Amazon Simple Notification Service

개발자 안내서

API Version 2010-03-31

Amazon Web Services

Amazon Simple Notification Service: 개발자 안내서

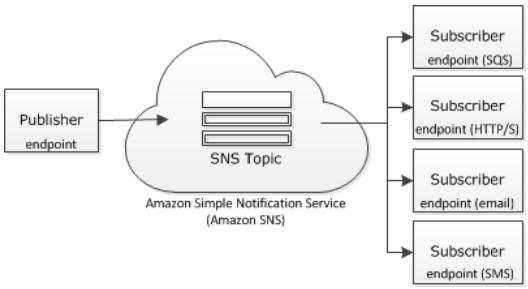
Amazon Web Services

Abstract

Amazon Simple Notification Service 개념 및 프로그래밍 정보의 간결한 개요입니다.

Amazon Simple Notification Service 이란 무엇입니까?

Amazon Simple Notification Service(Amazon SNS)은 endpoint 또는 클라이언트를 구독하는 메시지의 전송 또는 송신을 조정 및 관리하는 웹 서비스입니다. In Amazon SNS에는 게시자 및 구독자 두 유형의 클라이언트가 있으며 생산자 및 소비자라고도 합니다. 게시자는 주제에 대한 메시지를 생산 및 발송함으로써 구독자와 비동시적으로 통신하는 논리적 액세스 및 커뮤니케이션 채널입니다. 구독자는(예. 웹 서버, 이메일 주소, Amazon SQS 대기열) 주제를 구독하는 경우 지원되는 프로토콜(예. Amazon SQS, HTTP/S, 이메일, SMS) 중 하나를 거쳐 메시지 또는 알림을 소비 또는 수신합니다.



Amazon SNS을 이용할 때 사용자는(소유자로써) 주제를 어떤 게시자/구독자와 통신할지를 결정하는 정책을 지정함으로써 주제 생성 및 이에 대한 액세스를 제어합니다. 게시자는 생성한 주제 또는 게시할 권한이 있는 주제에 메시지를 보냅니다. 게시자는 각 메시지에 특정 대상 주소를 포함하는 대신 해당 주제에 메시지를 전송할 수 있습니다. Amazon SNS은 해당 주제를 구독하는 구독자 목록에 주제를 일치시켜 각각의 구독자에게 메시지를 전송합니다. 각 주제는 Amazon SNS endpoint를 확인하는 고유한 이름을 가지므로 게시자는 메시지를 게시하고 구독자는 알림 받도록 등록할 수 있습니다. 구독자는 구독하는 주제에 게시된 모든 메시지를 수신하며 주제에 대한 모든 구독자는 동일한 메시지를 수신합니다.

Amazon Simple Notification Service을 처음 사용하십니까?

Amazon SNS를 처음 사용할 경우 먼저 다음의 섹션을 읽을 것을 권장합니다.

- What is Amazon SNS – 이 섹션의 나머지 부분은 Amazon SNS를 소개하는 동영상을 포함하고 Amazon Simple Notification Service 시작하기 (p. 4)에 나온 예제를 설명하며 일반적인 사용 사례 시나리오를 보여줍니다.
- Getting Started – Amazon Simple Notification Service 시작하기 (p. 4) 섹션은 주제를 생성 및 구독하고, 주제에 대해 메시지를 게시하며, 구독을 해지하거나 주제를 삭제하는 방법을 설명합니다.

시작하기 섹션 이후

시작하기 섹션 이후 사용자는 Amazon SNS 옵션에 대한 자세한 내용을 학습하고자 할 것입니다. 다음의 섹션은 Amazon SNS 작동에 대한 자세한 정보를 제공합니다.

- Amazon SNS 주제에 대한 액세스 관리 (p. 11)

  사용자는 어떤 endpoint가 주제를 허용하도록 할지, 누가 주제를 게시할 수 있을지, 어떤 조건에서 게시할 수 있는지 등을 상세하게 규제하게 됩니다. 이 부록은 *액세스 제어 정책*을 통해 액세스를 제어하는 방법을 설명합니다.
- Amazon CloudWatch를 사용한 Amazon SNS 모니터링 (p. 48)

  Amazon SNS 및 Amazon CloudWatch는 통합되어 사용자가 모든 활성 Amazon SNS 주제의 메트릭을 수집하고 확인하며 분석할 수 있습니다.
- Amazon SQS 대기열에 Amazon SNS 메시지 전송 (p. 79)

  사용자는 Amazon SNS를 사용하여 하나 이상의 Amazon SQS 대기열에 메시지를 전송할 수 있습니다.
- Amazon SNS를 사용한 SMS 알림 송수신 (p. 95)

  Amazon Simple Notification Service(Amazon SNS)을 사용하여 SMS 사용 가능한 휴대전화 및 스마트폰에 SMS 알림을 전송할 수 있습니다.
- Amazon SNS 메시지를 HTTP/HTTPS Endpoint로 전송 (p. 104)

  Amazon SNS를 사용하여 하나 이상의 HTTP endpoint 또는 HTTPS endpoint에 알림 메시지를 전송할 수 있습니다.

Amazon SNS에 액세스

사용자는 Command Line Interface(CLI)라는 AWS Management Console을 사용하고 (http://aws.amazon.com/developertools/3688) Amazon SNS Query API에 직접 코드를 작성함으로써 Amazon SNS에 액세스할 수 있습니다(Amazon Simple Notification Service API Reference 참조).

SDK를 사용해 선호하는 프로그래밍 언어로 Amazon SNS에 액세스할 수도 있습니다. SDK는 다음의 사항을 자동으로 수행하는 기능을 갖추고 있습니다.

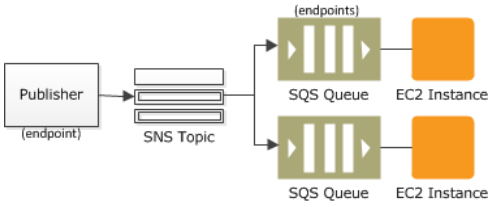
- 서비스 요청에 대한 암호화 서명
- 요청 재시도
- 오류 응답 처리

사용 가능한 SDK 목록은 Amazon Web Services용 도구를 참조하십시오.

일반적인 Amazon SNS 시나리오

팬아웃

"팬아웃" 시나리오는 Amazon SNS 메시지가 주제에 전송 및 복제되어 다중 Amazon SQS 대기열, HTTP endpoints 또는 이메일 주소로 푸시되는 경우를 말합니다. 따라서 평행한 비동시적 처리가 가능합니다. 예를 들어, 사용자는 제품에 대한 주문이 생성될 때 주제에 Amazon SNS 메시지를 전송하는 애플리케이션을 개발할 수 있을 것입니다. 그러면 해당 주제를 구독하는 Amazon SQS 대기열은 새 주문에 대해 동일한 알림을 수신할 것입니다. 대기열 중 하나에 연결된 Amazon EC2 서버 인스턴스는 다른 서버 인스턴스가 수신된 모든 주문을 분석하기 위해 데이터 웨어하우스에 연결될 동안 해당 주문의 프로세스 또는 완료를 처리할 수 있을 것입니다.



"팬아웃"을 사용하는 또 다른 방법은 생산 환경에 개발 환경으로 전송된 데이터를 복제하는 것입니다. 기존 예제를 확대해보면, 사용자는 새로운 수신 주문의 동일한 주제에 대한 다른 대기열을 아직 구독할 수 있을 것입니다. 그러면 이 새로운 대기열을 개발 환경에 연결함으로써 사용자는 개선을 계속하면서 생산 환경에서 수신한 데이터를 사용하여 애플리케이션을 테스트할 수 있습니다. Amazon SQS 대기열에 Amazon SNS 메시지를 전송하는 것에 대한 자세한 내용은 Amazon SQS 대기열에 Amazon SNS 메시지 전송 (p. 79)를 참조하십시오. HTTP/S endpoint에 Amazon SNS 메시지 전송하기에 대한 자세한 내용은 Amazon SNS 메시지를 HTTP/HTTPS Endpoint로 전송 (p. 104)를 참조하십시오.

애플리케이션 및 시스템 경보

애플리케이션 및 시스템 경보는 사전 지정된 임계값에 의해 트리거되는 알림으로, 지정된 사용자에게 SMS 및/또는 이메일을 보냅니다. 예를 들어, 많은 AWS 서비스가 Amazon SNS를 사용함에 따라 AWS Auto Scaling 그룹에 특정 변경 사항이 발생하는 등의 경우 사용자는 즉시 알림을 받을 수 있습니다.

푸시 이메일 및 문자 메시지

푸시 이메일 및 문자 메시지는 이메일 및/또는 SMS를 통해 개인 또는 그룹에 메시지를 전송하는 두 가지 방법입니다. 예를 들어, 사용자는 구독자에게 이메일 또는 SMS로 특정 뉴스 헤드라인을 푸시하기 위해 Amazon SNS를 사용할 수 있습니다. 수신한 이메일 또는 SMS 문자에 흥미를 느낀 사람은 자세한 내용을 확인하기 위해 웹사이트를 방문하거나 애플리케이션을 시작할 수 있습니다. SMS 알림을 전송하기 위한 Amazon SNS 사용에 대한 자세한 내용은 Amazon SNS를 사용한 SMS 알림 송수신 (p. 95)을 참조하십시오.

모바일 푸시 알림

모바일 푸시 알림을 통해 메시지를 모바일 앱으로 바로 전송할 수 있습니다. 예를 들어, Amazon SNS를 사용하여 업데이트가 가능하다고 표시하며 앱으로 알림을 전송할 수 있습니다. 알림 메시지는 업데이트를 다운로드 및 설치하기 위한 링크를 포함할 수 있습니다. 모바일 지점에 직접 알림 메시지를 전송하기 위한 Amazon SNS 사용에 대한 자세한 내용은 Amazon SNS 모바일 푸시 알림 (p. 52)을 참조하십시오.

Amazon Simple Notification Service 시작하기

이 섹션은 Amazon SNS 개념을 이해하고 사용 가능한 도구 및 인터페이스를 빠르게 설치 및 사용하도록 주제 생성 및 게시에 대한 정보를 설명합니다.

Topics

시작하기 전

Amazon SNS를 사용하려면 AWS 계정이 있어야 합니다. 아직 계정이 없는 경우 Amazon SNS에 가입하면 계정을 만들도록 요청하는 메시지가 표시됩니다.

Amazon SNS에 가입

1. http://aws.amazon.com/sns/로 이동하여 Sign Up for Amazon SNS를 클릭합니다.
2. 화면에 표시되는 지시 사항을 따릅니다.

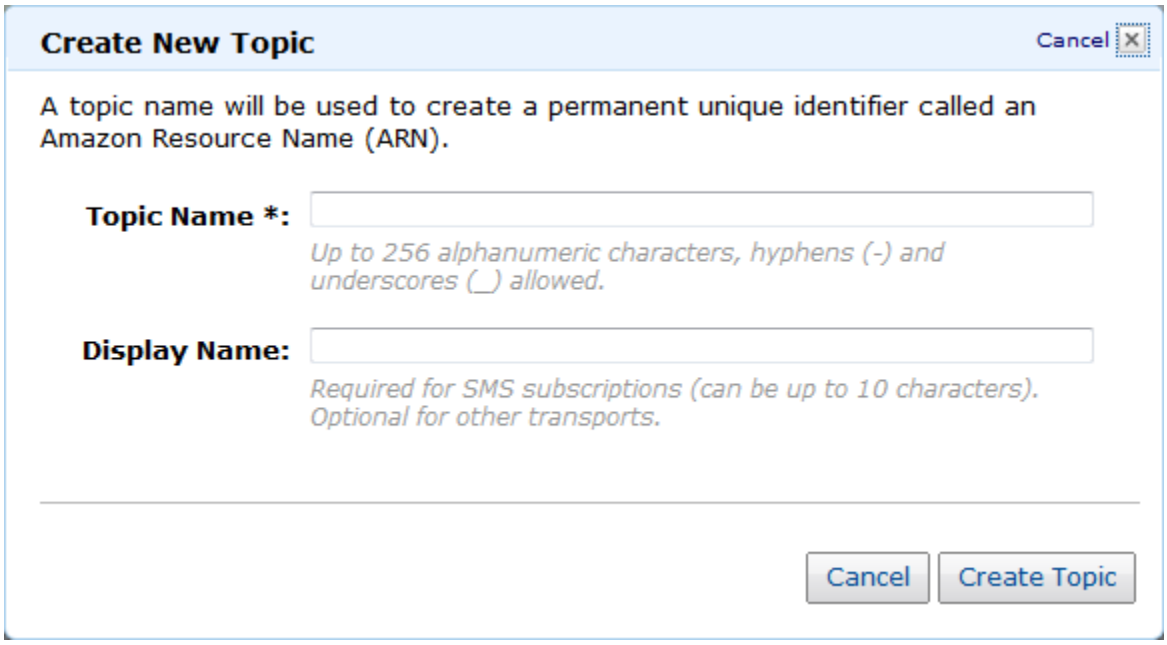
계정이 활성화되고 사용 가능한 상태가 되면 AWS에서 사용자에게 이메일로 알립니다.

주제 생성

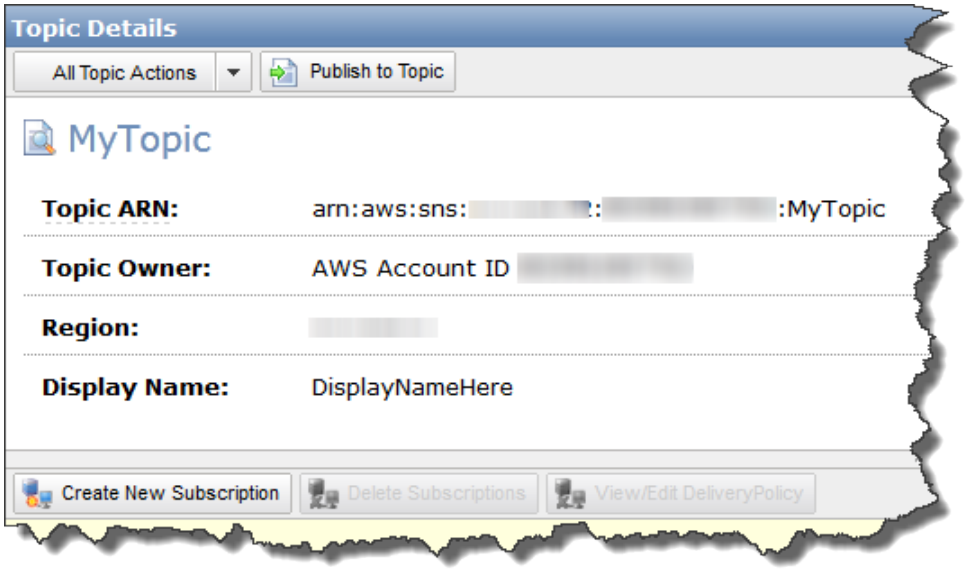
이제 Amazon SNS에 가입하여 주제를 생성할 준비가 되었습니다. 주제는 메시지 전송 및 알림 구독을 위한 커뮤니케이션 채널입니다. 게시자와 구독자가 서로 통신할 수 있는 액세스 지점을 제공합니다. 이 섹션에서는 *MyTopic*이라는 이름의 주제를 생성합니다.

주제 생성

1. Sign in to the AWS Management Console and open the Amazon SNS console at https://console.aws.amazon.com/sns/.
2. Create New Topic을 클릭합니다.
   Create New Topic 대화 상자가 표시됩니다.



3. Topic Name 필드에 주제 이름을 입력합니다.
   다음의 예에서는 주제 이름을 *MyTopic*으로 합니다.
4. Create Topic을 클릭합니다.
   새 주제는 Topic Details 페이지에 나타납니다.



5. 다음 작업을 위해 Topic ARN을 복사합니다.

