIdentityFeedbackAttributes](#)
- [PutEmailIdentityMailFromAttributes](#)
- [UpdateConfigurationSetEventDestination](#)

Note

The following Amazon Pinpoint Email API action **isn't** logged in CloudTrail:

- [SendEmail](#)

Amazon Pinpoint SMS and Voice API Actions That Can be Logged by CloudTrail

The Amazon Pinpoint SMS and Voice API supports logging the following actions as events in CloudTrail log files:

- [CreateConfigurationSet](#)
- [DeleteConfigurationSet](#)
- [GetConfigurationSetEventDestinations](#)
- [CreateConfigurationSetEventDestination](#)
- [UpdateConfigurationSetEventDestination](#)
- [DeleteConfigurationSetEventDestination](#)

Note

The following Amazon Pinpoint SMS and Voice API action **isn't** logged in CloudTrail:

- SendVoiceMessage

Example: Amazon Pinpoint Log File Entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the GetCampaigns and CreateCampaign actions of the Amazon Pinpoint API.

```
{
  "Records": [
    {
      "awsRegion": "us-east-1",
      "eventID": "example0-09a3-47d6-a810-c5f9fd2534fe",
      "eventName": "GetCampaigns",
      "eventSource": "pinpoint.amazonaws.com",
      "eventTime": "2018-02-03T00:56:48Z",
      "eventType": "AwsApiCall",
      "eventVersion": "1.05",
      "readOnly": true,
      "recipientAccountId": "123456789012",
      "requestID": "example1-b9bb-50fa-abdb-80f274981d60",
      "requestParameters": {
        "application-id": "example71dfa4c1aab66332a5839798f",
        "page-size": "1000"
      },
      "responseElements": null,
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "Jersey/${project.version} (URLConnection 1.8.0_144)",
      "userIdentity": {
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "accountId": "123456789012",
        "arn": "arn:aws:iam:123456789012:root",
        "principalId": "123456789012",
        "sessionContext": {
          "attributes": {
            "creationDate": "2018-02-02T16:55:29Z",
            "mfaAuthenticated": "false"
          }
        }
      },
      "type": "Root"
    },
    {
      "awsRegion": "us-east-1",
      "eventID": "example0-09a3-47d6-a810-c5f9fd2534fe",
      "eventName": "CreateCampaign",
      "eventSource": "pinpoint.amazonaws.com",
      "eventTime": "2018-02-03T01:05:16Z",
      "eventType": "AwsApiCall",
      "eventVersion": "1.05",
      "readOnly": false,
      "recipientAccountId": "123456789012",
      "requestID": "example1-b9bb-50fa-abdb-80f274981d60",
    }
  ]
}
```

```
"requestParameters": {
  "Description": "****",
  "HoldoutPercent": 0,
  "IsPaused": false,
  "MessageConfiguration": "****",
  "Name": "****",
  "Schedule": {
    "Frequency": "ONCE",
    "IsLocalTime": true,
    "StartTime": "2018-02-03T00:00:00-08:00",
    "Timezone": "utc-08"
  },
  "SegmentId": "exampleda204adf991a80281aa0e591",
  "SegmentVersion": 1,
  "application-id": "example71dfa4c1aab66332a5839798f"
},
"responseElements": {
  "ApplicationId": "example71dfa4c1aab66332a5839798f",
  "CreationDate": "2018-02-03T01:05:16.425Z",
  "Description": "****",
  "HoldoutPercent": 0,
  "Id": "example54a654f80948680cbba240ede",
  "IsPaused": false,
  "LastModifiedDate": "2018-02-03T01:05:16.425Z",
  "MessageConfiguration": "****",
  "Name": "****",
  "Schedule": {
    "Frequency": "ONCE",
    "IsLocalTime": true,
    "StartTime": "2018-02-03T00:00:00-08:00",
    "Timezone": "utc-08"
  },
  "SegmentId": "example4da204adf991a80281example",
  "SegmentVersion": 1,
  "State": {
    "CampaignStatus": "SCHEDULED"
  },
  "Version": 1
},
"sourceIPAddress": "192.0.2.0",
"userAgent": "aws-cli/1.14.9 Python/3.4.3 Linux/3.4.0+ botocore/1.8.34",
"userIdentity": {
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "accountId": "123456789012",
  "arn": "arn:aws:iam:123456789012:user/userName",
  "principalId": "AIDAIHTRCDA62EXAMPLE",
  "type": "IAMUser",
  "userName": "userName"
}
}
]
```

The following example shows a CloudTrail log entry that demonstrates the `CreateConfigurationSet` and `CreateConfigurationSetEventDestination` actions in the Amazon Pinpoint SMS and Voice API.

```
{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAIHTRCDA62EXAMPLE",

```

Amazon Pinpoint Developer Guide
Example: Amazon Pinpoint Log File Entries

```
    "arn": "arn:aws:iam:111122223333:user/SampleUser",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "SampleUser"
  },
  "eventTime": "2018-11-06T21:45:55Z",
  "eventSource": "sms-voice.amazonaws.com",
  "eventName": "CreateConfigurationSet",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.0.1",
  "userAgent": "PostmanRuntime/7.3.0",
  "requestParameters": {
    "ConfigurationSetName": "MyConfigurationSet"
  },
  "responseElements": null,
  "requestID": "56dcc091-e20d-11e8-87d2-9994aexample",
  "eventID": "725843fc-8846-41f4-871a-7c52dexample",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
},
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAIHTRCDA62EXAMPLE",
    "arn": "arn:aws:iam:111122223333:user/SampleUser",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "SampleUser"
  },
  "eventTime": "2018-11-06T21:47:08Z",
  "eventSource": "sms-voice.amazonaws.com",
  "eventName": "CreateConfigurationSetEventDestination",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.0.1",
  "userAgent": "PostmanRuntime/7.3.0",
  "requestParameters": {
    "EventDestinationName": "CloudWatchEventDestination",
    "ConfigurationSetName": "MyConfigurationSet",
    "EventDestination": {
      "Enabled": true,
      "MatchingEventTypes": [
        "INITIATED_CALL",
        "INITIATED_CALL"
      ],
      "CloudWatchLogsDestination": {
        "IamRoleArn": "arn:aws:iam:111122223333:role/iamrole-01",
        "LogGroupArn": "arn:aws:logs:us-east-1:111122223333:log-group:clientloggroup-01"
      }
    }
  }
},
  "responseElements": null,
  "requestID": "81de1e73-e20d-11e8-b158-d5536example",
  "eventID": "fcafc21f-7c93-4a3f-9e72-fca2dexample",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
]
}
```

Deleting Data from Amazon Pinpoint

Depending on how you use it, Amazon Pinpoint might store certain data that could be considered personal. For example, each endpoint in Amazon Pinpoint contains contact information for an end user, such as that person's email address or mobile phone number.

You can use the Amazon Pinpoint API to permanently delete personal data. This section includes procedures for deleting various types of data that could be considered personal.

Deleting Endpoints

An endpoint represents a single method of contacting one of your customers. Each endpoint can refer to a customer's email address, mobile device identifier, or email address. In many jurisdictions, this type of information might be considered personal.