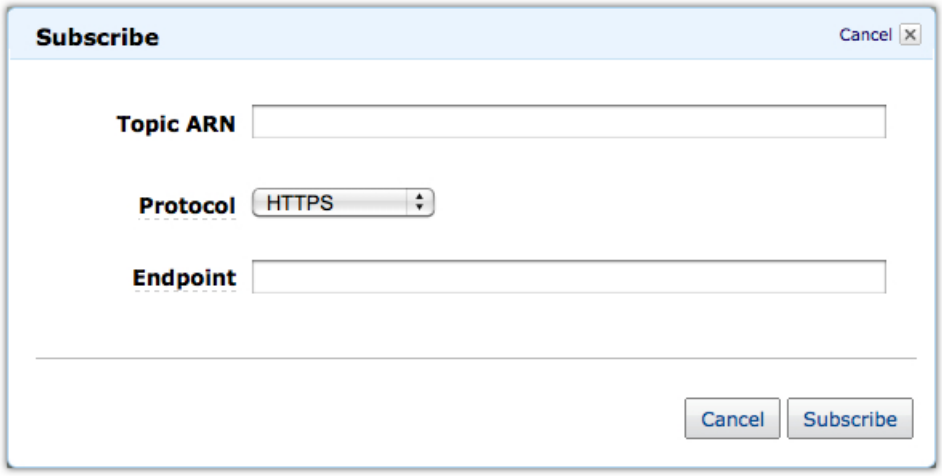
주제 구독

주제에 게시된 메시지를 받으려면 해당 주제를 구독하도록 endpoint를 등록해야 합니다. Endpoint는 Amazon SNS로부터 알림 메시지를 받을 수 있는 웹 서버, 이메일 주소 또는 Amazon SQS 대기열입니다. 주제를 구독하도록 endpoint를 등록하고 구독이 확인되면, endpoint는 주제에 게시된 모든 메시지를 받게 됩니다.

이 섹션에서는 이전 세션에서 생성한 주제를 구독하도록 endpoint를 등록합니다. 이메일 계정으로 주제 메시지를 전송할 수 있도록 구독을 구성합니다.

주제 구독

1. AWS Management Console을 열고 Navigation 창에서 My Subscriptions를 클릭합니다.
   My Subscriptions 페이지가 열립니다.
2. Create New Subscription 버튼을 클릭합니다.
   Subscribe 대화 상자가 나타납니다.



3. Topic ARN 필드에서 이전 작업에서 생성한 주제 ARN를 다음과 같이 붙여넣습니다 `arn:aws:sns:us-west-2:111122223333:MyTopic`.
4. Protocol 드롭다운 상자의 Email을 선택합니다.
5. 알림을 받는 데 사용할 수 있는 이메일 주소를 Endpoint 필드에 입력합니다.

   Important

   Entourage Users: Entourage는 확인 URL을 제거합니다. 다른 이메일 애플리케이션에서 액세스할 수 있는 이메일 주소를 입력하십시오.

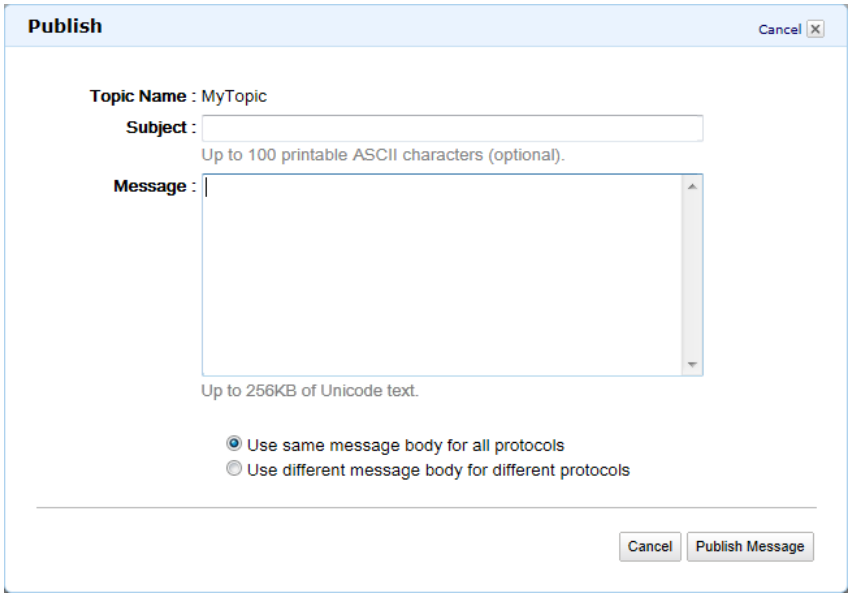
6. Subscribe를 클릭합니다.
7. 이메일 애플리케이션의 AWS 알림에서 메시지를 열고 해당 링크를 클릭하여 구독을 확인합니다.
   Amazon SNS로부터의 확인 반응이 웹 브라우저에 표시됩니다.

주제 게시

게시자는 메시지를 주제로 전송합니다. 새 메시지가 게시되면 Amazon SNS는 주제를 구독하는 모든 endpoint에 해당 메시지를 전송합니다. 이 섹션에서는 이전 작업에서 지정한 이메일 주소로 메시지를 게시합니다.

주제 게시

1. AWS Management Console을 열고 Navigation 창의 My Topics에서 게시할 주제를 클릭합니다.
   Topic Details 페이지가 열립니다.
2. Publish to Topic 버튼을 클릭합니다.
   Publish 대화 상자가 표시됩니다.



3. Subject 필드에 메시지의 제목을 입력합니다.
4. Message 필드에 간단한 메시지를 입력합니다.
5. Publish Message를 클릭합니다.
   확인 대화 상자가 표시됩니다.
6. Close를 클릭하여 확인 대화 상자를 닫습니다.

이제 이메일 애플리케이션을 사용하여 AWS 알림 메시지를 열고 읽을 수 있습니다.

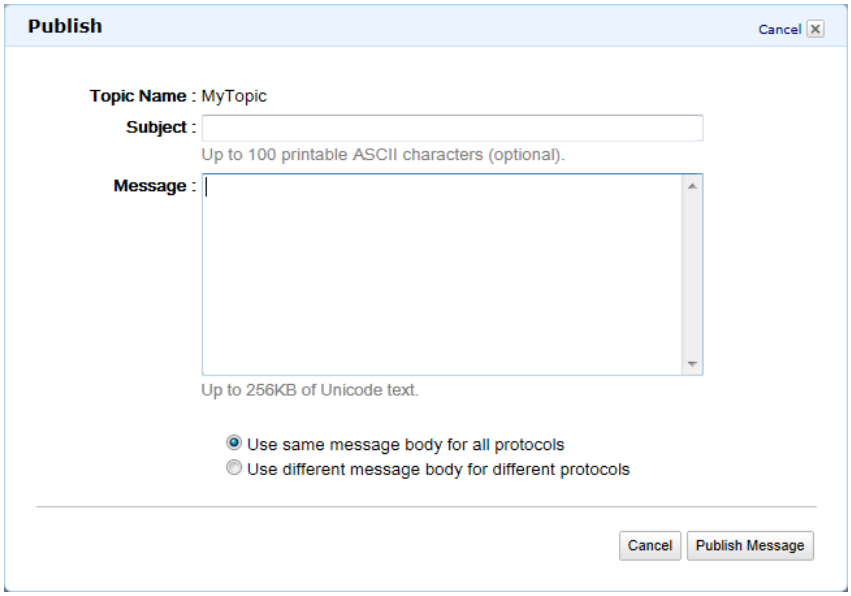
각 프로토콜에 대해 다른 메시지 생성

메시지 형식 지원을 사용하여 각 프로토콜에 전송하는 메시지를 맞춤화할 수 있습니다. 예를 들어, 이메일 및 SMS 구독자에게 전송되는 알림을 각 고객 유형에 맞출 수 있습니다. SMS 사용자들은 SMS 표준이 지원하는 140자로 된 메시지 버전을 받을 수 있으며, 이메일 사용자는 동일한 콘텐츠를 더 길고 자세한 버전으로 받을 수 있습니다.

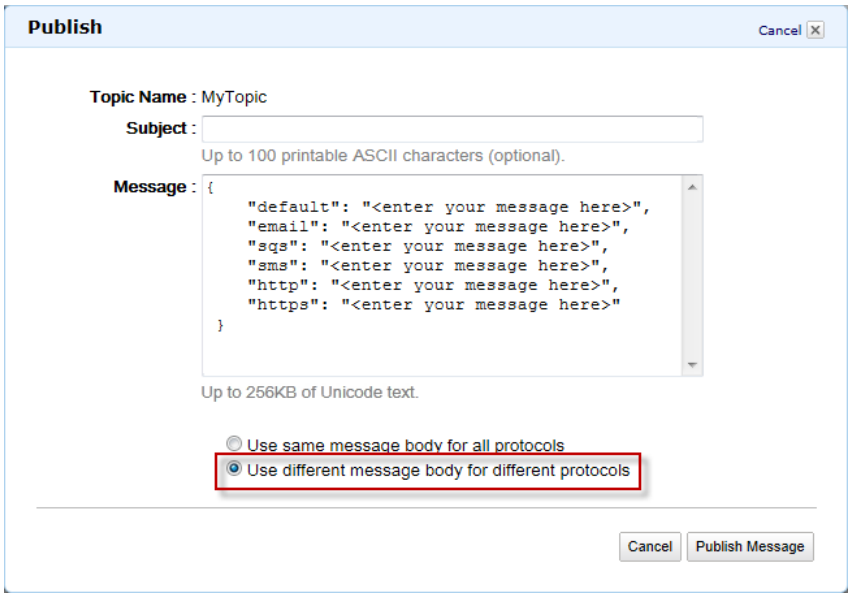
메시지 형식으로 주제에 게시

1. Sign in to the AWS Management Console and open the Amazon SNS console at https://console.aws.amazon.com/sns/.

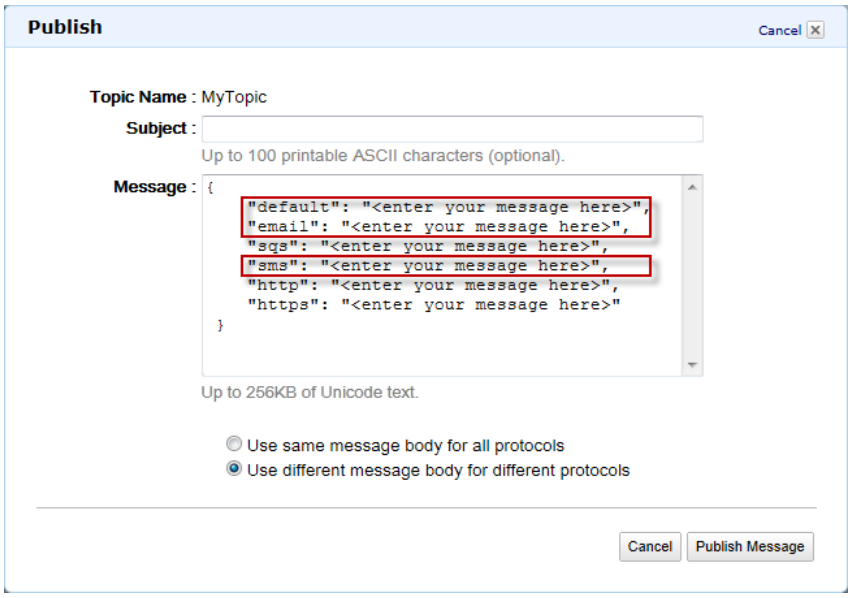
2. Navigation 창의 My Topics에서 게시하고자 하는 주제를 클릭합니다.
   Topic Details 페이지가 열립니다.
3. Publish to Topic 버튼을 클릭합니다.
   Publish 대화 상자가 표시됩니다.



4. Use different message body for different protocols를 선택합니다.



5. Subject 필드에 메시지의 제목을 입력합니다.
6. 관심 있는 각 프로토콜에 대해 Message 필드에 간단한 메시지를 입력합니다.
   다음의 예에서는 메시지가 기본, 이메일 및 SMS 프로토콜용으로 지정됩니다. 목록의 어떠한 프로토콜도 삭제하지 마십시오.



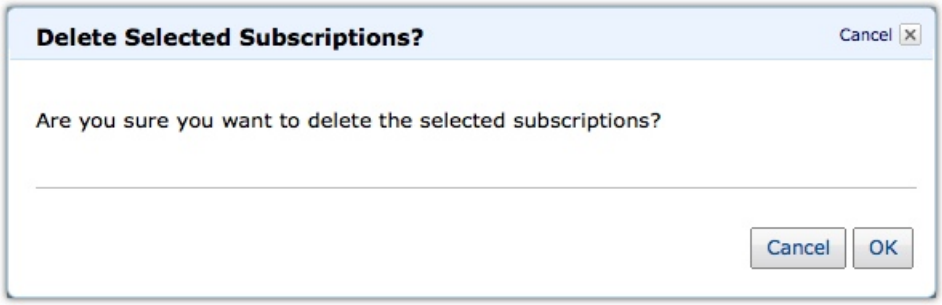
7. Publish Message를 클릭합니다.
   확인 대화 상자가 표시됩니다.
8. Close를 클릭하여 확인 대화 상자를 닫습니다.

정리

지금까지 주제를 생성하고 구독했으며 주제에 메시지를 게시했습니다. 이제 주제를 구독 해지 및 삭제함으로써 환경을 정리하겠습니다.

주제에서 구독 해지

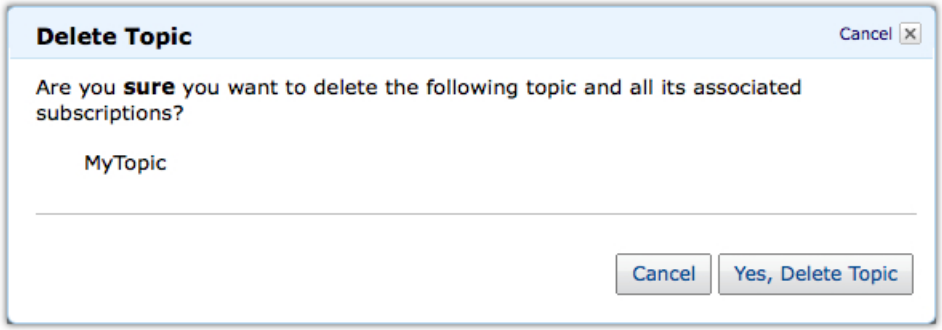
1. AWS Management Console을 열고 Navigation 창에서 My Subscriptions를 클릭합니다.
   My Subscriptions 페이지가 열립니다.
2. 구독 목록에서 주제 옆의 확인란을 클릭합니다. 하나의 구독만 생성한 경우, 페이지에는 이 목록만 나타날 것입니다.
3. Delete Subscriptions 버튼을 클릭합니다.
   Delete Selected Subscriptions? 확인 대화 상자가 나타납니다.



4. OK를 클릭합니다.

주제 삭제

1. Navigation 창의 My Topics에서 삭제하고자 하는 주제를 클릭합니다.
   Topic Details 페이지가 열립니다.
2. All Topic Actions 드롭다운 목록을 클릭하고 Delete Topic을 선택합니다.
   Delete Topic 확인 대화 상자가 표시됩니다.



3. Yes, Delete Topic을 클릭합니다.
   주제를 삭제하면, 해당 주제에 대한 모든 구독도 삭제됩니다.

Amazon SNS 주제에 대한 액세스 관리

Topics

Amazon SNS는 이메일 외에 다른 프로토콜을 지원합니다. HTTP, HTTPS 및 Amazon SQS 대기열을 사용할 수 있습니다. 사용자는 어떤 endpoint가 주제를 허용하도록 할지, 누가 주제를 게시할 수 있을지, 어떤 조건에서 게시할 수 있는지 등을 상세하게 규제하게 됩니다. 이 부록은 *액세스 제어 정책*을 통해 제어하는 방법을 설명합니다.

이 섹션의 주 내용은 규제를 평가하고 요청자에게 리소스에 대한 액세스 권한 부여 여부를 결정하기 위해 사용자가 이해해야 할 기본 개념, 정책 작성 방법 및 Amazon Web Services(AWS)가 사용하는 로직입니다. 이 섹션에 있는 대부분의 정보가 서비스와 상관이 없을지라도 사용자가 숙지해야 하는 몇 가지 Amazon SNS 특정 세부 정보가 있습니다. 자세한 내용은 Amazon SNS 정책에 대한 특별한 정보 (p. 37)을 참조하십시오.

Overview

Topics

This section describes basic concepts you need to understand to use the access policy language to write policies. It also describes the general process for how access control works with the access policy language, and how policies are evaluated.

When to Use Access Control

You have a great deal of flexibility in how you grant or deny access to a resource. However, the typical use cases are fairly simple:

- You want to grant another AWS account a particular type of topic action (e.g., Publish). For more information, see Allowing AWS account Access to a Topic (p. 21).
- You want to limit subscriptions to your topic to only the HTTPS protocol. For more information, see Limiting Subscriptions to HTTPS (p. 21).
- You want to allow Amazon SNS to publish messages to your Amazon SQS queue. For more information, see Publishing to an Amazon SQS Queue (p. 22).

Key Concepts

The following sections describe the concepts you need to understand to use the access policy language. They're presented in a logical order, with the first terms you need to know at the top of the list.

Permission

A *permission* is the concept of allowing or disallowing some kind of access to a particular resource. Permissions essentially follow this form: "A is/isn't allowed to do B to C where D applies." For example, *Jane* (A) has permission to *publish* (B) to *TopicA* (C) as long as *she uses the HTTP protocol* (D). Whenever Jane publishes to TopicA, the service checks to see if she has permission and if the request satisfies the conditions set forth in the permission.

Statement

A *statement* is the formal description of a single permission, written in the access policy language. You always write a statement as part of a broader container document known as a *policy* (see the next concept).

Policy

A *policy* is a document (written in the access policy language) that acts as a container for one or more statements. For example, a policy could have two statements in it: one that states that Jane can subscribe using the email protocol, and another that states that Bob cannot publish to TopicA. As shown in the following figure, an equivalent scenario would be to have two policies, one that states that Jane can subscribe using the email protocol, and another that states that Bob cannot publish to TopicA.

Issuer

The *issuer* is the person who writes a policy to grant permissions for a resource. The issuer (by definition) is always the resource owner. AWS does not permit AWS service users to create policies for resources they don't own. If John is the resource owner, AWS authenticates John's identity when he submits the policy he's written to grant permissions for that resource.

Principal

The *principal* is the person or persons who receive the permission in the policy. The principal is A in the statement "A has permission to do B to C where D applies." In a policy, you can set the principal to "anyone" (i.e., you can specify a wildcard to represent all people). You might do this, for example, if you don't want to restrict access based on the actual identity of the requester, but instead on some other identifying characteristic such as the requester's IP address.

Action

The *action* is the activity the principal has permission to perform. The action is B in the statement "A has permission to do B to C where D applies." Typically, the action is just the operation in the request to AWS. For example, Jane sends a request to Amazon SNS with `Action=Subscribe`. You can specify one or multiple actions in a policy.

Resource

The *resource* is the object the principal is requesting access to. The resource is C in the statement "A has permission to do B to C where D applies."

Conditions and Keys

The *conditions* are any restrictions or details about the permission. The condition is D in the statement "A has permission to do B to C where D applies." The part of the policy that specifies the conditions can be the most detailed and complex of all the parts. Typical conditions are related to:

- Date and time (e.g., the request must arrive before a specific day)
- IP address (e.g., the requester's IP address must be part of a particular CIDR range)

A *key* is the specific characteristic that is the basis for access restriction. For example, the date and time of request.

You use both *conditions* and *keys* together to express the restriction. The easiest way to understand how you actually implement a restriction is with an example: If you want to restrict access to before May 30,

2010, you use the condition called `DateLessThan`. You use the key called `aws:CurrentTime` and set it to the value `2010-05-30T00:00:00Z`. AWS defines the conditions and keys you can use. The AWS service itself (e.g., Amazon SQS or Amazon SNS) might also define service-specific keys. For more information about conditions, see Condition (p. 29). For more information about the available keys, see Available Keys (p. 31).

Requester

The *requester* is the person who sends a request to an AWS service and asks for access to a particular resource. The requester sends a request to AWS that essentially says: "Will you allow me to do B to C where D applies?"

Evaluation

Evaluation is the process the AWS service uses to determine if an incoming request should be denied or allowed based on the applicable policies. For information about the evaluation logic, see Evaluation Logic (p. 17).

Effect

The *effect* is the result that you want a policy statement to return at evaluation time. You specify this value when you write the statements in a policy, and the possible values are *deny* and *allow*.

For example, you could write a policy that has a statement that *denies* all requests that come from Antarctica (effect=deny given that the request uses an IP address allocated to Antarctica). Alternately, you could write a policy that has a statement that *allows* all requests that *don't* come from Antarctica (effect=allow, given that the request doesn't come from Antarctica). Although the two statements sound like they do the same thing, in the access policy language logic, they are different. For more information, see Evaluation Logic (p. 17).

Although there are only two possible values you can specify for the effect (allow or deny), there can be three different results at policy evaluation time: *default deny*, *allow*, or *explicit deny*. For more information, see the following concepts and Evaluation Logic (p. 17).

Default Deny

A *default deny* is the default result from a policy in the absence of an allow or explicit deny.

Allow

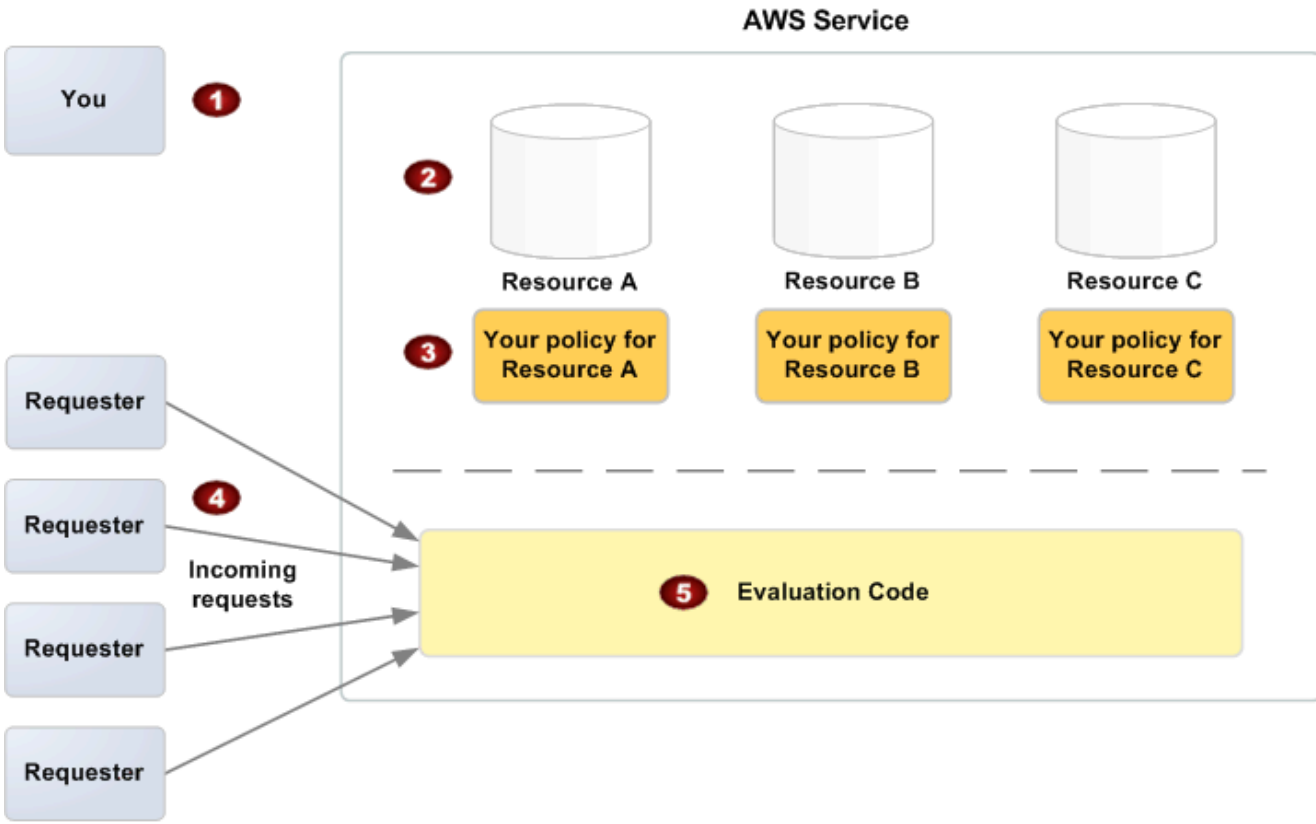
An *allow* results from a statement that has effect=allow, assuming any stated conditions are met. Example: Allow requests if they are received before 1:00 p.m. on April 30, 2010. An allow overrides all default denies, but never an explicit deny.

Explicit Deny

An *explicit deny* results from a statement that has effect=deny, assuming any stated conditions are met. Example: Deny all requests if they are from Antarctica. Any request that comes from Antarctica will always be denied no matter what any other policies might allow.

Architectural Overview

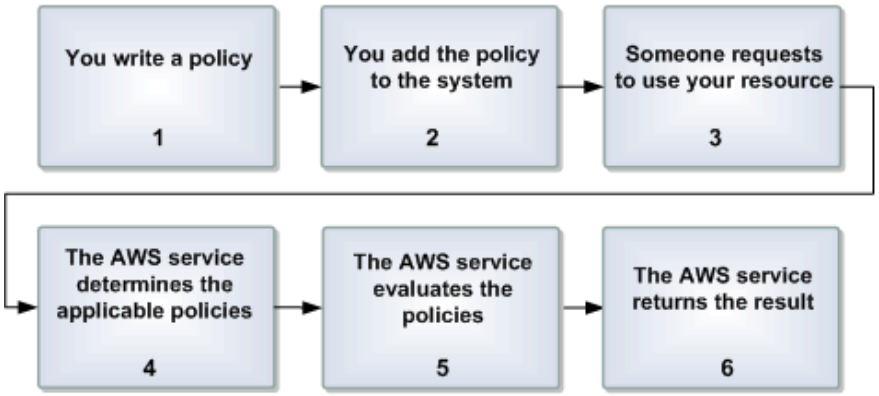
The following figure and table describe the main components that interact to provide access control for your resources.



1	You, the resource owner.
2	Your resources (contained within the AWS service; e.g., Amazon SQS queues).
3	Your policies. Typically you have one policy per resource, although you could have multiple. The AWS service itself provides an API you use to upload and manage your policies. For information about the content of the policies, see How to Write a Policy (p. 25).
4	Requesters and their incoming requests to the AWS service.
5	The access policy language evaluation code. This is the set of code within the AWS service that evaluates incoming requests against the applicable policies and determines whether the requester is allowed access to the resource. For information about how the service makes the decision, see Evaluation Logic (p. 17).

Using the Access Policy Language

The following figure and table describe the general process of how access control works with the access policy language.



Process for Using Access Control with the Access Policy Language

1	You write a policy for your resource. For example, you write a policy to specify permissions for your Amazon SNS topics. For more information, see How to Write a Policy (p. 25).
2	You upload your policy to AWS. The AWS service itself provides an API you use to upload your policies. For example, you use the Amazon SNS `SetTopicAttributes` action to upload a policy for a particular Amazon SNS topic.
3	Someone sends a request to use your resource. For example, a user sends a request to Amazon SNS to use one of your topics.
4	The AWS service determines which policies are applicable to the request. For example, Amazon SNS looks at all the available Amazon SNS policies and determines which ones are applicable (based on what the resource is, who the requester is, etc.).
5	The AWS service evaluates the policies. For example, Amazon SNS evaluates the policies and determines if the requester is allowed to use your topic or not. For information about the decision logic, see Evaluation Logic (p. 17).
6	The AWS service either denies the request or continues to process it. For example, based on the policy evaluation result, the service either returns an "Access denied" error to the requester or continues to process the request.

Related Topics

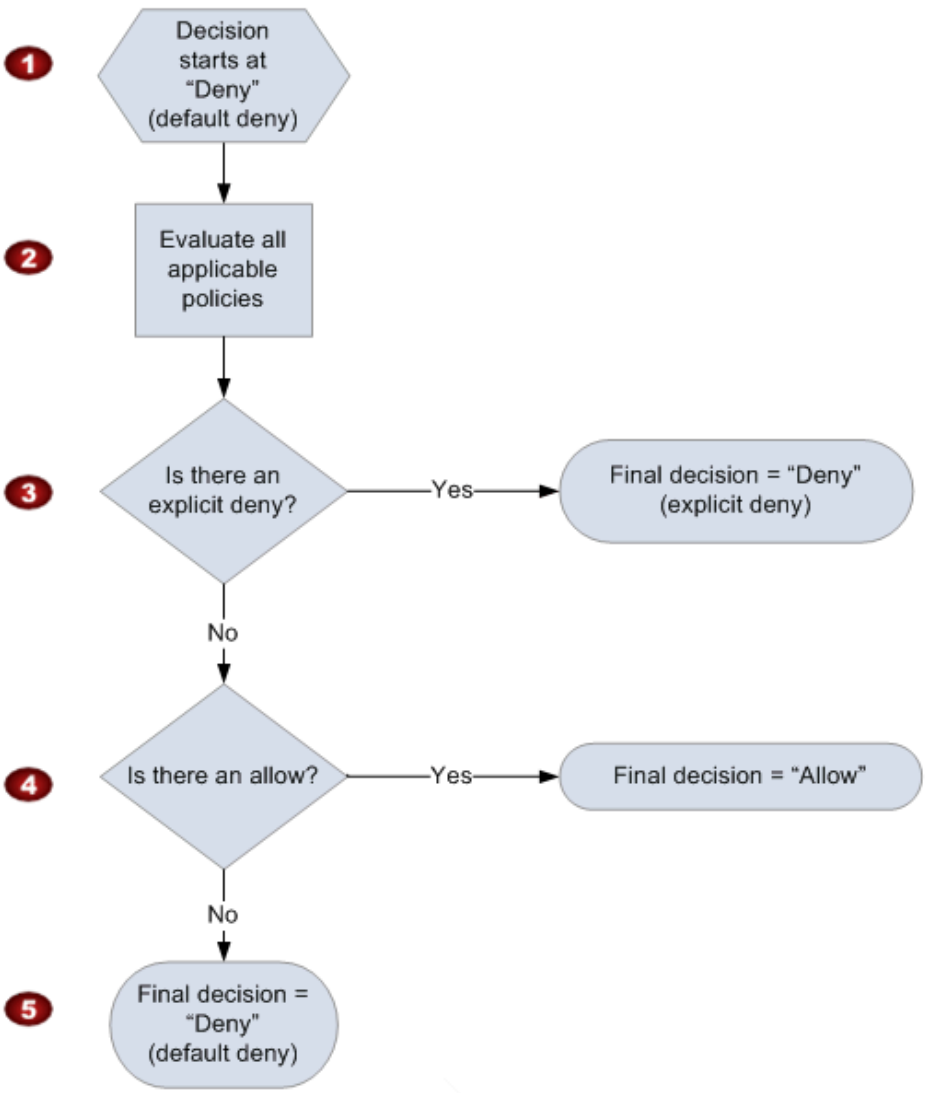
- Architectural Overview (p. 14)

Evaluation Logic

The goal at evaluation time is to decide whether a given request should be allowed or denied. The evaluation logic follows several basic rules:

- By default, all requests to use your resource coming from anyone but you are denied
- An allow overrides any default denies
- An explicit deny overrides any allows
- The order in which the policies are evaluated is not important

The following flow chart and discussion describe in more detail how the decision is made.