The procedure in this section uses the AWS CLI to interact with the Amazon Pinpoint API. This procedure assumes that you've already installed and configured the AWS CLI. For more information, see [Installing the AWS Command Line Interface](#) in the *AWS Command Line Interface User Guide*.

To delete an endpoint by using the AWS CLI

- At the command line, type the following command:

```
aws pinpoint delete-endpoint --application-id example1884c7d659a2feaa0c5 --endpoint-id ad015a3bf4f1b2b0b82example
```

In the preceding command, replace *example1884c7d659a2feaa0c5* with the ID of the application or project that the endpoint is located in. Also, replace *ad015a3bf4f1b2b0b82example* with the unique ID of the endpoint itself.

Deleting Segment and Endpoint Data Stored in Amazon S3

You can import segments from a file that's stored in an Amazon S3 bucket by using the Amazon Pinpoint console or the API. You can also export application, segment, or endpoint data from Amazon Pinpoint to an Amazon S3 bucket. Both the imported and exported files can contain personal data, including email addresses, mobile phone numbers, and information about the physical location of the endpoint.

Content delivered to Amazon S3 buckets might contain customer content. For more information about removing sensitive data, see [How Do I Empty an S3 Bucket?](#) or [How Do I Delete an S3 Bucket?](#)

Delete All AWS Data by Closing Your AWS Account

It's also possible to delete all data that you've stored in Amazon Pinpoint by closing your AWS account. However, this action also deletes all other data—personal or non-personal—that you've stored in every other AWS service.

When you close your AWS account, we retain the data in your AWS account for 90 days. At the end of this retention period, we delete this data permanently and irreversibly.

Warning

The following procedure completely removes all data that's stored in your AWS account across all AWS services and regions.

You can close your AWS account by using the AWS Management Console.

To close your AWS account

1. Open the AWS Management Console at <https://console.aws.amazon.com>.
2. Go to the **Account Settings** page at <https://console.aws.amazon.com/billing/home?#/account>.

Warning

The following two steps permanently delete all of the data you've stored in all AWS services across all AWS Regions.

3. Under **Close Account**, read the disclaimer that describes the consequences of closing your AWS account. If you agree to the terms, select the check box, and then choose **Close Account**.
4. On the confirmation dialog box, choose **Close Account**.

Permissions

To use Amazon Pinpoint, users in your AWS account require permissions that allow them to view analytics data, define user segments, create and deploy campaigns, and more. Users of your app also 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 \(p. 115\)](#).

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 \(p. 117\)](#) 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*.

To import endpoint definitions, you must grant Amazon Pinpoint [read-only access to an Amazon S3 bucket \(p. 130\)](#).

Topics

- [IAM Policies for Amazon Pinpoint Users \(p. 115\)](#)
- [IAM Role for Importing Endpoints or Segments \(p. 130\)](#)
- [IAM Role for Exporting Endpoints or Segments \(p. 131\)](#)
- [IAM Role for Streaming Events to Kinesis \(p. 134\)](#)

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/project-id/segments` URI. This policy also allows the user to view the segments for a project 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/project-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 API Actions

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:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Read-Only Access

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

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

In the preceding policy example, replace *accountId* with your AWS Account ID.

You can also create a policy that allows read-only access to a specific Amazon Pinpoint project. To do this, specify an AWS Region and a project ID, as shown in the following example:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "mobiletargeting:Get*"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:mobiletargeting:region:account-id:apps/project-id"
      ]
    }
  ]
}
```

```
}
```

In the preceding policy example, replace *region* with the name of the AWS Region you're using, *account-id* with your AWS account ID, and *project-id* with the unique ID of your Amazon Pinpoint project.

Amazon Pinpoint SMS and Voice API Actions

Admin Access

The following policy grants full access to the Amazon Pinpoint SMS and Voice API:

```
{
  "Version": "2018-09-05",
  "Statement": [
    {
      "Action": [
        "sms-voice:*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Read-Only Access