1	The decision starts with a default deny.

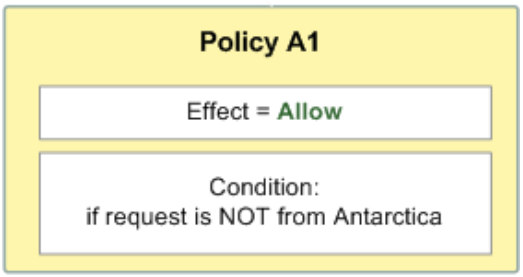
2	The enforcement code then evaluates all the policies that are applicable to the request (based on the resource, principal, action, and conditions). The order in which the enforcement code evaluates the policies is not important.
3	In all those policies, the enforcement code looks for an explicit deny instruction that would apply to the request. If it finds even one, the enforcement code returns a decision of "deny" and the process is finished (this is an explicit deny; for more information, see Explicit Deny (p. 14)).
4	If no explicit deny is found, the enforcement code looks for any "allow" instructions that would apply to the request. If it finds even one, the enforcement code returns a decision of "allow" and the process is done (the service continues to process the request).
5	If no allow is found, then the final decision is "deny" (because there was no explicit deny or allow, this is considered a *default deny* (for more information, see Default Deny (p. 14)).

The Interplay of Explicit and Default Denials

A policy results in a default deny if it doesn't directly apply to the request. For example, if a user requests to use Amazon SNS, but the policy on the topic doesn't refer to the user's AWS account at all, then that policy results in a default deny.

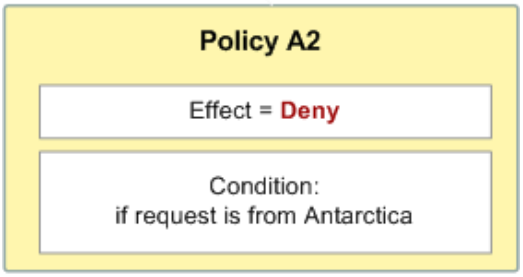
A policy also results in a default deny if a condition in a statement isn't met. If all conditions in the statement are met, then the policy results in either an allow or an explicit deny, based on the value of the Effect element in the policy. Policies don't specify what to do if a condition isn't met, and so the default result in that case is a default deny.

For example, let's say you want to prevent requests coming in from Antarctica. You write a policy (called Policy A1) that allows a request only if it doesn't come from Antarctica. The following diagram illustrates the policy.



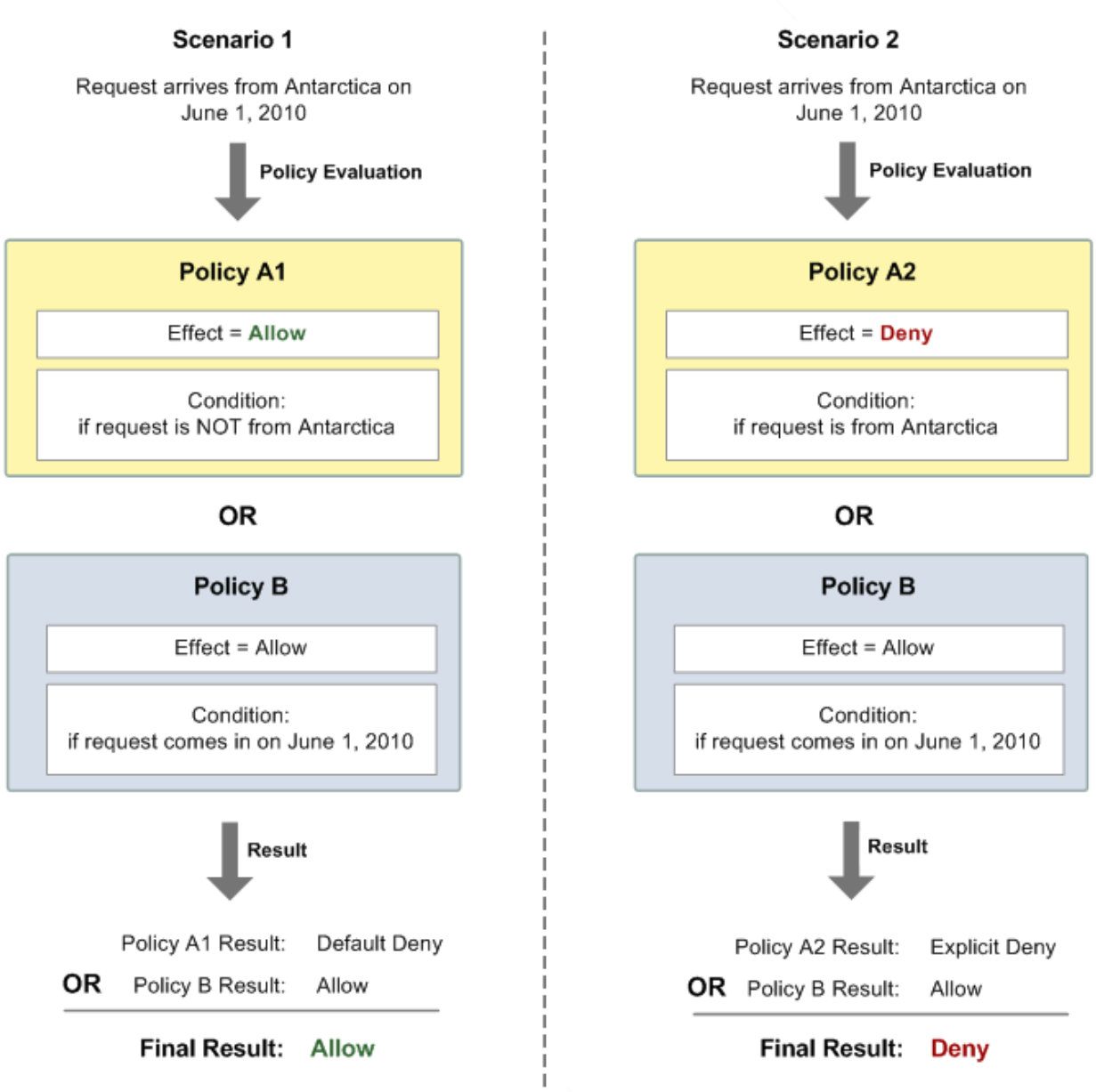
If someone sends a request from the U.S., the condition is met (the request is not from Antarctica). Therefore, the request is allowed. But, if someone sends a request from Antarctica, the condition isn't met, and the policy's result is therefore a default deny.

You could turn the result into an explicit deny by rewriting the policy (named Policy A2) as in the following diagram. Here, the policy explicitly denies a request if it comes from Antarctica.



If someone sends a request from Antarctica, the condition is met, and the policy's result is therefore an explicit deny.

The distinction between a default deny and an explicit deny is important because a default deny can be overridden by an allow, but an explicit deny can't. For example, let's say there's another policy that allows requests if they arrive on June 1, 2010. How does this policy affect the overall outcome when coupled with the policy restricting access from Antarctica? We'll compare the overall outcome when coupling the date-based policy (we'll call Policy B) with the preceding policies A1 and A2. Scenario 1 couples Policy A1 with Policy B, and Scenario 2 couples Policy A2 with Policy B. The following figure and discussion show the results when a request comes in from Antarctica on June 1, 2010.



In Scenario 1, Policy A1 returns a default deny, as described earlier in this section. Policy B returns an allow because the policy (by definition) allows requests that come in on June 1, 2010. The allow from Policy B overrides the default deny from Policy A1, and the request is therefore allowed.

In Scenario 2, Policy B2 returns an explicit deny, as described earlier in this section. Again, Policy B returns an allow. The explicit deny from Policy A2 overrides the allow from Policy B, and the request is therefore denied.

Example Cases for Amazon SNS Access Control

Topics

This section gives a few examples of typical use cases for access control.

Allowing AWS account Access to a Topic

Let's say you have a topic in the Amazon SNS system. In the simplest case, you want to allow one or more AWS accounts access to a specific topic action (e.g., Publish).

You can do this by using the Amazon SNS API action `AddPermission`. It takes a topic, a list of AWS account IDs, a list of actions, and a label, and automatically creates a new statement in the topic's access control policy. In this case, you don't write a policy yourself, because Amazon SNS automatically generates the new policy statement for you. You can remove the policy statement later by calling `RemovePermission` with its label.

For example, if you called `AddPermission` on the topic arn:aws:sns:us-east-1:444455556666:MyTopic, with AWS account ID 1111-2222-3333, the `Publish` action, and the label `give-1234-publish`, Amazon SNS would generate and insert the following access control policy statement:

```
{
    "Version":"2012-10-17",
    "Id":"AWSAccountTopicAccess",
    "Statement" : [
        {
            "Sid":"give-1234-publish",
            "Effect":"Allow",
            "Principal" : {
                "AWS": "111122223333"
             },
            "Action":["sns:Publish"],
            "Resource": "arn:aws:sns:us-east-1:444455556666:MyTopic"
        }
    ]
}
```

Once this statement is added, the user with AWS account 1234-5678-9012 can publish messages to the topic.

Limiting Subscriptions to HTTPS

In this use case, you want to allow subscription requests to your topic *only by HTTPS*, for security.

You need to know how to write your own policy for the topic because the Amazon SNS `AddPermission` action doesn't let you specify a protocol restriction when granting someone access to your topic. In this case, you would write your own policy, and then use the `SetTopicAttributes` action to set the topic's `Policy` attribute to your new policy.

The following example of a full policy gives the AWS account ID 1234-5678-9012 the ability to subscribe to notifications from a topic.

Note

`Subscribe` and `Receive` are separate actions in the policy. You can apply different conditions to the subscriber and the message recipient.

```
{
    "Version":"2012-10-17",
    "Id":"SomePolicyId",
    "Statement" : [
        {
            "Sid":"Statement1",
            "Effect":"Allow",
            "Principal" : {
                "AWS": "111122223333"
              },
            "Action":["sns:Subscribe"],
            "Resource": "arn:aws:sns:us-east-1:444455556666:MyTopic",
            "Condition" : {
                "StringEquals" : {
                    "sns:Protocol":"https"
                 }
            }
        }
    ]
}
```

Publishing to an Amazon SQS Queue

In this use case, you want to publish messages from your topic to your Amazon SQS queue. Like Amazon SNS, Amazon SQS uses Amazon's access control policy language. To allow Amazon SNS to send messages, you'll need to use the Amazon SQS action `SetQueueAttributes` to set a policy on the queue.

Again, you'll need to know how to write your own policy because the Amazon SQS `AddPermission` action doesn't create policy statements with conditions.

Note that the example presented below is an Amazon SQS policy (controlling access to your queue), not an Amazon SNS policy (controlling access to your topic). The actions are Amazon SQS actions, and the resource is the Amazon Resource Name (ARN) of the queue. You can determine the queue's ARN by retrieving the queue's `QueueArn` attribute with the `GetQueueAttributes` action.

```
{
    "Version":"2012-10-17",
    "Id":"MyQueuePolicy",
    "Statement" : [
        {
         "Sid":"Allow-SNS-SendMessage",
         "Effect":"Allow",
         "Principal" : {
             "AWS": "*"
          },
         "Action":["sqs:SendMessage"],
         "Resource": "arn:aws:sqs:us-east-1:444455556666:MyQueue",
         "Condition" : {
             "ArnEquals" : {
              "aws:SourceArn":"arn:aws:sns:us-east-1:444455556666:MyTopic"
                 }
            }
        }
    ]
}
```

This policy uses the `aws:SourceArn` condition to restrict access to the queue based on the source of the message being sent to the queue. You can use this type of policy to allow Amazon SNS to send messages to your queue only if the messages are coming from one of your own topics. In this case, you specify a particular one of your topics, whose ARN is arn:aws:sns:us-east-1:444455556666:MyTopic.

The preceding policy is an example of the Amazon SQS policy you could write and add to a specific queue. It would grant Amazon SNS and other AWS products access. Amazon SNS gives a default policy to all newly created topics. The default policy gives all other AWS products access to your topic. This default policy uses an `aws:SourceArn` condition to ensure that AWS products access your topic only on behalf of AWS resources you own.

Allowing Any AWS Resource to Publish to a Topic

In this case, you want to configure a topic's policy so that another AWS account's resource (e.g., Amazon S3 bucket, Amazon EC2 instance, or Amazon SQS queue) can publish to your topic. This example assumes that you write your own policy and then use the `SetTopicAttributes` action to set the topic's `Policy` attribute to your new policy.

In the following example statement, the topic owner in these policies is 1111-2222-3333 and the AWS resource owner is 4444-5555-6666. The example gives the AWS account ID 4444-5555-6666 the ability to publish to My-Topic from any AWS resource owned by the account.

```
{
    "Version":"2012-10-17",
    "Id":"MyAWSPolicy",
    "Statement" : [
        {
            "Sid": "My-statement-id",
            "Effect": "Allow",
            "Principal" : { "AWS": "*" },
            "Action": "sns:Publish",
            "Resource": "arn:aws:sns:us-east-1:111122223333:My-Topic",
            "Condition": {
               "StringEquals": {
                  "AWS:SourceOwner":"444455556666"
                }
            }
        }
    ]
}
```

Allowing an Amazon S3 Bucket to Publish to a Topic

In this case, you want to configure a topic's policy so that another AWS account's Amazon S3 bucket can publish to your topic. For more information about publishing notifications from Amazon S3, go to Setting Up Notifications of Bucket Events.

This example assumes that you write your own policy and then use the `SetTopicAttributes` action to set the topic's `Policy` attribute to your new policy.

The following example statement uses the `ArnLike` condition to make sure the ARN of the resource making the request (the `AWS:SourceARN`) is an Amazon S3 ARN. You could use a similar condition to restrict the permission to a set of Amazon S3 buckets, or even to a specific bucket. In this example, the topic owner is 1111-2222-3333 and the Amazon S3 owner is 4444-5555-6666. The example states that any Amazon S3 bucket owned by 4444-5555-6666 is allowed to publish to My-Topic.

```
{
    "Version":"2012-10-17",
    "Id":"MyAWSPolicy",
    "Statement" : [
        {
            "Sid": "My-statement-id",
            "Effect": "Allow",
            "Principal" : { "AWS": "*" },
            "Action": "sns:Publish",
            "Resource": "arn:aws:sns:us-east-1:111122223333:My-Topic",
            "Condition": {
                "StringEquals": { "AWS:SourceOwner":"444455556666" } ,
                "ArnLike": { "AWS:SourceArn": "arn:aws:s3:*:*:*" }
            }
        }
    ]
}
```

How to Write a Policy

Topics

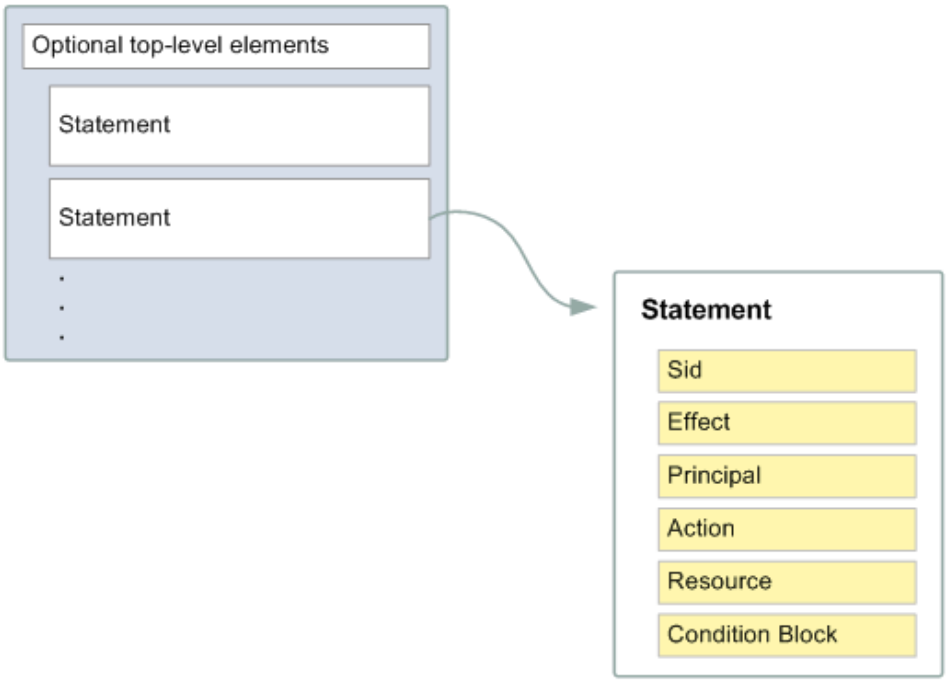
This section describes how to write policies and gives reference information about each policy element.