The following policy allows read-only access to the Amazon Pinpoint SMS and Voice API:

```
{
  "Version": "2018-09-05",
  "Statement": [
    {
      "Action": [
        "sms-voice:Get*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Amazon Pinpoint API Actions

This section contains API actions that you can add to the IAM policies in your AWS account. By adding these policies to an IAM user account, you can specify which Amazon Pinpoint features that user is allowed to perform.

Categories:

- [Campaigns \(p. 118\)](#)
- [Channels \(p. 119\)](#)
- [Endpoints \(p. 123\)](#)
- [Event Streams \(p. 123\)](#)
- [Export Jobs \(p. 124\)](#)
- [Import Jobs \(p. 124\)](#)
- [Messages \(p. 125\)](#)

- [Phone Number Validate](#) (p. 125)
- [Projects](#) (p. 126)
- [Reports](#) (p. 126)
- [Segments](#) (p. 127)
- [Users](#) (p. 128)

Campaigns

The following permissions are related to managing campaigns in your Amazon Pinpoint account.

mobiletargeting:CreateCampaign

Create a campaign for a project.

- URI – [/apps/project-id/campaigns](#)
- Method – POST
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/campaigns`

mobiletargeting>DeleteCampaign

Delete a specific campaign.

- URI – [/apps/project-id/campaigns/campaign-id](#)
- Method – DELETE
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/campaigns/campaign-id`

mobiletargeting:GetCampaign

Retrieve information about a specific campaign.

- URI – [/apps/project-id/campaigns/campaign-id](#)
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/campaigns/campaign-id`

mobiletargeting:GetCampaignActivities

Retrieve information about the activities performed by a campaign.

- URI – [/apps/project-id/campaigns/campaign-id/activities](#)
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/campaigns/campaign-id`

mobiletargeting:GetCampaigns

Retrieve information about all campaigns for a project.

- URI – [/apps/project-id/campaigns](#)
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id`

mobiletargeting:GetCampaignVersion

Retrieve information about a specific campaign version.

- URI – [/apps/project-id/campaigns/campaign-id/versions/version-id](#)
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/campaigns/campaign-id`

mobiletargeting:GetCampaignVersions

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

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

mobiletargeting:UpdateCampaign

Update a specific campaign.

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

Channels

The following permissions are related to managing channels in your Amazon Pinpoint account. In Amazon Pinpoint, *channels* refer to the methods you use to contact your customers, such as by sending email, SMS messages, or push notifications.

mobiletargeting>DeleteAdmChannel

Delete the Amazon Device Messaging (ADM) channel for a project.

- URI – `/apps/project-id/channels/adm`
- Method – DELETE
- Resource ARN – `arn:aws:mobiletargeting:region:accountid:apps/project-id/channels/adm`

mobiletargeting:GetAdmChannel

Retrieve information about the Amazon Device Messaging (ADM) channel for a project.

- URI – `/apps/project-id/channels/adm`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:accountid:apps/project-id/channels/adm`

mobiletargeting:UpdateAdmChannel

Update the Amazon Device Messaging (ADM) channel for a project.

- URI – `/apps/project-id/channels/adm`
- Method – PUT
- Resource ARN – `arn:aws:mobiletargeting:region:accountid:apps/project-id/channels/adm`

mobiletargeting>DeleteApnsChannel

Delete the APNs channel for a project.

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

mobiletargeting:GetApnsChannel

Retrieve information about the APNs channel for a project.

- URI – `/apps/project-id/channels/apns`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-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/project-id/channels/apns`
- Method – PUT
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/channels/apns`

mobiletargeting>DeleteApnsSandboxChannel

Delete the APNs sandbox channel for a project.

- URI – `/apps/project-id/channels/apns_sandbox`
- Method – DELETE
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/channels/apns_sandbox`

mobiletargeting:GetApnsSandboxChannel

Retrieve information about the APNs sandbox channel for a project.

- URI – `/apps/project-id/channels/apns_sandbox`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/channels/apns_sandbox`

mobiletargeting:UpdateApnsSandboxChannel

Update the APNs sandbox channel for a project.

- URI – `/apps/project-id/channels/apns_sandbox`
- Method – PUT
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/channels/apns_sandbox`

mobiletargeting>DeleteApnsVoipChannel

Delete the APNs VoIP channel for a project.

- URI – `/apps/project-id/channels/apns_voip`
- Method – DELETE
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/channels/apns_voip`

mobiletargeting:GetApnsVoipChannel

Retrieve information about the APNs VoIP channel for a project.

- URI – `/apps/project-id/channels/apns_voip`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/channels/apns_voip`

mobiletargeting:UpdateApnsVoipChannel

Update the APNs VoIP channel for a project.

- URI – `/apps/project-id/channels/apns_voip`

- Method – PUT
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/channels/apns_voip`

mobiletargeting:DeleteApnsVoipChannel

Delete the APNs VoIP sandbox channel for a project.

- URI – `/apps/project-id/channels/apns_voip_sandbox`
- Method – DELETE
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/channels/apns_voip_sandbox`

mobiletargeting:GetApnsVoipChannel

Retrieve information about the APNs VoIP sandbox channel for a project.

- URI – `/apps/project-id/channels/apns_voip_sandbox`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/channels/apns_voip_sandbox`

mobiletargeting:UpdateApnsVoipChannel

Update the APNs VoIP sandbox channel for a project.

- URI – `/apps/project-id/channels/apns_voip_sandbox`
- Method – PUT
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/channels/apns_voip_sandbox`

mobiletargeting>DeleteBaiduChannel

Delete the Baidu channel for a project.

- URI – `/apps/project-id/channels/baidu`
- Method – DELETE
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/channels/baidu`

mobiletargeting:GetBaiduChannel

Retrieve information about the Baidu channel for a project.

- URI – `/apps/project-id/channels/baidu`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/channels/baidu`

mobiletargeting:UpdateBaiduChannel

Update the Baidu channel for a project.

- URI – `/apps/project-id/channels/baidu`
- Method – PUT
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/channels/baidu`

mobiletargeting>DeleteEmailChannel

Delete the email channel in a project.

- URI – `/apps/project-id/channels/email`
- Method – DELETE

- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/channels/email`

mobiletargeting:GetEmailChannel

Obtain information about the email channel in a project.

- URI – `/apps/project-id/channels/email`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/channels/email`

mobiletargeting:UpdateEmailChannel

Update the email channel in a project.

- URI – `/apps/project-id/channels/email`
- Method – PUT
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/channels/email`

mobiletargeting>DeleteGcmChannel

Delete the GCM channel for a project.

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

mobiletargeting:GetGcmChannel

Retrieve information about the GCM channel for a project.

- URI – `/apps/project-id/channels/gcm`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-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/project-id/channels/gcm`
- Method – PUT
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/channels/gcm`

mobiletargeting>DeleteSmsChannel

Delete the SMS channel in a project.

- URI – `/apps/project-id/channels/sms`
- Method – DELETE
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/channels/sms`

mobiletargeting:GetSmsChannel

Obtain information about the SMS channel in a project.

- URI – `/apps/project-id/channels/sms`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/channels/sms`

mobiletargeting:UpdateSmsChannel

Update the SMS channel in a project.

- URI – `/apps/project-id/channels/sms`
- Method – PUT
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/channels/sms`

Endpoints

The following permissions are related to managing endpoints in your Amazon Pinpoint account. In Amazon Pinpoint, an *endpoint* is a single destination for your messages. For example, an endpoint could be a customer's email address, telephone number, or mobile device token.

mobiletargeting>DeleteEndpoint

Delete an endpoint.

- URI – `/apps/project-id/endpoints/endpoint-id`
- Method – DELETE
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/endpoints/endpoint-id`

mobiletargeting:GetEndpoint

Retrieve information about a specific endpoint.

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

mobiletargeting:UpdateEndpoint

Create an endpoint or update the information for an endpoint.

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

mobiletargeting:UpdateEndpointsBatch

Create or update endpoints as a batch operation.

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

Event Streams

The following permissions are related to managing campaigns in your Amazon Pinpoint account.

mobiletargeting>DeleteEventStream

Delete the event stream for a project.

- URI – `/apps/project-id/eventstream/`
- Method – DELETE

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

mobiletargeting:GetEventStream

Retrieve information about the event stream for a project.

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

mobiletargeting:PutEventStream

Create or update an event stream for a project.

- URI – `/apps/project-id/eventstream/`
- Method – PUT
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/eventstream`

Export Jobs

The following permissions are related to managing export jobs in your Amazon Pinpoint account. In Amazon Pinpoint, you create *export jobs* to send information about endpoints to an Amazon S3 bucket for storage or analysis.

mobiletargeting:CreateExportJobs

Create an export job for exporting endpoint definitions to Amazon S3.

- URI – `/apps/project-id/jobs/export`
- Method – POST
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/jobs/export`

mobiletargeting:GetExportJob

Obtain information a specific export job.

- URI – `/apps/project-id/jobs/export/job-id`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/jobs/export/job-id`

mobiletargeting:GetExportJobs

Retrieve a list of all of the export jobs for a project.

- URI – `/apps/project-id/jobs/export`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/jobs/export`

Import Jobs

The following permissions are related to managing import jobs in your Amazon Pinpoint account. In Amazon Pinpoint, you create *import jobs* to create segments based on endpoint definitions stored in an Amazon S3 bucket.

mobiletargeting:CreateImportJob

Import endpoint definitions from Amazon S3 to create a segment.

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

mobiletargeting:GetImportJob

Retrieve information about a specific import job.