Basic Policy Structure

Each policy is a JSON document. As illustrated in the following figure, a policy includes:

- Optional policy-wide information (at the top of the document)
- One or more individual *statements*

Each statement includes the core information about a single permission. If a policy includes multiple statements, we apply a logical OR across the statements at evaluation time. If multiple policies are applicable to a request, we apply a logical OR across the policies at evaluation time.



The information in a statement is contained within a series of *elements*. For information about these elements, see Element Descriptions (p. 25).

Element Descriptions

Topics

This section describes the elements you can use in a policy and its statements. The elements are listed here in the general order you use them in a policy. The `Id`, `Version`, and `Statement` are top-level policy elements; the rest are statement-level elements. JSON examples are provided.

All elements are optional for the purposes of parsing the policy document itself. The order of the elements doesn't matter (e.g., the `Resource` element can come before the `Action` element). You're not required to specify any Conditions in the policy.

Version

The `Version` element specifies the access policy language version. The only allowed values are these:

- `2012-10-17`. This is the current version of the policy language, and you should use this version number for all policies.
- `2008-10-17`. This was an earlier version of the policy language. You might see this version on existing policies. Do not use this version for any new policies or any existing policies that you are updating.

If you do not include a `Version` element, the value defaults to `2008-10-17`. However, it is a good practice to always include a `Version` element and set it to `2012-10-17`.

Note

If your policy includes *policy variables*, you *must* include a `Version` element and set it to `2012-10-17`. If you don't include a `Version` element set to `2012-10-17`, variables such as `${aws:username}` won't be recognized as variables and will instead be treated as literal strings in the policy.

```
"Version":"2012-10-17"
```

Id

The `Id` is an optional identifier for the policy. We recommend you use a UUID for the value, or incorporate a UUID as part of the ID to ensure uniqueness.

Important

The AWS service (e.g., Amazon SNS) implementing the access policy language might require this element and have uniqueness requirements for it. For service-specific information about writing policies, see Amazon SNS 정책에 대한 특별한 정보 (p. 37).

```
"Id":"cd3ad3d9-2776-4ef1-a904-4c229d1642ee"
```

Statement

The `Statement` is the main element for a statement. It can include multiple elements (see the subsequent sections in this guide).

The `Statement` element contains an array of individual statements. Each individual statement is a distinct JSON block enclosed in curly brackets { }.

```
"Statement":[{...},{...},{...}]
```

Sid

The `Sid` (statement ID) is an optional identifier you provide for the policy statement. Essentially it is just a sub-ID of the policy document's ID.

Important

The AWS service (e.g., Amazon SNS) implementing the access policy language might require this element and have uniqueness requirements for it. For service-specific information about writing policies, see Amazon SNS 정책에 대한 특별한 정보 (p. 37).

```
"Sid" : "1"
```

Effect

The `Effect` is a required element that indicates whether you want the statement to result in an allow or an explicit deny (for more information, see Explicit Deny (p. 14)).

Valid values for `Effect` are `Allow` and `Deny`.

```
"Effect":"Allow"
```

Principal

The `Principal` is the person or persons who receive or are denied permission according to the policy. You must specify the principal by using the principal's AWS account ID (e.g., 1234-5678-9012, with or without the hyphens). You can specify multiple principals, or a wildcard (*) to indicate all possible users. You can view your account ID by logging in to your AWS account at http://aws.amazon.com and clicking Account Activity.

In JSON, you use `"AWS":` as a prefix for the principal's AWS account ID. In the following example, two principals are included in the statement.

```
"Principal": {
 "AWS":["123456789012", "AWS":"999999999999"]}
```

NotPrincipal

The `NotPrincipal` element lets you specify an exception to a list of principals. For example, you can use this to prevent all AWS accounts *except* a specific account from accessing a resource. Conversely, you can deny access to all principals except the one named in the `NotPrincipal` element. As with `Principal`, you specify the user or account that should be allowed or denied permission; the difference is that `NotPrincipal` translates to everyone *except* that person or account.

In the following example, all users are denied access to a resource except for the user named Bob in the AWS account 123456789012.

```
"Effect": "Deny",
"NotPrincipal": {
    "AWS":"arn:aws:iam::123456789012:user/Bob"}
```

Action

The `Action` is the specific type or types of access allowed or denied (for example, read or write). You can specify multiple values for this element. The values are free-form but must match values the AWS service expects (for more information, see Amazon SNS 정책에 대한 특별한 정보 (p. 37)). You can use a wildcard (*) to give the principal access to all the actions the specific AWS service lets you share with other developers.

```
"Action":["sns:Publish","sns:Subscribe"]
```

The prefix and the action name are case insensitive. For example, `sns:Subscribe` is equivalent to `SNS:subscribe`.

NotAction

The `NotAction` element matches everything except the specified action. This is useful if you want to make an exception to a list of actions being allowed or denied. The example below matches any action, except `Publish`.

```
"NotAction":"sns:Publish"
```

Resource

The `Resource` is the object or objects the policy covers. You specify the resource using the following *Amazon Resource Name (ARN)* format.

```
arn:aws:<vendor>:<region>:<namespace>:<relative-id>
```

Where:

- `vendor` identifies the AWS product (e.g., Amazon SNS).
- `region` is the region the resource resides in (e.g., us-east-1), if any.
- `namespace` is the AWS account ID with no hyphens (e.g., 123456789012).
- `relative-id` is the portion that identifies the specific resource.

NotResource

The `NotResource` element is useful if you want to make an exception to a list of resources. You could use this, for example, if you want your users to be able to access a specific Amazon SNS topic belonging to the AWS account. If the AWS account were to create a new topic for the company, an admin wouldn't have to update the policy with the new topic's name in order to prevent users from being able to use the topic. By default, the users wouldn't be able to use it.

The following example refers to all resources *other* than your company's topic called my_corporate_topic. You would use this in a policy with `"Effect":"Deny"` to keep users from accessing any queue besides my_corporate_topic.

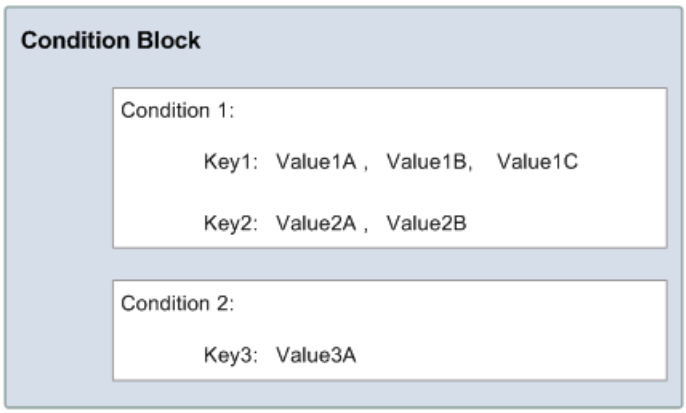
```
"NotResource":"arn:aws:sqs:*:123456789012:my_corporate_topic"
```

Condition

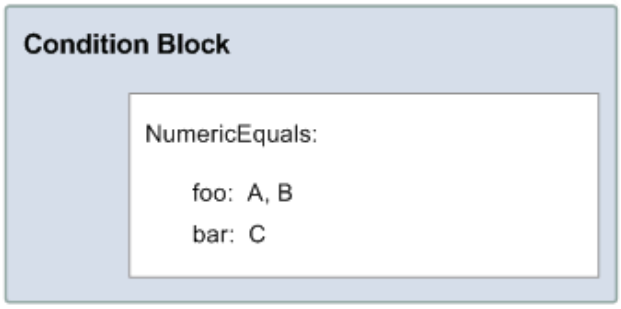
This section describes the `Condition` element and the information you can use inside the element.

The Condition Block

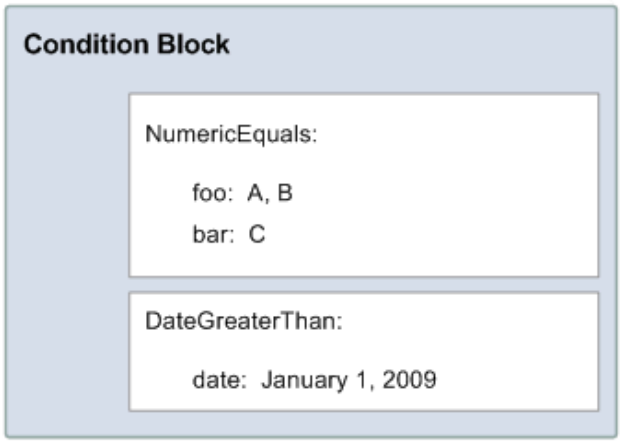
The `Condition` element is the most complex part of the policy statement. We refer to it as the *condition block*, because although it has a single `Condition` element, it can contain multiple conditions, and each condition can contain multiple key-value pairs. The following figure illustrates this. Unless otherwise specified for a particular key, all keys can have multiple values.



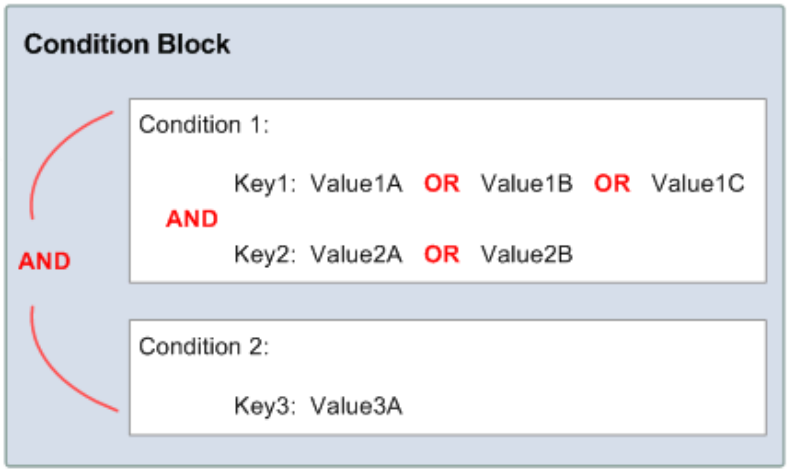
When creating a condition block, you specify the name of each condition, and at least one key-value pair for each condition. AWS defines the conditions and keys you can use (they're listed in the subsequent sections). An example of a condition is `NumericEquals`. Let's say you have a fictional resource, and you want to let John use it only if some particular numeric value *foo* equals either A or B, and another numeric value *bar* equals C. Then you would create a condition block that looks like the following figure.



Let's say you also want to restrict John's access to after January 1, 2009. Then you would add another condition, `DateGreaterThan`, with a date equal to January 1, 2009. The condition block would then look like the following figure.



As illustrated in the following figure, we always apply a logical `AND` to the conditions within a condition block, and to the keys within a condition. We always apply a logical `OR` to the values for a single key. All conditions must be met to return an allow or an explicit deny decision. If a condition isn't met, the result is a default deny.



As mentioned, AWS defines the conditions and keys you can use (for example, one of the keys is `aws:CurrentTime`, which lets you restrict access based on the date and time). The AWS service itself can also define its own service-specific keys. For a list of available keys, see Available Keys (p. 31).

For a concrete example that uses real keys, let's say you want to let John publish to your topic under the following three conditions:

- The time is after 12:00 noon on 8/16/2010
- The time is before 3:00 p.m. on 8/16/2010
- The request comes from an IP address within the 192.168.176.0/24 range or the 192.168.143.0/24 range

Your condition block has three separate conditions, and all three of them must be met for John to have access to your topic.

The following shows what the condition block looks like in your policy.

```
"Condition" :  {
      "DateGreaterThan" : {
         "aws:CurrentTime" : "2009-04-16T12:00:00Z"
       },
      "DateLessThan": {
         "aws:CurrentTime" : "2009-04-16T15:00:00Z"
       },
       "IpAddress" : {
          "aws:SourceIp" : ["192.168.176.0/24","192.168.143.0/24"]
      }
}
```

Available Keys

AWS provides a set of common keys supported by all AWS services that adopt the access policy language for access control. These keys are:

- `aws:CurrentTime`–For date/time conditions (see Date Conditions (p. 33))
- `aws:EpochTime`–The date in epoch or UNIX time, for use with date/time conditions (see Date Conditions (p. 33))
- `aws:MultiFactorAuthAge`–Key that provides a numeric value indicating how long ago (in seconds) the MFA-validated security credentials making the request were issued using Multi-Factor Authentication (MFA). Unlike other keys, if MFA is not used successfully, this key is not present (see Existence of Condition Keys (p. 35), Numeric Conditions (p. 33) and Using Multi-Factor Authentication (MFA) Devices with AWS).
- `aws:principaltype`–To check the type of principal (user, account, federated user, etc.) for the current request (see String Conditions (p. 32)).
- `aws:SecureTransport`–Boolean representing whether the request was sent using SSL (see Boolean Conditions (p. 34))
- `aws:SourceArn`–The Amazon Resource Name (ARN) of the source (see Amazon Resource Name (ARN) (p. 34))
- `aws:SourceIp`–The requester's IP address, for use with IP address conditions (see IP Address (p. 34))
- `aws:UserAgent`–Information about the requester's client application, for use with string conditions (see String Conditions (p. 32))
- `aws:userid`–To check the requester's user ID (see String Conditions (p. 32)).
- `aws:username`–To check the requester's user name (see String Conditions (p. 32)).

The key names are case insensitive. For example, `aws:CurrentTime` is equivalent to `AWS:currenttime`.

Note

If you use `aws:SourceIp`, and the request comes from an Amazon EC2 instance, we evaluate the instance's public IP address to determine if access is allowed.

Each AWS service that uses the access policy language might also provide service-specific keys. For a list of any service-specific keys you can use, see Amazon SNS 정책에 대한 특별한 정보 (p. 37).

Condition Types

These are the general types of conditions you can specify:

- String
- Numeric
- Date and time
- Boolean
- IP address
- Amazon Resource Name (ARN)
- Existence of condition keys

String Conditions

String conditions let you constrain using string matching rules. The actual data type you use is a string.

Condition	Description
`StringEquals`	Strict matching Short version: `streq`
`StringNotEquals`	Strict negated matching Short version: `strneq`
`StringEqualsIgnoreCase`	Strict matching, ignoring case Short version: `streqi`
`StringNotEqualsIgnoreCase`	Strict negated matching, ignoring case Short version: `strneqi`
`StringLike`	Loose case-sensitive matching. The values can include a multi-character match wildcard (*) or a single-character match wildcard (?) anywhere in the string. Short version: `strl`
`StringNotLike`	Negated loose case-insensitive matching. The values can include a multi-character match wildcard (*) or a single-character match wildcard (?) anywhere in the string. Short version: `strnl`

Numeric Conditions

Numeric conditions let you constrain using numeric matching rules. You can use both whole integers or decimal numbers. Fractional or irrational syntax is not supported.