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

mobiletargeting:GetImportJobs

Retrieve information about all import jobs for a project.

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

Messages

The following permissions are related to sending SMS messages and push notifications from your Amazon Pinpoint account. You can use the `SendMessage`s and `SendUsersMessages` operations to send messages to specific endpoints without creating segments and campaigns first.

mobiletargeting:SendMessage

Send an SMS message or push notification to specific endpoints.

- URI – `/apps/project-id/messages`
- Method – POST
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/messages`

mobiletargeting:SendUsersMessages

Send an SMS message or push notification to all endpoints that are associated a specific user ID.

- URI – `/apps/project-id/users-messages`
- Method – POST
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/users-messages`

Phone Number Validate

The following permissions are related to using the Phone Number Validate feature in Amazon Pinpoint.

mobiletargeting:PhoneNumberValidate

Obtain information about a phone number.

- URI – `/phone/number/validate`
- Method – POST
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:phone/number/validate`

Projects

The following permissions are related to managing projects in your Amazon Pinpoint account. Originally, projects were referred to as *applications*. For the purposes of these operations, a Amazon Pinpoint application is the same as a Amazon Pinpoint project.

mobiletargeting:CreateApp

Create a project.

- URI – [/apps](#)
- Method – POST
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps`

mobiletargeting>DeleteApp

Delete a project.

- URI – [/apps/project-id](#)
- Method – DELETE
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id`

mobiletargeting:GetApp

Retrieve information about a specific project in your Amazon Pinpoint account.

- URI – [/apps/project-id](#)
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id`

mobiletargeting:GetApps

Retrieve a list of projects in your Amazon Pinpoint account.

- URI – [/apps](#)
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps`

mobiletargeting:GetApplicationSettings

Retrieve the default settings for an Amazon Pinpoint project.

- URI – [/apps/project-id/settings](#)
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id`

mobiletargeting:UpdateApplicationSettings

Retrieve the default settings for an Amazon Pinpoint project.

- URI – [/apps/project-id/settings](#)
- Method – PUT
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id`

Reports

The following permission is related to retrieving reports and metrics related to your Amazon Pinpoint account.

mobiletargeting:GetReports

View analytics in the Amazon Pinpoint console.

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

Segments

The following permissions are related to managing segments in your Amazon Pinpoint account. In Amazon Pinpoint, *segments* are groups of recipients for your campaigns that share certain attributes that you define.

mobiletargeting:CreateSegment

Create a segment. To allow a user to create a segment by importing endpoint data from outside of Amazon Pinpoint, allow the `mobiletargeting:CreateImportJob` action.

- URI – `/apps/project-id/segments`
- Method – POST
- Resource ARN – arn:aws:mobiletargeting:*region*:*account-id*:apps/*project-id*

mobiletargeting>DeleteSegment

Delete a segment.

- URI – `/apps/project-id/segments/segment-id`
- Method – DELETE
- Resource ARN – arn:aws:mobiletargeting:*region*:*account-id*:apps/*project-id*/segments/*segment-id*

mobiletargeting:GetSegment

Retrieve information about a specific segment.

- URI – `/apps/project-id/segments/segment-id`
- Method – GET
- Resource ARN – arn:aws:mobiletargeting:*region*:*account-id*:apps/*project-id*/segments/*segment-id*

mobiletargeting:GetSegmentExportJobs

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

- URI – `/apps/project-id/segments/segment-id/jobs/export`
- Method – GET
- Resource ARN – arn:aws:mobiletargeting:*region*:*account-id*:apps/*project-id*/segments/*segment-id*/jobs/export

mobiletargeting:GetSegments

Retrieve information about the segments for a project.

- URI – `/apps/project-id/segments`
- Method – GET
- Resource ARN – arn:aws:mobiletargeting:*region*:*account-id*:apps/*project-id*

mobiletargeting:GetSegmentImportJobs

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

- URI – `/apps/project-id/segments/segment-id/jobs/import`
- Method – GET

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

mobiletargeting:GetSegmentVersion

Retrieve information about a specific segment version.

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

mobiletargeting:GetSegmentVersions

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

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

mobiletargeting:UpdateSegment

Update a specific segment.

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

Users

The following permissions are related to managing users in your Amazon Pinpoint account. In Amazon Pinpoint, *users* correspond to individuals who receive messages from you. A single user might be associated with more than one endpoint.

mobiletargeting>DeleteUser

Delete all of the endpoints that are associated with a user ID.

- URI – `/apps/project-id/users/user-id`
- Method – DELETE
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/users/user-id`

mobiletargeting:GetUser

Retrieve information about the endpoints that are associated with a user ID.

- URI – `/apps/project-id/users/user-id`
- Method – GET
- Resource ARN – `arn:aws:mobiletargeting:region:account-id:apps/project-id/users/user-id`

Amazon Pinpoint SMS and Voice API Actions

This section contains API actions that you can add to the IAM policies in your AWS account. By adding these policies to an IAM user account, you can specify which features of the Amazon Pinpoint SMS and Voice a user is allowed to use.

To learn more about the Amazon Pinpoint SMS and Voice API, see [the Amazon Pinpoint SMS and Voice API Reference](#).

sms-voice:CreateConfigurationSet

Create a configuration set for sending voice messages.

- URI – /sms-voice/configuration-sets
- Method – POST
- Resource ARN – not available; use *

sms-voice>DeleteConfigurationSet

Delete a voice message configuration set.

- URI – /sms-voice/configuration-sets/*ConfigurationSetName*
- Method – DELETE
- Resource ARN – not available; use *

sms-voice:GetConfigurationSetEventDestinations

Get information about a configuration set and the event destinations that it contains.

- URI – /sms-voice/configuration-sets/*ConfigurationSetName*/event-destinations
- Method – GET
- Resource ARN – not available; use *

sms-voice:CreateConfigurationSetEventDestination

Create an event destination for voice events.

- URI – /sms-voice/configuration-sets/*ConfigurationSetName*/event-destinations
- Method – POST
- Resource ARN – not available; use *

sms-voice:UpdateConfigurationSetEventDestination

Update an event destination for voice events.

- URI – /sms-voice/configuration-sets/*ConfigurationSetName*/event-destinations/*EventDestinationName*
- Method – PUT
- Resource ARN – not available; use *

sms-voice>DeleteConfigurationSetEventDestination

Delete an event destination for voice events.

- URI – /sms-voice/configuration-sets/*ConfigurationSetName*/event-destinations/*EventDestinationName*
- Method – DELETE
- Resource ARN – not available; use *

sms-voice:SendVoiceMessage

Create and send voice messages.

- URI – /sms-voice/voice/message
- Method – POST
- Resource ARN – not available; use *

IAM Role for Importing Endpoints or 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 \(p. 51\)](#). 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.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 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",  
          "Principal": {  
            "Service": "pinpoint.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 Exporting Endpoints or Segments

You can obtain a list of endpoints by creating an export job. When you create an export job, you have to specify a project ID, and you can optionally specify a segment ID. Amazon Pinpoint then exports a list of the endpoints associated with the project or segment to an Amazon Simple Storage Service (Amazon S3) bucket. The resulting file contains a JSON-formatted list of endpoints and their attributes (such as channel, address, opt-in/opt-out status, creation date, and endpoint ID).

To create an export job, you have to configure an IAM role that allows Amazon Pinpoint to write to an Amazon S3 bucket. The process of configuring the role consists of two steps:

1. Create an IAM policy that allows an entity (in this case, Amazon Pinpoint) to write to a specific Amazon S3 bucket.
2. Create an IAM role and attach the policy to it.

This section contains procedures for completing both of these steps. These procedures assume that you've already created an Amazon S3 bucket, as well as a folder within that bucket, for storing exported segments. For more information about creating buckets, see [Create a Bucket](#) in the *Amazon Simple Storage Service Getting Started Guide*.

These procedures also assume that you've already installed and configured the AWS CLI. For more information about setting up the AWS CLI, see [Installing the AWS Command Line Interface](#) in the *AWS Command Line Interface User Guide*.

Step 1: Create the IAM Policy

An IAM policy defines the permissions for an entity, such as an identity or resource. To create a role for exporting Amazon Pinpoint endpoints, you have to create a policy that grants permission to write to a specific folder in a specific Amazon S3 bucket. The following policy example follows the security practice of granting least privilege—that is, it only grants the permissions that are required to perform a single task.

To create the IAM policy

1. In a text editor, create a new file. Paste the following code into the file:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUserToSeeBucketListInTheConsole",
      "Action": [
        "s3:ListAllMyBuckets",
        "s3:GetBucketLocation"
      ],
      "Effect": "Allow",
      "Resource": [ "arn:aws:s3:*:*" ]
    },
    {
      "Sid": "AllowRootAndHomeListingOfBucket",
      "Action": [
        "s3:ListBucket"
      ],
      "Effect": "Allow",
      "Resource": [ "arn:aws:s3:::example-bucket" ],
      "Condition": {
        "StringEquals": {
          "s3:delimiter": [ "/" ],
          "s3:prefix": [
            "",
            "Exports/"
          ]
        }
      }
    },
    {
      "Sid": "AllowListingOfUserFolder",
      "Action": [
        "s3:ListBucket"
      ],
      "Effect": "Allow",
      "Resource": [ "arn:aws:s3:::example-bucket" ],
      "Condition": {
        "StringLike": {
          "s3:prefix": [
            "Exports/*"
          ]
        }
      }
    },
    {
      "Sid": "AllowAllS3ActionsInUserFolder",
      "Action": [ "s3:*" ],
```

```
        "Effect": "Allow",
        "Resource": [ "arn:aws:s3::example-bucket/Exports/*" ]
    }
  ]
}
```