Condition	Description
`NumericEquals`	Strict matching Short version: `numeq`
`NumericNotEquals`	Strict negated matching Short version: `numneq`
`NumericLessThan`	"Less than" matching Short version: `numlt`
`NumericLessThanEquals`	"Less than or equals" matching Short version: `numlteq`
`NumericGreaterThan`	"Greater than" matching Short version: `numgt`
`NumericGreaterThanEquals`	"Greater than or equals" matching Short version: `numgteq`

Date Conditions

Date conditions let you constrain using date and time matching rules. You must specify all date/time values with one of the W3C implementations of the ISO 8601 date formats (for more information, go to http://www.w3.org/TR/NOTE-datetime). You use these conditions with the `aws:CurrentTime` key or the `aws:EpochTime` key to restrict access based on request time.

Note

Wildcards are not permitted for date conditions.

Condition	Description
`DateEquals`	Strict matching Short version: `dateeq`
`DateNotEquals`	Strict negated matching Short version: `dateneq`
`DateLessThan`	A point in time at which a key stops taking effect Short version: `datelt`
`DateLessThanEquals`	A point in time at which a key stops taking effect Short version: `datelteq`
`DateGreaterThan`	A point in time at which a key starts taking effect Short version: `dategt`

Condition	Description
`DateGreaterThanEquals`	A point in time at which a key starts taking effect Short version: `dategteq`

Boolean Conditions

Condition	Description
`Bool`	Strict Boolean matching

IP Address

IP address conditions let you constrain based on IP address matching rules. You use these with the `aws:SourceIp` key. The value must be in the standard CIDR format (for example, 10.52.176.0/24). For more information, go to RFC 4632.

Condition	Description
`IpAddress`	Approval based on the IP address or range
`NotIpAddress`	Denial based on the IP address or range

Amazon Resource Name (ARN)

Amazon Resource Name (ARN) conditions let you constrain based on ARN matching rules. The actual data type you use is a string.

Condition	Description
`ArnEquals`	Strict matching for ARN
`ArnNotEquals`	Strict negated matching for ARN
`ArnLike`	Loose case-insensitive matching of the ARN. Each of the six colon-delimited components of the ARN is checked separately and each can include a multi-character match wildcard (*) or a single-character match wildcard (?).
`ArnNotLike`	Negated loose case-insensitive matching of the ARN. The values can include a multi-character match wildcard (*) or a single-character match wildcard (?) anywhere in the string.

Following is an example of the kind of policy you need to attach to any queue that you want Amazon SNS to send messages to. It gives Amazon SNS permission to send messages to the queue (or queues) of your choice, but only if the service is sending the messages on behalf of a particular Amazon SNS topic (or topics). You specify the queue in the `Resource` field, and the Amazon SNS topic as the value for the `SourceArn` key.

```
{
   "Statement":[{
      "Effect": "Allow",
      "Principal": {
```

```
         "AWS": "210987654321"
      },
      "Action": "sqs:SendMessage",
      "Resource": "arn:aws:sqs:us-east-1:01234567891:your_queue_xyz",
      "Condition" : {
         "ArnEquals" : {
          "aws:SourceArn":"arn:aws:sns:us-east-1:123456789012:your_special_top
ic_1"}
            }
      }
   ]
   }
```

Existence of Condition Keys

Use a `Null` condition to check if a condition key is present at the time of authorization. In the policy statement, use either `true` (the key doesn't exist) or `false` (the key exists and its value is not null). You can use this condition to determine if a user has authenticated with MFA (multi-factor authentication). For example, the following condition states that MFA must exist (be *not null*) for the user to use the Amazon EC2 API.

```
{
  "Statement":[{
      "Action":["ec2:*"],
      "Effect":"Allow",
      "Resource":["*"],
      "Condition":{
         "Null":{"aws:MultiFactorAuthAge":"false"}
      }
    }
  ]
}
```

Supported Data Types

This section lists the set of data types the access policy language supports. The language doesn't support all types for each policy element (for the supported data types for each element, see Element Descriptions (p. 25)).

The access policy language supports the following data types:

- Strings
- Numbers (Ints and Floats)
- Boolean
- Null
- Lists
- Maps
- Structs (which are just nested Maps)

The following table maps each data type to the serialization. Note that all policies must be in UTF-8. For information about the JSON data types, go to RFC 4627.

Type	JSON
String	String
Integer	Number
Float	Number
Boolean	true false
Null	null
Date	String adhering to the W3C Profile of ISO 8601
IpAddress	String adhering to RFC 4632
List	Array
Object	Object

Amazon SNS 정책에 대한 특별한 정보

다음의 목록은 액세스 제어의 Amazon SNS 구현에 대한 특정 정보를 제공합니다.

- 각각의 정책은 하나의 주제 또는 대기열에만 적용(규제 작성 시 다른 주제 또는 대기열에 적용하는 문구를 포함하지 마십시오)해야 합니다.
- 각각의 정책에는 고유한 규제가 있어야 합니다. `Id`
- 정책의 각 문구에는 고유한 문구가 있어야 합니다. `sid`

Amazon SNS 정책 제한

다음의 표는 정책 정보의 최대 제한을 나열한 것입니다.

이름	최대 제한
바이트	20kb
Statement	20
Principal	20
리소스	1 (값은 정책의 주제 ARN과 일치해야 함)

유효한 Amazon SNS 정책 작업

Amazon SNS는 다음의 표에 표시된 작업을 지원합니다.

작업	설명
sns:AddPermission	주제 정책에 대한 권한을 추가할 권한을 부여합니다.
sns:CreatePlatformApplication	APNS 및 GCM과 같이 지원되는 푸시 알림 서비스 중 하나에 플랫폼 애플리케이션 객체를 생성할 권한을 부여합니다. 자세한 내용은 Amazon Simple Notification Service API Reference의 CreatePlatformApplication을 참조하십시오.
sns:CreatePlatformEndpoint	지원되는 푸시 알림 서비스 중 하나에 디바이스 및 모바일 앱용 endpoint를 생성할 권한을 부여합니다. 자세한 내용은 Amazon Simple Notification Service API Reference의 CreatePlatformEndpoint를 참조하십시오.
sns:DeleteEndpoint	지원되는 푸시 알림 서비스 중 하나에 디바이스 및 모바일 앱용 endpoint에 권한을 부여합니다. 자세한 내용은 Amazon Simple Notification Service API Reference의 DeleteEndpoint를 참조하십시오.
sns:DeletePlatformApplication	플랫폼 애플리케이션 객체를 삭제할 권한을 부여합니다. 자세한 내용은 Amazon Simple Notification Service API Reference의 DeletePlatformApplication을 참조하십시오.
sns:DeleteTopic	주제를 삭제할 권한을 부여합니다.
sns:GetEndpointAttributes	디바이스 및 모바일 앱용 endpoint 속성을 검색할 권한을 부여합니다. 자세한 내용은 Amazon Simple Notification Service API Reference의 GetEndpointAttributes를 참조하십시오.

작업	설명
sns:GetPlatformApplicationAttributes	플랫폼 애플리케이션 객체의 속성을 검색할 권한을 부여합니다. 자세한 내용은 Amazon Simple Notification Service API Reference의 GetPlatformApplicationAttributes를 참조하십시오.
sns:GetTopicAttributes	모든 주제 속성을 수신할 권한을 부여합니다.
sns:ListEndpointsByPlatformApplication	지원되는 푸시 알림 서비스에서 디바이스 및 모바일 앱용 endpoint 및 endpoint 속성을 나열할 권한을 부여합니다. 자세한 내용은 Amazon Simple Notification Service API Reference의 ListEndpointsByPlatformApplication을 참조하십시오.
sns:ListPlatformApplications	지원되는 푸시 알림 서비스에 대해 플랫폼 애플리케이션 객체를 나열할 권한을 부여합니다. 자세한 내용은 Amazon Simple Notification Service API Reference의 ListPlatformApplications를 참조하십시오.
sns:ListSubscriptionsByTopic	특정 주제에 대한 모든 구독을 검색할 권한을 부여합니다.
sns:Publish	주제 또는 endpoint를 게시할 권한을 부여합니다. 자세한 내용은 Amazon Simple Notification Service API Reference의 Publish를 참조하십시오.
sns:Receive	주제의 알림을 수신할 권한을 부여합니다.
sns:RemovePermission	주제 정책에서 어떠한 권한을 제거할 권한을 부여합니다.
sns:SetEndpointAttributes	디바이스 및 모바일 앱용 endpoint에 대한 속성을 설정할 권한을 부여합니다. 자세한 내용은 Amazon Simple Notification Service API Reference의 SetEndpointAttributes를 참조하십시오.
sns:SetPlatformApplicationAttributes	플랫폼 애플리케이션 객체의 속성을 설정할 권한을 부여합니다. 자세한 내용은 Amazon Simple Notification Service API Reference의 SetPlatformApplicationAttributes를 참조하십시오.
sns:SetTopicAttributes	주제의 속성을 설정할 권한을 부여합니다.
sns:Subscribe	주제를 구독할 권한을 부여합니다.

Amazon SNS 키

Amazon SNS는 다음의 서비스 지정 키를 사용합니다. `Subscribe` 요청 및 `Receive` 요청에 대한 액세스를 제한하는 다음의 정책을 사용할 수 있습니다.

- sns:Endpoint – `Subscribe` 요청 또는 기존에 확인된 구독의 URL, 이메일 주소 또는 ARN입니다. 특정 endpoint에 대한 액세스를 제한하는 스트링 조건(String Conditions (p. 32) 참조)을 사용합니다(예. *@example.com).
- sns:Protocol – `protocol`값(`Subscribe` 요청 또는 기존에 확인된 구독) 입니다. 특정 전송 프로토콜에 대한 액세스를 제한하는 스트링 조건(String Conditions (p. 32) 참조)을 사용합니다(예. http).

Important

sns:Endpoint로 액세스를 규제하는 정책 사용 시 DNS 문제가 향후 endpoint 이름 확인에 영향을 미칠 수 있습니다.

Controlling User Access to Your AWS Account

Topics

Amazon Simple Notification Service integrates with AWS Identity and Access Management (IAM) so that you can specify which Amazon SNS actions a user in your AWS account can perform with Amazon SNS resources. You can specify a particular topic in the policy. For example, you could use variables when creating an IAM policy that gives certain users in your organization permission to use the `Publish` action with specific topics in your AWS account. For more information, see Policy Variables in the *Using IAM* guide.

Important

Using Amazon SNS with IAM doesn't change how you use Amazon SNS. There are no changes to Amazon SNS actions, and no new Amazon SNS actions related to users and access control.

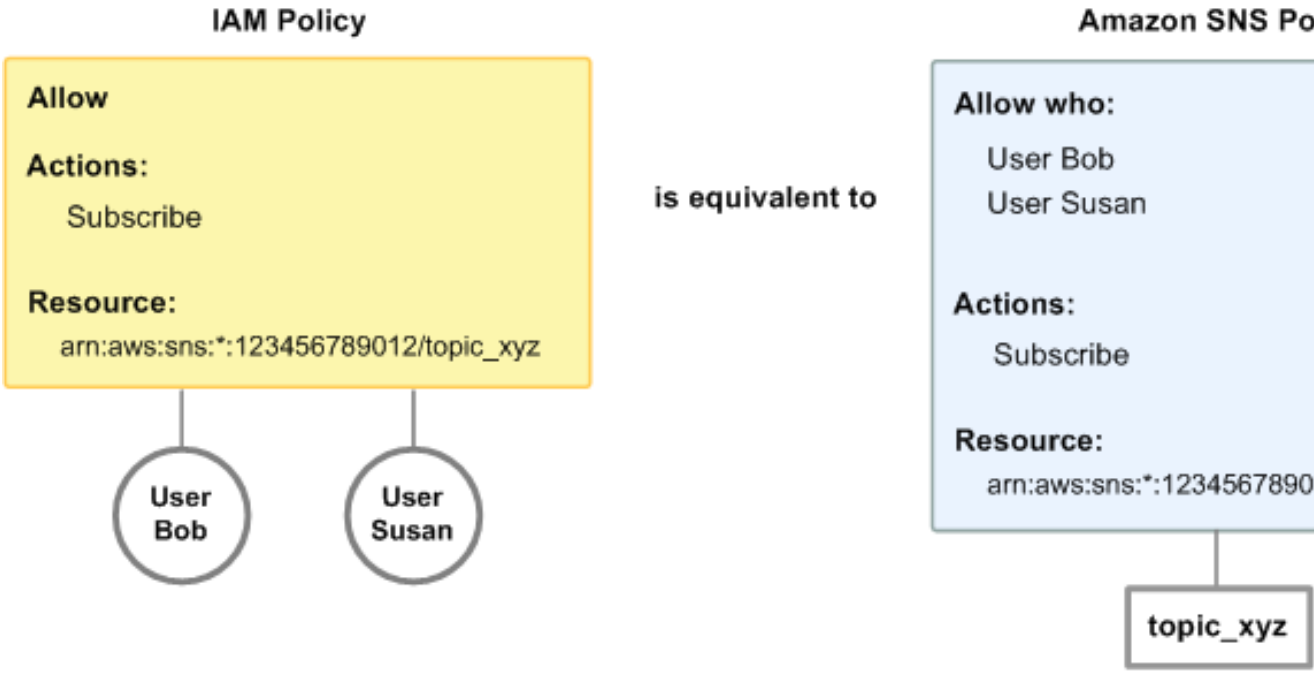
For examples of policies that cover Amazon SNS actions and resources, see Example Policies for Amazon SNS (p. 44).

IAM and Amazon SNS Policies Together

You use an IAM policy to restrict your users' access to Amazon SNS actions and topics. An IAM policy can restrict access only to users within your AWS account, not to other AWS accounts.

You use an Amazon SNS policy with a particular topic to restrict who can work with that topic (e.g., who can publish messages to it, who can subscribe to it, etc.). Amazon SNS policies can give access to other AWS accounts, or to users within your own AWS account.

To give your users permissions for your Amazon SNS topics, you can use IAM policies, Amazon SNS policies, or both. For the most part, you can achieve the same results with either. For example, the following diagram shows an IAM policy and an Amazon SNS policy that are equivalent. The IAM policy allows the Amazon SNS `Subscribe` action for the topic called topic_xyz in your AWS account. The IAM policy is attached to the users Bob and Susan (which means that Bob and Susan have the permissions stated in the policy). The Amazon SNS policy likewise gives Bob and Susan permission to access `Subscribe` for topic_xyz.