In the preceding code, replace all instances of `example-bucket` with the name of the Amazon S3 bucket that contains the folder that you want to export the segment information into. Also, replace all instances of `Example` with the name of the folder itself.

When you finish, save the file as `s3policy.json`.

2. At the command line, navigate to the directory where `s3policy.json` is located. Then type the following command to create the policy:

```
aws iam create-policy --policy-name s3ExportPolicy --policy-document
file://s3policy.json
```

If the policy was created successfully, you see output similar to the following:

```
{
  "Policy": {
    "CreateDate": "2018-04-11T18:44:34.805Z",
    "IsAttachable": true,
    "DefaultVersionId": "v1",
    "AttachmentCount": 0,
    "PolicyId": "ANPAJ2YJQRJCG3EXAMPLE",
    "UpdateDate": "2018-04-11T18:44:34.805Z",
    "Arn": "arn:aws:iam::123456789012:policy/s3ExportPolicy",
    "PolicyName": "s3ExportPolicy",
    "Path": "/"
  }
}
```

Copy the Amazon Resource Name (ARN) of the policy (`arn:aws:iam::123456789012:policy/s3ExportPolicy` in the preceding example). In the next section, you must supply this ARN when you create the role.

Note

If you see a message stating that your account isn't authorized to perform the `CreatePolicy` operation, then you need to attach a policy to your user account that lets you create new IAM policies and roles. For more information, see [Attaching IAM Policies \(Console\)](#) in the *IAM User Guide*.

Step 2: Create the IAM Role

Now that you've created an IAM policy, you can create a role and attach the policy to it. Each IAM role contains a *trust policy*—a set of rules that specifies which entities are allowed to assume the role. In this section, you create a trust policy that allows Amazon Pinpoint to assume the role. Next, you create the role itself, and then attach the policy that you created in the previous section.

To create the IAM role

1. In a text editor, create a new file. Paste the following code into the file:

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

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "pinpoint.amazonaws.com"
  },
  "Action": "sts:AssumeRole"
}
]
```

Save the file as `trustpolicy.json`.

2. At the command line, navigate to the directory where `trustpolicy.json` is located. Then type the following command to create a new role:

```
aws iam create-role --role-name s3ExportRole --assume-role-policy-document
file://trustpolicy.json
```

If the command runs successfully, you see output similar to the following:

```
{
  "Role": {
    "RoleName": "s3ExportRole",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "pinpoint.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    },
    "RoleId": "AROAIKCP0353GIPEXAMPLE",
    "Arn": "arn:aws:iam::123456789012:role/s3ExportRole",
    "CreateDate": "2018-04-11T18:52:36.712Z",
    "Path": "/"
  }
}
```

3. At the command line, type the following command to attach the policy that you created in the previous section to the role you just created:

```
aws iam attach-role-policy --policy-arn arn:aws:iam::123456789012:policy/s3ExportPolicy
--role-name s3ExportRole
```

In the preceding command, replace `arn:aws:iam::123456789012:policy/s3ExportPolicy` with the ARN of the policy that you created in the previous section.

IAM Role for Streaming Events to Kinesis

Amazon Pinpoint can automatically send app usage data, or *event data*, from your app to a Kinesis stream or Amazon Kinesis Data 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 Kinesis \(p. 99\)](#).

Permissions Policies

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

Amazon Kinesis Data Streams

The following policy allows Amazon Pinpoint to send event data to a 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 Data Firehose

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

```
{
  "Version": "2012-10-17",
  "Statement": {
    "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 \(p. 135\)](#) 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.

Limits in Amazon Pinpoint

The following sections describe limits within Amazon Pinpoint.

Topics

- [General Limits \(p. 138\)](#)
- [Endpoint Limits \(p. 138\)](#)
- [Endpoint Import Limits \(p. 139\)](#)
- [Segment Limits \(p. 139\)](#)
- [Campaign Limits \(p. 139\)](#)
- [Mobile Push Limits \(p. 140\)](#)
- [Email Limits \(p. 141\)](#)
- [SMS Limits \(p. 142\)](#)
- [Voice Limits \(p. 143\)](#)
- [Event Ingestion Limits \(p. 144\)](#)
- [Requesting a Limit Increase \(p. 145\)](#)

General Limits

The following limits affect general use of Amazon Pinpoint.

Resource	Default Limit	Eligible for Increase
API request payload size	7 MB per request	No
Apps	100 per account	No

Endpoint Limits

The following limits apply to the [Endpoints](#) resource in the Amazon Pinpoint API.

Resource	Default Limit	Eligible for Increase
Attributes assigned to the <code>Attributes</code> , <code>Metrics</code> , and <code>UserAttributes</code> parameters collectively	40 per app	No
Attributes assigned to the <code>Attributes</code> parameter	40 per app	No
Attributes assigned to the <code>Metrics</code> parameter	40 per app	No
Attributes assigned to the <code>UserAttributes</code> parameter	40 per app	No
Attribute name length	50 characters	No
Attribute value length	100 characters	No

Resource	Default Limit	Eligible for Increase
EndpointBatchItem objects in an EndpointBatchRequest payload	100 per payload. The payload size can't exceed 7 MB.	No
Endpoints with the same user ID	10 unique endpoints per user ID	No
Values assigned to Attributes parameter attributes	50 per attribute	No
Values assigned to UserAttributes parameter attributes	50 per attribute	No

Endpoint Import Limits

The following limits apply when you import endpoints into Amazon Pinpoint.

Resource	Default Limit	Eligible for Increase
Concurrent import jobs	2 per account	Yes (p. 145)
Import size	1 GB per import job (For example, if each endpoint is 4 KB or less, you can import 250,000 endpoints.)	Yes (p. 145)

Segment Limits

The following limits apply to the [Segments](#) resource in the Amazon Pinpoint API.

Resource	Default Limit	Eligible for Increase
Maximum number of dimensions that can be used to create a segment	100 per segment	No

Campaign Limits

The following limits apply to the [Campaigns](#) resource in the Amazon Pinpoint API.