Note

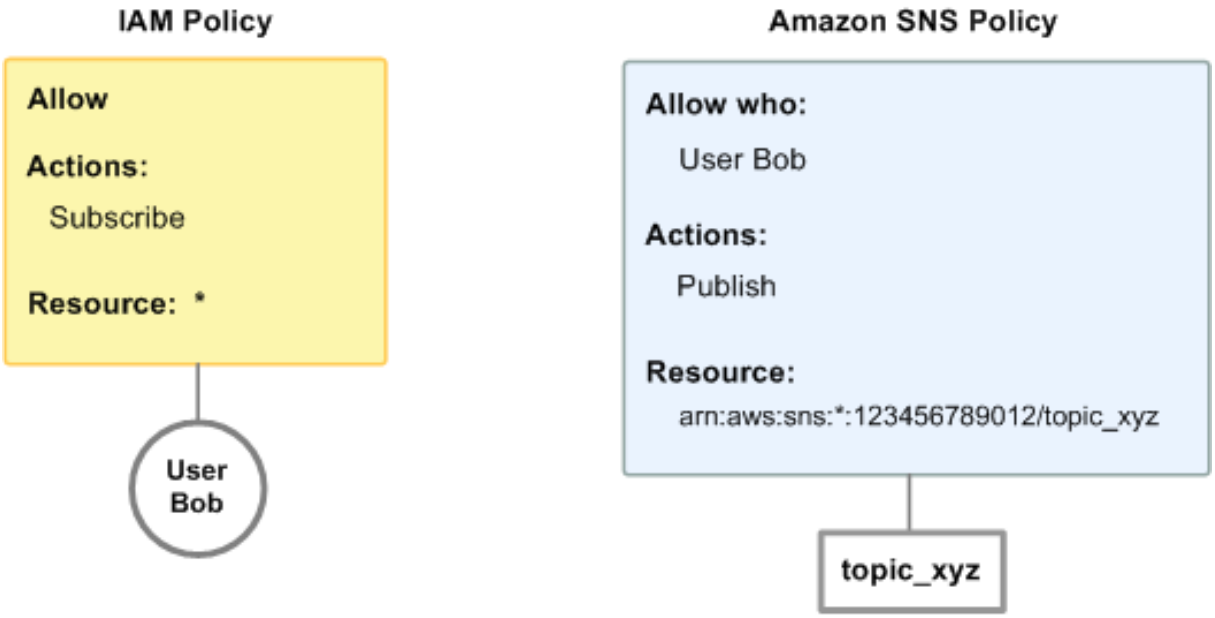
The preceding example shows simple policies with no conditions. You could specify a particular condition in either policy and get the same result.

There is one difference between AWS IAM and Amazon SNS policies: The Amazon SNS policy system lets you grant permission to other AWS accounts, whereas the IAM policy doesn't.

It's up to you how you use both of the systems together to manage your permissions, based on your needs. The following examples show how the two policy systems work together.

Example 1

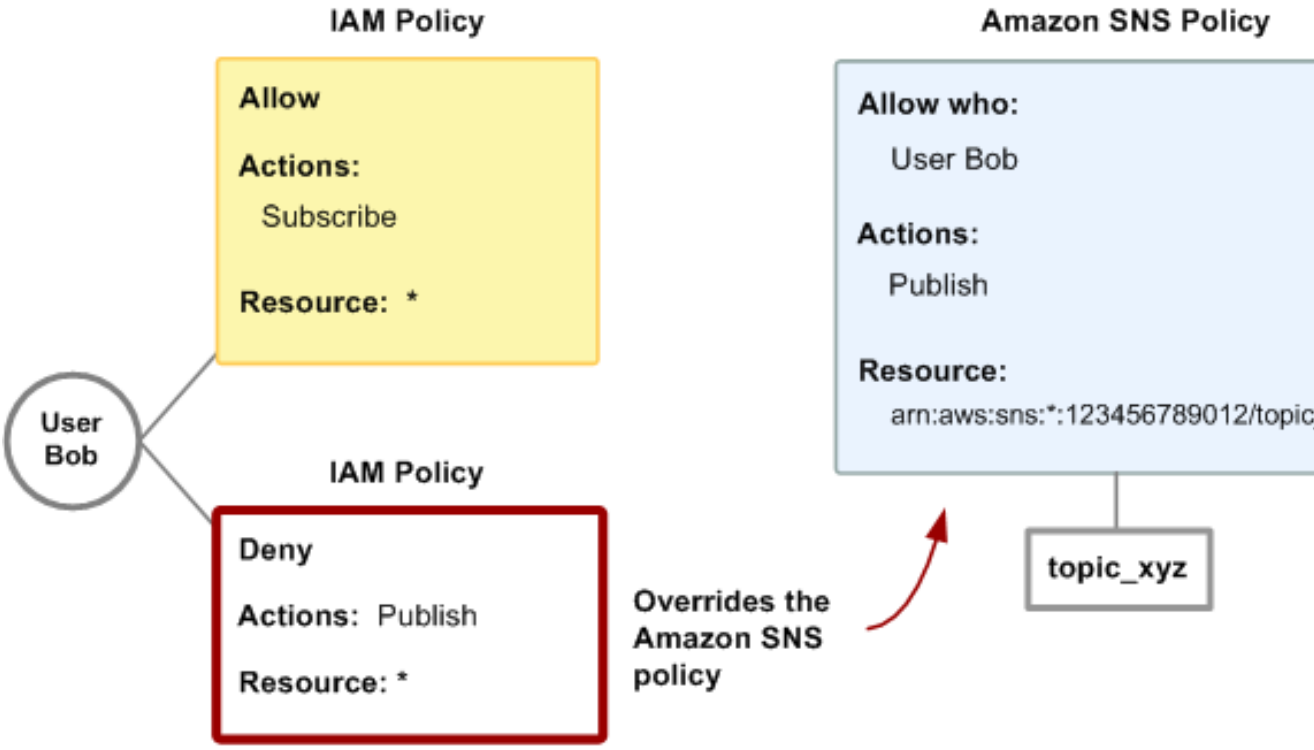
In this example, both an IAM policy and an Amazon SNS policy apply to Bob. The IAM policy gives him permission for `Subscribe` on any of the AWS account's topics, whereas the Amazon SNS policy gives him permission to use `Publish` on a specific topic (topic_xyz). The following diagram illustrates the concept.



If Bob were to send a request to subscribe to any topic in the AWS account, the IAM policy would allow the action. If Bob were to send a request to publish a message to topic_xyz, the Amazon SNS policy would allow the action.

Example 2

In this example, we build on example 1 (where Bob has two policies that apply to him). Let's say that Bob publishes messages to topic_xyz that he shouldn't have, so you want to entirely remove his ability to publish to topics. The easiest thing to do is to add an IAM policy that denies him access to the `Publish` action on all topics. This third policy overrides the Amazon SNS policy that originally gave him permission to publish to topic_xyz, because an explicit deny always overrides an allow (for more information about policy evaluation logic, see Evaluation Logic (p. 17)). The following diagram illustrates the concept.



For examples of policies that cover Amazon SNS actions and resources, see Example Policies for Amazon SNS (p. 44). For more information about writing Amazon SNS policies, go to the technical documentation for Amazon SNS.

Amazon SNS ARNs

For Amazon SNS, topics are the only resource type you can specify in a policy. Following is the Amazon Resource Name (ARN) format for topics.

```
arn:aws:sns:region:account_ID:topic_name
```

For more information about ARNs, go to ARNs in *Using IAM*.

Example

Following is an ARN for a topic named my_topic in the us-east-1 region, belonging to AWS account 123456789012.

```
arn:aws:sns:us-east-1:123456789012:my_topic
```

Example

If you had a topic named my_topic in each of the different Regions that Amazon SNS supports, you could specify the topics with the following ARN.

```
arn:aws:sns:*:123456789012:my_topic
```

You can use * and ? wildcards in the topic name. For example, the following could refer to all the topics created by Bob that he has prefixed with `bob_`.

```
arn:aws:sns:*:123456789012:bob_*
```

As a convenience to you, when you create a topic, Amazon SNS returns the topic's ARN in the response.

Amazon SNS Actions

In an IAM policy, you can specify any actions that Amazon SNS offers. However, the `ConfirmSubscription` and `Unsubscribe` actions do not require authentication, which means that even if you specify those actions in a policy, IAM won't restrict users' access to those actions.

Each action you specify in a policy must be prefixed with the lowercase string `sns:`. To specify all Amazon SNS actions, for example, you would use `sns:*`. For a list of the actions, go to the Amazon Simple Notification Service API Reference.

Amazon SNS Keys

Amazon SNS implements the following AWS-wide policy keys, plus some service-specific keys. For more information about policy keys, see Condition (p. 29).

AWS-Wide Policy Keys

- `aws:CurrentTime`–To check for date/time conditions.
- `aws:EpochTime`–To check for date/time conditions using a date in epoch or UNIX time.
- `aws:MultiFactorAuthAge`–To check how long ago (in seconds) the MFA-validated security credentials making the request were issued using Multi-Factor Authentication (MFA). Unlike other keys, if MFA is not used, this key is not present.
- `aws:principaltype`–To check the type of principal (user, account, federated user, etc.) for the current request.
- `aws:SecureTransport`–To check whether the request was sent using SSL. For services that use only SSL, such as Amazon RDS and Amazon Route 53, the `aws:SecureTransport` key has no meaning.
- `aws:SourceArn`–To check the source of the request, using the Amazon Resource Name (ARN) of the source. (This value is available for only some services. For more information, see Amazon Resource Name (ARN) under "Element Descriptions" in the *Amazon Simple Queue Service Developer Guide*.)
- `aws:SourceIp`–To check the IP address of the requester. Note that if you use `aws:SourceIp`, and the request comes from an Amazon EC2 instance, the public IP address of the instance is evaluated.

- `aws:UserAgent`–To check the client application that made the request.
- `aws:userid`–To check the user ID of the requester.
- `aws:username`–To check the user name of the requester, if available.

Note

Key names are case sensitive.

Amazon SNS Keys

Amazon SNS uses the following service-specific keys. Use these keys in policies that restrict access to `Subscribe` requests.

- sns:Endpoint—The URL, email address, or ARN from a `Subscribe` request or a previously confirmed subscription. Use with string conditions (see String Conditions (p. 32)) to restrict access to specific endpoints (e.g., *@yourcompany.com).
- sns:Protocol—The `protocol` value from a `Subscribe` request or a previously confirmed subscription. Use with string conditions (see String Conditions (p. 32)) to restrict publication to specific delivery protocols (e.g., https).

Example Policies for Amazon SNS

This section shows several simple policies for controlling user access to Amazon SNS.

Note

In the future, Amazon SNS might add new actions that should logically be included in one of the following policies, based on the policy's stated goals.

Example 1: Allow a group to create and manage topics

In this example, we create a policy that gives access to `CreateTopic`, `ListTopics`, `SetTopicAttributes`, and `DeleteTopic`.

```
{
   "Statement":[{
      "Effect":"Allow",
      "Action":["sns:CreateTopic","sns:ListTopics","sns:SetTopicAttrib
utes","sns:DeleteTopic"],
      "Resource":"*"
      }
   ]
}
```

Example 2: Allow the IT group to publish messages to a particular topic

In this example, we create a group for IT, and assign a policy that gives access to `Publish` on the specific topic of interest.

```
{
   "Statement":[{
      "Effect":"Allow",
      "Action":"sns:Publish",
      "Resource":"arn:aws:sns:*:123456789012:topic_xyz"
      }
   ]
}
```

Example 3: Give users in the AWS account ability to subscribe to topics

In this example, we create a policy that gives access to the `Subscribe`action, with string matching conditions for the `sns:Protocol` and `sns:Endpoint` policy keys.

```
{
   "Statement":[{
      "Effect":"Allow",
      "Action":["sns:Subscribe"],
      "Resource":"*",
      "Condition":{
         "StringLike": {
   "SNS:Endpoint":"*@yourcompany.com"
   },
   "StringEquals":{
            "sns:Protocol":"email"
            }
         }
      }
   ]
}
```

Example 4: Allow a partner to publish messages to a particular topic

You can use an Amazon SNS policy or an IAM policy to allow a partner to publish to a specific topic. If your partner has an AWS account, it might be easier to use an Amazon SNS policy. However, anyone in the partner's company who possesses the AWS security credentials could publish messages to the topic. This example assumes that you want to limit access to a particular person (or application). To do this you need to treat the partner like a user within your own company, and use a IAM policy instead of an Amazon SNS policy.

For this example, we create a group called WidgetCo that represents the partner company; we create a user for the specific person (or application) at the partner company who needs access; and then we put the user in the group.

We then attach a policy that gives the group `Publish` access on the specific topic named *WidgetPartnerTopic*.

We also want to prevent the WidgetCo group from doing anything else with topics, so we add a statement that denies permission to any Amazon SNS actions other than `Publish` on any topics other than WidgetPartnerTopic. This is necessary only if there's a broad policy elsewhere in the system that gives users wide access to Amazon SNS.

```
{
   "Statement":[{
         "Effect":"Allow",
         "Action":"sns:Publish",
         "Resource":"arn:aws:sns:*:123456789012:WidgetPartnerTopic"
      },
      {
         "Effect":"Deny",
         "NotAction":"sns:Publish",
         "NotResource":"arn:aws:sns:*:123456789012:WidgetPartnerTopic"
      }
   ]
}
```

Using Temporary Security Credentials

In addition to creating IAM users with their own security credentials, IAM also enables you to grant temporary security credentials to any user allowing this user to access your AWS services and resources. You can manage users who have AWS accounts; these users are IAM users. You can also manage users for your system who do not have AWS accounts; these users are called federated users. Additionally, "users" can also be applications that you create to access your AWS resources.

You can use these temporary security credentials in making requests to Amazon SNS. The API libraries compute the necessary signature value using those credentials to authenticate your request. If you send requests using expired credentials Amazon SNS denies the request.

For more information about IAM support for temporary security credentials, go to Granting Temporary Access to Your AWS Resources in *Using IAM*.

Example Using Temporary Security Credentials to Authenticate an Amazon SNS Request

The following example demonstrates how to obtain temporary security credentials to authenticate an Amazon SNS request.

```
http://sns.us-east-1.amazonaws.com/
?Name=My-Topic
&Action=CreateTopic
&Signature=gfzIF53exFVdpSNb8AiwN3Lv%2FNYXh6S%2Br3yySK70oX4%3D
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2010-03-31T12%3A00%3A00.000Z
&SecurityToken=SecurityTokenValue
&AWSAccessKeyId=Access Key ID provided by AWS Security Token Service
```

Amazon CloudWatch를 사용한 Amazon SNS 모니터링

Amazon SNS 및 Amazon CloudWatch는 통합되어 사용자가 모든 활성 Amazon SNS 주제의 메트릭을 수집하고 확인하며 분석할 수 있습니다. Amazon SNS에 Amazon CloudWatch를 구성하고 나면 사용자는 Amazon SNS 주제 및 Amazon SNS 푸시 알림에 대한 더 깊은 통찰력을 확보할 수 있습니다. 예를 들어, NumberOfNotificationsFailed 등의 Amazon SNS 메트릭에 대한 지정된 임계값에 도달하면 사용자에게 이메일 알림을 전송하도록 경보를 설정할 수 있습니다. Amazon SNS가 &CW에 보내는 모든 메트릭 목록은 Amazon SNS 메트릭 (p. 50)을 참조하십시오. Amazon SNS 푸시 알림에 대한 자세한 내용은 Amazon SNS 모바일 푸시 알림 (p. 52)을 참조하십시오.

Amazon CloudWatch로 구성하는 Amazon SNS 주제에 대한 메트릭은 5분 마다 자동으로 수집되어 Amazon CloudWatch에 푸시됩니다. 이러한 메트릭은 Amazon CloudWatch 지침을 따라 활성화된 모든 주제에서 수집됩니다. 주제는 Amazon CloudWatch에 의해 해당 주제에 대한 마지막 활동(즉, API 호출)으로부터 최대 여섯 시간까지 활성으로 간주됩니다.

Note

Amazon CloudWatch에 기록된 Amazon SNS 메트릭에 대한 과금은 없으며 Amazon SNS 서비스의 일부로 제공됩니다.