Resource	Default Limit	Eligible for Increase
Active campaigns	200 per account Note An <i>active campaign</i> is a campaign that hasn't completed or failed.	Yes (p. 145)

Resource	Default Limit	Eligible for Increase
	Active campaigns have a status of <code>SCHEDULED</code> , <code>EXECUTING</code> , or <code>PENDING_NEXT_RUN</code> .	
Message sends	100 million per campaign activity	Yes (p. 145)
Event-based campaigns	<p>Each project can include up to 10 campaigns that are sent when events occur.</p> <p>Campaigns that use event-based triggers have to use dynamic segments (that is, they can't use imported segments).</p> <p>Event-based campaigns are only sent to customers who use apps that run version 2.7.2 or later of the AWS Mobile SDK for Android or version 2.6.30 or later of the AWS Mobile SDK for iOS.</p> <p>If Amazon Pinpoint can't deliver a message from an event-based campaign within five minutes, it drops the message and doesn't attempt to re-deliver it.</p>	No

Mobile Push Limits

The following limits apply to messages that Amazon Pinpoint delivers through mobile push channels.

Resource	Default Limit	Eligible for Increase
Maximum number of mobile push notifications that can be sent per second	25,000 notifications per second	No
Amazon Device Messaging (ADM) message payload size	6 KB per message	No
Apple Push Notification service (APNs) message payload size	4 KB per message	No
APNs sandbox message payload size	4 KB per message	No
Baidu Cloud Push message payload size	4 KB per message	No
Firebase Cloud Messaging (FCM) or Google Cloud Messaging (GCM) message payload size	4 KB per message	No

Email Limits

The limits in the following sections apply to the Email channel.

Email Sending Limits

Resource	Default Limit	Eligible for Increase
Number of emails that can be sent per 24-hour period (<i>sending quota</i>)	<p>If your account is in the sandbox: 200 emails per 24-hour period.</p> <p>If your account is out of the sandbox, the quota varies based on your specific use case.</p> <p>Note This quota is based on the number of recipients, as opposed to the number of unique messages sent. A <i>recipient</i> is any email address on the To: line.</p>	Yes (p. 145)
Number of emails that can be sent each second (<i>sending rate</i>)	<p>If your account is in the sandbox: 1 email per second.</p> <p>If your account is out of the sandbox, the rate varies based on your specific use case.</p> <p>Note This rate is based on the number of recipients, as opposed to the number of unique messages sent. A <i>recipient</i> is any email address on the To: line.</p>	Yes (p. 145)

Email Message Limits

Resource	Default Limit	Eligible for Increase
Maximum message size (including attachments)	10 MB per message.	No
Number of verified identities	<p>10,000 identities.</p> <p>Note <i>Identities</i> refers to email addresses or domains, or any combination of the two. Every email</p>	No

Resource	Default Limit	Eligible for Increase
	you send using Amazon Pinpoint must be sent from a verified identity.	

Email Sender and Recipient Limits

Resource	Default Limit	Eligible for Increase
Sender address	All sending addresses or domains must be verified.	No
Recipient address	If your account is still in the sandbox, all recipient email addresses or domains must be verified. If your account is out of the sandbox, you can send to any valid address.	Yes (p. 145)
Number of recipients per message	50 recipients per message.	No
Number of identities that you can verify	10,000 identities per AWS Region. Note <i>Identities</i> refers to email addresses or domains, or any combination of the two. Every email you send using Amazon Pinpoint must be sent from a verified identity.	No

SMS Limits

The following limits apply to the SMS channel.

Resource	Default Limit	Eligible for Increase
Account spend threshold	USD\$1.00 per account.	Yes (p. 145)
Number of SMS messages that can be sent each second (<i>sending rate</i> #)	20 messages per second.	No
Number of Amazon SNS topics for two-way SMS	100,000 per account.	Yes (p. 145)

Voice Limits

The following limits apply to the Voice channel.

Resource	Default Limit	
Number of voice messages that can be sent in a 24-hour period	<p>If your account is in the sandbox: 20 messages.</p> <p>If your account is out of the sandbox: unlimited.</p>	
Number of voice messages that can be sent to a single recipient in a 24-hour period	5 messages.	
Number of voice messages that can be sent per minute	<p>If your account is in the sandbox: 5 calls per minute.</p> <p>If your account is out of the sandbox: 20 calls per minute.</p>	
Number of voice messages that can be sent from a single originating phone number per second	1 message per second.	
Voice message length	<p>If your account is in the sandbox: 30 seconds.</p> <p>If your account is out of the sandbox: 5 minutes.</p>	
Ability to send voice messages to international phone numbers	<p>If your account is in the sandbox, you can only send messages to recipients in the following countries:</p> <ul style="list-style-type: none"> • Australia • Canada • China • Germany • Hong Kong • Israel • Japan • Mexico • Singapore • Sweden • United States • United Kingdom <p>If your account is out of the sandbox, you can send messages to recipients in any country.</p>	

Resource	Default Limit	
	Note International calls are subject to additional fees, which vary by destination country or region.	
Number of characters in a voice message	3,000 billable characters (characters in words that are spoken) 6,000 characters total (including billable characters and SSML tags)	
Number of configuration sets	10,000 voice configuration sets per AWS Region.	

Event Ingestion Limits

The following limits apply to the ingestion of events using the AWS Mobile SDKs and the [Amazon Pinpoint Events API](#).

Resource	Default Limit	Eligible for Increase
Maximum number of custom event types	1,500 per app	No
Maximum number of custom attribute keys	500 per app	No
Maximum number of custom attribute values per attribute key	100,000	No
Maximum number of characters per attribute key	50	No
Maximum number of characters per attribute value	200	No
Maximum number of custom metric keys	500 per app	No
Maximum number events in a request	100 per request	No
Maximum size of a request	4 MB	No
Maximum size of an individual event	1,000 KB	No
Maximum number of attribute keys and metric keys for each event	40 per request	No

Requesting a Limit Increase

If the value in the **Eligible for Increase** column in any of the tables above is **Yes**, you can request a change to that limit.

To request a limit increase

1. Sign in to the AWS Management Console at <https://console.aws.amazon.com/>.
2. Create a new Support case at <https://console.aws.amazon.com/support/home#/case/create>.
3. On the **Create Case** page, make the following selections:
 - For **Regarding**, choose **Service Limit Increase**.
 - For **Limit Type**, choose one of the following options:
 - Choose **Amazon Pinpoint** for limit increases related to Amazon Pinpoint campaigns and imports.
 - Choose **Amazon Pinpoint Email** for limit increases related to the email channel.
 - Choose **Amazon Pinpoint SMS** for limit increases related to the SMS channel.
4. For **Use Case Description**, explain why you are requesting the limit increase.
5. For **Support Language**, choose the language you prefer to use when communicating with the AWS Support team.
6. For **Contact Method**, choose your preferred method of communicating with the AWS Support team.
7. Choose **Submit**.

Creating Custom Channels with AWS Lambda

This is prerelease documentation for a feature in public beta release. It is subject to change.

Amazon Pinpoint supports messaging channels for mobile push, email, and SMS. However, some messaging use cases might require unsupported channels. For example, you might want to send a message to an instant messaging service, such as Facebook Messenger, or you might want to display a notification within your web application. In such cases, you can use AWS Lambda to create a custom channel that performs the message delivery outside of Amazon Pinpoint.

AWS Lambda is a compute service that you can use to run code without provisioning or managing servers. You package your code and upload it to Lambda as *Lambda functions*. Lambda runs a function when the function is invoked, which might be done manually by you or automatically in response to events.

For more information, see [Lambda Functions](#) in the *AWS Lambda Developer Guide*.

To create a custom channel, you define a Lambda function that handles the message delivery for an Amazon Pinpoint campaign. Then, you assign the function to a campaign by defining the campaign's `CampaignHook` settings. These settings include the Lambda function name and the `CampaignHook` mode. By setting the mode to `DELIVERY`, you specify that the Lambda function handles the message delivery instead of Amazon Pinpoint.

A Lambda function that you assign to a campaign is referred to as an Amazon Pinpoint *extension*.

With the `CampaignHook` settings defined, Amazon Pinpoint automatically invokes the Lambda function when it runs the campaign, without sending the campaign's message to a standard channel. Instead, Amazon Pinpoint sends *event data* about the message delivery to the function, and it allows the function to handle the delivery. The event data includes the message body and the list of endpoints to which the message should be delivered.

After Amazon Pinpoint successfully invokes the function, it generates a successful send event for the campaign.

Note

You can also use the `CampaignHook` settings to assign a Lambda function that modifies and returns a campaign's segment before Amazon Pinpoint delivers the campaign's message. For more information, see [Customizing Segments with AWS Lambda \(p. 54\)](#).

To create a custom channel with AWS Lambda, first create a function that accepts the event data sent by Amazon Pinpoint and handles the message delivery. Then, authorize Amazon Pinpoint to invoke the function by assigning a Lambda function policy. Finally, assign the function to one or more campaigns by defining `CampaignHook` settings.

Event Data

When Amazon Pinpoint invokes your Lambda function, it provides the following payload as the event data:

```
{
```

```
"MessageConfiguration": {Message configuration}
"ApplicationId": ApplicationId,
"CampaignId": CampaignId,
"TreatmentId": TreatmentId,
"ActivityId": ActivityId,
"ScheduledTime": Scheduled Time,
"Endpoints": {
  EndpointId: {Endpoint definition}
  . . .
}
}
```

The event data provides the following attributes:

- **MessageConfiguration** – Has the same structure as the [DirectMessageConfiguration](#) in the Messages resource in the Amazon Pinpoint API.
- **ApplicationId** – The ID of the Amazon Pinpoint project to which the campaign belongs.
- **CampaignId** – The ID of the Amazon Pinpoint project for which the function is invoked.
- **TreatmentId** – The ID of a campaign variation used for A/B testing.
- **ActivityId** – The ID of the activity being performed by the campaign.
- **ScheduledTime** – The schedule time at which the campaign's messages are delivered in ISO 8601 format.
- **Endpoints** – A map that associates endpoint IDs with endpoint definitions. Each event data payload contains up to 50 endpoints. If the campaign segment contains more than 50 endpoints, Amazon Pinpoint invokes the function repeatedly, with up to 50 endpoints at a time, until all endpoints have been processed.

Creating a Lambda Function

To create a Lambda function, refer to [Building Lambda Functions](#) in the *AWS Lambda Developer Guide*.

Example Lambda Function

The following example Lambda function receives event data when Amazon Pinpoint runs a campaign, and it sends the campaign's message to Facebook Messenger:

```
"use strict";

var https = require("https");
var q = require("q");

var VERIFY_TOKEN = "my_token";
var PAGE_ACCESS_TOKEN = "EAF...DZD";
/* this constant can be put in a constants file and shared between this function and your
Facebook Messenger webhook code */
var FACEBOOK_MESSENGER_PSID_ATTRIBUTE_KEY = "facebookMessengerPsid";

exports.handler = function(event, context, callback) {

  var deliverViaMessengerPromises = [];

  if (event.Message && event.Endpoints) {
    for (var endpoint in event.Endpoints) {
      if (isFbookMessengerActive(event.Endpoints[endpoint])) {
        deliverViaMessengerPromises.push(deliverViaMessenger(event.Message,
event.Endpoints[endpoint].User));
      }
    }
  }
}
```



```
    }
  }
}

/* default OK response */
var response = {
  body: "ok",
  statusCode: 200
};

if (deliverViaMessengerPromises.length > 0) {
  q.all(deliverViaMessengerPromises).done(function() {
    callback(null, response);
  });
} else {
  callback(null, response);
}
}

/**
Example Pinpoint Endpoint User object where we've added custom attribute
facebookMessengerPsid to store the PSID needed by Facebook's API
{
  "UserId": "7a9870b7-493c-4521-b0ca-08bbbc36e595",
  "UserAttributes": {
    "facebookMessengerPsid": [ "1667566386619741" ]
  }
}
**/
function isFbookMessengerActive(endpoint) {
  return endpoint.User && endpoint.User.UserAttributes &&
  endpoint.User.UserAttributes[FACEBOOK_MESSENGER_PSID_ATTRIBUTE_KEY];
}

/**
Sample message object from Pinpoint. This sample was an SMS so it has "smsmessage"
attribute but this will vary for each messaging channel
{
  "smsmessage": {
    "body": "This message should be intercepted by a campaign hook."
  }
}
**/
function deliverViaMessenger(message, user) {
  var deferred = q.defer();

  var messageText = message["smsmessage"]["body"];
  var pinpointUserId = user.UserId;
  var facebookPsid = user.UserAttributes[FACEBOOK_MESSENGER_PSID_ATTRIBUTE_KEY][0];
  console.log("Sending message for user %s and page %s with message:", pinpointUserId,
facebookPsid, messageText);

  var messageData = {
    recipient: {
      id: facebookPsid
    },
    message: {
      text: messageText
    }
  };
};

var body = JSON.stringify(messageData);
var path = "/v2.6/me/messages?access_token=" + PAGE_ACCESS_TOKEN;
var options = {
  host: "graph.facebook.com",
```

```
    path: path,
    method: "POST",
    headers: {
      "Content-Type": "application/json"
    }
  };

  var req = https.request(options, httpsCallback);

  req.on("error", function(e) {
    console.log("Error posting to Facebook Messenger: " + e);
    deferred.reject(e);
  });

  req.write(body);
  req.end();

  return deferred.promise;

  function httpsCallback(response) {
    var str = "";
    response.on("data", function(chunk) {
      str += chunk;
    });
    response.on("end", function() {
      console.log(str);
      deferred.resolve(response);
    });
  }
}
```

Assigning a Lambda Function Policy

Before you can use your Lambda function to process your endpoints, you must authorize Amazon Pinpoint to invoke your Lambda function. To grant invocation permission, assign a *Lambda function policy* to the function. A Lambda function policy is a resource-based permissions policy that designates which entities can use your function and what actions those entities can take.

For more information, see [Using Resource-Based Policies for AWS Lambda \(Lambda Function Policies\)](#) in the *AWS Lambda Developer Guide*.

Example Function Policy

The following policy grants permission to the Amazon Pinpoint service principal to use the `lambda:InvokeFunction` action:

```
{
  "Sid": "sid",
  "Effect": "Allow",
  "Principal": {
    "Service": "pinpoint.us-east-1.amazonaws.com"
  },
  "Action": "lambda:InvokeFunction",
  "Resource": "{arn:aws:lambda:us-east-1:account-id:function:function-name}",
  "Condition": {
    "ArnLike": {
      "AWS:SourceArn": "arn:aws:mobiletargeting:us-east-1:account-id:/apps/application-id/campaigns/campaign-id"
    }
  }
}
```

```
}
```

Your function policy requires a `Condition` block that includes an `AWS:SourceArn` key. This code states which Amazon Pinpoint campaign is allowed to invoke the function. In this example, the policy grants permission to only a single campaign ID. To write a more generic policy, use multi-character match wildcards (*). For example, you can use the following `Condition` block to allow any Amazon Pinpoint campaign in your AWS account to invoke the function:

```
"Condition": {  
  "ArnLike": {  
    "AWS:SourceArn": "arn:aws:mobiletargeting:us-east-1:account-id:/apps/*/campaigns/*"  
  }  
}
```

Granting Amazon Pinpoint Invocation Permission

You can use the AWS Command Line Interface (AWS CLI) to add permissions to the Lambda function policy assigned to your Lambda function. To allow Amazon Pinpoint to invoke a function, use the Lambda `add-permission` command, as shown by the following example:

```
$ aws lambda add-permission \  
> --function-name function-name \  
> --statement-id sid \  
> --action lambda:InvokeFunction \  
> --principal pinpoint.us-east-1.amazonaws.com \  
> --source-arn arn:aws:mobiletargeting:us-east-1:account-id:/apps/application-id/  
campaigns/campaign-id
```

If you want to provide a campaign ID for the `--source-arn` parameter, you can look up your campaign IDs by using the Amazon Pinpoint `get-campaigns` command with the AWS CLI. This command requires an `--application-id` parameter. To look up your application IDs, sign in to the Amazon Pinpoint console at <https://console.aws.amazon.com/pinpoint/>, and go to the **Projects** page. The console shows an **ID** for each project, which is the project's application ID.

When you run the Lambda `add-permission` command, AWS Lambda returns the following output:

```
{  
  "Statement": "{\"Sid\":\"sid\",  
    \"Effect\":\"Allow\",  
    \"Principal\":{\"Service\":\"pinpoint.us-east-1.amazonaws.com\"},  
    \"Action\":\"lambda:InvokeFunction\",  
    \"Resource\":\"arn:aws:lambda:us-east-1:111122223333:function:function-name\",  
    \"Condition\":  
      {\"ArnLike\":  
        {\"AWS:SourceArn\":  
          \"arn:aws:mobiletargeting:us-east-1:111122223333:/apps/application-id/campaigns/  
campaign-id\"}}}"  
}
```

The `Statement` value is a JSON string version of the statement added to the Lambda function policy.

Assigning a Lambda Function to a Campaign

You can assign a Lambda function to an individual Amazon Pinpoint campaign. Or, you can set the Lambda function as the default used by all campaigns for a project, except for those campaigns to which you assign a function individually.

To assign a Lambda function to an individual campaign, use the Amazon Pinpoint API to create or update a [Campaign](#) object, and define its `CampaignHook` attribute. To set a Lambda function as the default for all campaigns in a project, create or update the [Settings](#) resource for that project, and define its `CampaignHook` object.

In both cases, set the following `CampaignHook` attributes:

- `LambdaFunctionName` – The name or ARN of the Lambda function that Amazon Pinpoint invokes to send messages for the campaign.
- `Mode` – Set to `DELIVERY`. With this mode, Amazon Pinpoint uses the function to deliver the messages for a campaign, and it doesn't attempt to send the messages through the standard channels.

Document History for Amazon Pinpoint

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

- **Latest documentation update:** February 6, 2019

Change	Description	Date
Code examples	Added code examples (p. 63) in several programming languages that show you how to send email (p. 63) , SMS (p. 88) , and voice (p. 94) messages programmatically.	February 6, 2019
Regional availability	Amazon Pinpoint is now available in the US West (Oregon) and EU (Frankfurt) AWS Regions.	December 21, 2018
Voice channel	You can use the new Amazon Pinpoint voice channel to create voice messages and deliver them to your customers over the phone. Currently, you can only send voice messages by using the Amazon Pinpoint SMS and Voice API.	November 15, 2018
EU (Ireland) Availability	Amazon Pinpoint is now available in the EU (Ireland) AWS Region.	October 25, 2018
Events API	Use the Amazon Pinpoint API to record events (p. 7) and associate them with endpoints.	August 7, 2018
Code examples for defining and looking up endpoints	Code examples are added that show you how to define, update, delete, and look up endpoints. Examples are provided for the AWS CLI, AWS SDK for Java, and the Amazon Pinpoint API. For more information, see Defining Your Audience to Amazon Pinpoint (p. 10) and Accessing Audience Data in Amazon Pinpoint (p. 35) .	August 7, 2018
Endpoint export permissions	Configure an IAM policy (p. 131) that allows you to export	May 1, 2018

Change	Description	Date
	Amazon Pinpoint endpoints to an Amazon S3 bucket.	
Updated topics for Amazon Pinpoint integration	Integrate Amazon Pinpoint (p. 3) with your Android, iOS, or JavaScript application by using AWS SDKs or libraries.	March 23, 2018
AWS CloudTrail logging	Added information about logging Amazon Pinpoint API calls with CloudTrail (p. 106) .	February 6, 2018
Updated service limits	Updated Limits (p. 138) with additional information about email limits.	January 19, 2018
Public beta for Amazon Pinpoint extensions	Use AWS Lambda functions to customize segments (p. 54) or create custom messaging channels (p. 146) .	November 28, 2017
External ID removed from IAM trust policies	The external ID key is removed from the example trust policy (p. 130) and example Java code (p. 51) for importing segments.	October 26, 2017
Push notification payload limits	The limits include payload sizes for mobile push messages (p. 140) .	October 25, 2017
Updated service limits	Added SMS and email channel information to Limits (p. 138) .	October 9, 2017
ADM and Baidu mobile push	Update your app code to handle push notifications from the Baidu and ADM mobile push channels.	September 27, 2017
User IDs and authentication events with Amazon Cognito user pools.	If you use Amazon Cognito user pools to manage user sign-in in your mobile apps, Amazon Cognito assigns user IDs to endpoints, and it reports authentication events to Amazon Pinpoint.	September 26, 2017
User IDs	Assign user IDs to endpoints to monitor app usage from individual users. Examples are provided for the AWS Mobile SDKs (p. 5) and SDK for Java (p. 15) .	August 31, 2017

Change	Description	Date
Authentication events	Report authentication events to learn how frequently users authenticate with your app. Examples are provided in Reporting Events in Your Application (p. 6) .	August 31, 2017
Updated sample events	The example events (p. 100) include events that Amazon Pinpoint streams for email and SMS activity.	June 08, 2017
Android session management	Manage sessions 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. .	March 31, 2017
Event streams	You can configure Amazon Pinpoint to send your app and campaign events to an Kinesis stream (p. 99) .	March 24, 2017
Permissions	See Permissions (p. 115) 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