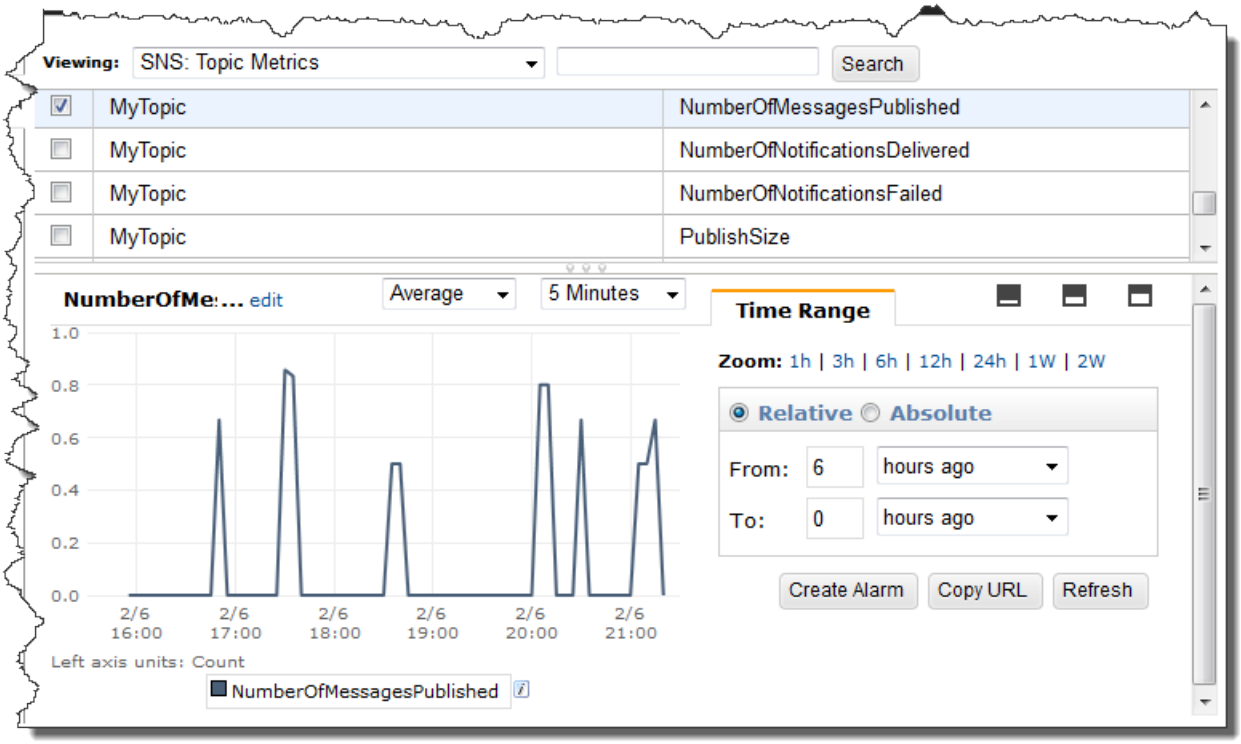
Amazon SNS용 Amazon CloudWatch 메트릭 액세스

사용자는 Amazon CloudWatch 콘솔 및 Amazon CloudWatch 자체 Command line interface(CLI)를 사용하거나 Amazon CloudWatch API를 프로그래밍 방식으로 사용하여 Amazon SNS 메트릭을 모니터링할 수 있습니다. 다음의 절차는 이러한 다른 옵션을 사용하여 메트릭에 액세스하는 방법을 설명합니다.

Amazon CloudWatch 콘솔을 사용한 메트릭 확인

1. Sign in to the AWS Management Console and open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. View Metrics를 클릭합니다.

3. Viewing 드롭다운 메뉴에서 SNS: Topic Metrics, SNS: Push Notifications by Application, SNS: Push Notifications by Application and Platform 또는 SNS: Push Notifications by Platform을 선택하여 사용 가능한 메트릭을 표시합니다.
4. 특정 항목을 클릭하여 수집된 데이터의 그래프 등 세부 정보를 확인합니다. 예를 들어, 다음의 선택된 메트릭 그래프인 NumberOfMessagesPublished는 여섯 시간의 범위에서 5분 동안 게시된 Amazon SNS 메시지의 평균 횟수를 나타냅니다.



Amazon CloudWatch CLI에서 메트릭에 액세스

- `mon-get-stats`을 호출합니다. 관련 내용 및 기타 메트릭 기능에 대한 자세한 내용은 Amazon CloudWatch Developer Guide를 참조하십시오.

Amazon CloudWatch API에서 메트릭에 액세스

- `GetMetricStatistics`을 호출합니다. 관련 내용 및 기타 메트릭 기능에 대한 자세한 내용은 Amazon CloudWatch API Reference를 참조하십시오.

Amazon SNS 메트릭용 Amazon CloudWatch 경보 설정

Amazon CloudWatch는 메트릭에 대한 임계값에 도달 시에도 경보 설정을 허용합니다. 예를 들어, 메트릭 NumberOfNotificationsFailed에 대해 경보를 설정할 수 있어 샘플링 기간 내에 지정한 임계값에 도달할 경우 이메일 알림이 전송되어 이를 알립니다.

Amazon CloudWatch 콘솔을 사용한 경보 설정

1. Sign in to the AWS Management Console and open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. Alarms를 클릭하고 나서 Create Alarm 버튼을 클릭합니다. 이로써 Create Alarm 마법사를 시작하게 됩니다.
3. Amazon SNS 메트릭을 통해 스크롤하여 경보를 설정하고자 하는 곳에 메트릭을 위치시킵니다. 메트릭을 선택하여 경보를 생성하고 Continue를 클릭합니다.
4. 메트릭의 Name, Description, Threshold, Time을 입력하고 나서 Continue를 클릭합니다.
5. 경보 상태대로 Alarm을 선택합니다. 경보 도달 시 Amazon CloudWatch로부터 이메일을 받고자 한다면 경보 기존의 Amazon SNS 주제를 선택하거나 Create New Email Topic을 클릭합니다. Create New Email Topic을 클릭하면 새로운 주제의 이름 및 이메일 주소를 설정할 수 있습니다. 이 목록은 저장되어 향후 경보의 드롭다운 상자에 표시됩니다. Continue를 클릭합니다.

   Note

   새 Amazon SNS을 생성하기 위해 Create New Email Topic을 사용할 경우 이메일 주소는 알림을 받기 전에 검증되어야 합니다. 이메일은 경보가 경보 상태에 입력될 때만 전송됩니다. 이러한 경보 상태 변경이 이메일이 검증되기 전에 발생할 경우에는 알림을 받지 못합니다.

6. 이 시점에서 Create Alarm 마법사는 사용자가 생성할 경보를 검토할 기회를 줍니다. 수정이 필요한 경우 오른쪽의 Edit 링크를 사용하면 됩니다. 수정이 완료되었다면 Create Alarm을 클릭합니다.

Amazon CloudWatch 및 경보에 대한 자세한 내용은 Amazon CloudWatch Documentation을 참조하십시오.

Amazon SNS 메트릭

Amazon SNS은 다음의 메트릭을 Amazon CloudWatch에 전송합니다.

메트릭	설명
`NumberOfMessagesPublished`	주제에 게시되는 메시지 개수입니다. Units: *Count* Valid Statistics: Sum
`PublishSize`	주제에 게시되는 메시지 크기입니다. Units: *Bytes* Valid Statistics: Minimum, Maximum, Average, Count
`NumberOfNotificationsDelivered`	주제의 모든 구독에 성공적으로 전송된 메시지 개수입니다. Units: *Count* Valid Statistics: Sum

메트릭	설명
`NumberOfNotificationsFailed`	전송 실패한 주제에 대한 모든 알림 시도 횟수입니다. Units: *Count* Valid Statistics: Sum

Amazon Simple Notification Service 메트릭 차원

Amazon Simple Notification Service은 다음의 차원을 Amazon CloudWatch에 전송합니다.

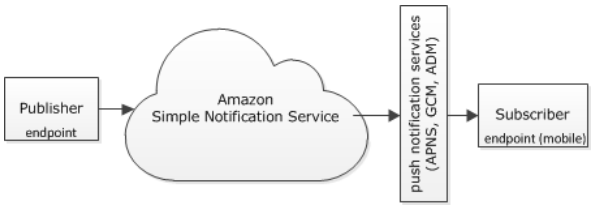
차원	설명
`Application`	APNS 및 GCM과 같이 지원되는 푸시 알림 서비스 중 하나를 등록한 앱 및 디바이스를 나타내는 애플리케이션 객체를 필터링합니다.
`Application,Platform`	APNS 및 GCM과 같이 지원되는 푸시 알림 서비스용 애플리케이션 및 플랫폼 객체를 필터링합니다.
`Platform`	APNS 및 GCM과 같은 푸시 알림 서비스용 플랫폼 객체를 필터링합니다.
`TopicName`	Amazon SNS 주제 이름을 필터링합니다.

Amazon SNS 모바일 푸시 알림

이제 Amazon SNS를 사용해 알림 메시지를 모바일 디바이스 상의 앱으로 직접 보낼 수 있습니다. 모바일 endpoint에 전송된 알림 메시지는 모바일 앱에서 메시지 알림, 배지 업데이트 또는 사운드 알림으로 나타날 수 있습니다. 사용자는 지원되는 다음의 푸시 알림 서비스 중 하나를 통해 모바일 디바이스로 알림 메시지를 전송합니다.

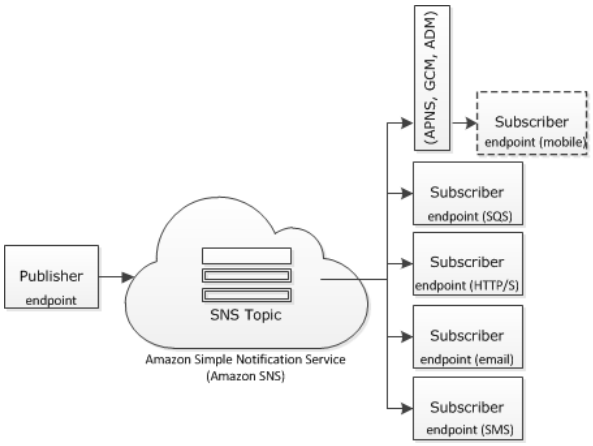
- Apple Push Notification Service (APNS)
- Google Cloud Messaging for Android (GCM)
- Amazon Device Messaging (ADM)

다음의 그림은 Amazon SNS가 알림 메시지를 모바일 endpoint에 직접 보내는 방법을 나타냅니다.



APNS 및 GCM과 같은 알림 서비스는 해당 서비스의 사용을 위해 등록된 각각의 앱 및 관련 모바일 디바이스와 연결을 유지합니다. 앱 및 모바일 디바이스 등록 시, 알림 서비스는 디바이스 토큰을 되돌립니다. Amazon SNS는 디바이스 토큰을 사용해 알림 메시지를 직접 전송할 수 있는 모바일 endpoint를 생성합니다. Amazon SNS이 다른 알림 서비스와 통신할 수 있도록 사용자는 사용할 Amazon SNS에 대한 알림 서비스 자격 증명을 제출합니다.

사용자는 알림 메시지를 직접 전송하는 것 외에도 Amazon SNS를 사용해 주제에 대해 구독한 모바일 endpoint에 메시지를 전송할 수도 있습니다. Amazon Simple Notification Service이란 무엇입니까? (p. 1)에 설명된 대로 Amazon SQS, HTTP/S, 이메일, SMS 등 주제에 대한 다른 endpoint 유형을 구독하는 것과 동일한 개념입니다. 차이점은 Amazon SNS가 알림 서비스를 통해 통신하며 구독한 모바일 endpoint는 주제에 전송된 알림을 수신한다는 점입니다. 다음의 그림은 Amazon SNS 주제에 대한 구독자로써의 모바일 endpoint를 나타냅니다. 모바일 endpoint는 다른 endpoint와는 달리 알림 서비스를 통해 통신합니다.



Amazon SNS 모바일 푸시 알림의 사용을 시작하려면 사용자는 먼저 APNS, GCM 또는 ADM과 같이 지원되는 푸시 알림 서비스 중 하나를 사용하는 모바일 endpoint 앱이 필요합니다. 이러한 서비스 중 하나를 사용하기 위해 앱을 등록 및 구성한 후에 Amazon SNS를 구성하여 모바일 endpoint로 푸시 알림을 전송할 수 있습니다. 어떤 Amazon SNS가 모바일 endpoint로 푸시 알림을 전송해야 하는지에 대한 정보는 필수 조건 (p. 53)을 참조하시기 바랍니다.

필수 조건

APNS 및 GCM과 같은 푸시 알림 서비스를 이용한 애플리케이션 등록에는 몇 가지 단계가 요구됩니다. Amazon SNS가 모바일 endpoint에 직접 알림 메시지를 전송하기 위해서는 사용자가 알림 서비스에 제공하는 몇 가지 정보가 필요합니다. 일반적으로, 사용자는 알림 서비스, 알림서비스에서 수신한 디바이스 토큰 또는 등록 ID(모바일 디바이스 및 모바일 앱) 및 알림 서비스에 등록된 모바일 앱에 연결하기 위해 자격 증명이 요구됩니다. 알림 서비스에서 요구되는 각 품목의 특정 이름은 사용되는 알림 서비스에 따라 다릅니다. 예를 들어, APNS를 알림 서비스로 사용할 때에는 *디바이스 토큰*이 필요합니다. 또는, GCM을 사용할 때에는 *registration ID*라고 불리는 동등한 디바이스 토큰이 필요합니다. *디바이스 토큰* 및 *등록 ID*는 모바일 앱 및 디바이스의 고유 식별자입니다.

다음의 섹션은 지원되는 각 알림 서비스에 대한 필수 조건을 설명합니다. 각 섹션의 처음 부분에서는 필수 조건 목록을 제공하고 추가 정보를 위해 알림 서비스 링크를 제공합니다. 각 섹션의 마지막 부분에는 Amazon SNS 모바일 푸시 API를 사용해 모바일 앱 및 디바이스로 메시지를 전송하는 방법을 설명하는 시작하기 섹션 링크가 있습니다. 필수 조건 정보를 획득했다면 AWS Management Console을 사용해 모바일 앱 및 디바이스에 메시지를 전송할 수도 있습니다. 자세한 내용은 Using Amazon SNS Mobile Push (p. 65)을 참조하십시오.

APNS 필수 조건

Amazon SNS 및 APNS를 사용해 모바일 디바이스에 푸시 알림을 전송하려면 다음의 사항이 필요합니다.

- Apple Push Notification service SSL certificate — Amazon SNS는 .pem 형식으로 된 애플리케이션의 Apple Push Notification service SSL certificate를 요구합니다. 사용자는 Mac 컴퓨터의 Keychain Access 애플리케이션을 사용하여 Apple Push Notification service SSL 인증서를 내보냅니다. SSL 인증서에 대한 자세한 내용은 Apple Local and Push Notification Programming Guide의 Provisioning and Development를 참조하십시오.
- Application private key – Amazon SNS는 .pem 형식의 애플리케이션 개인 키를 요구합니다. 사용자는 Mac 컴퓨터의 Keychain Access 애플리케이션을 사용하여 애플리케이션 개인 키를 내보냅니다.
- Device token – 애플리케이션 등록 시 APNS에서 수신한 64바이트의 16진법 값입니다. 자세한 내용은 Apple Local and Push Notification Programming Guide의 Registering for Remote Notifications를 참조하시기 바랍니다.

APNS 필수 조건의 획득 및 Amazon SNS 모바일 endpoint로의 푸시 알림 메시지 전송에 대한 자세한 내용은 Getting Started with APNS (p. 54)를 참조하십시오.

GCM 필수 조건

Amazon SNS 및 GCM을 사용해 모바일 디바이스에 푸시 알림을 전송하려면 다음의 사항이 필요합니다.

- Registration ID – 자세한 내용은 GCM Architectural Overview를 참조하십시오.
- API key (Sender Auth Token) – 자세한 내용은 GCM Architectural Overview를 참조하십시오.

GCM 필수 조건의 획득 및 Amazon SNS 모바일 endpoint로의 푸시 알림 메시지 전송에 대한 자세한 내용은 Getting Started with GCM (p. 58)를 참조하십시오.

ADM 필수 조건

Amazon SNS 및 ADM을 사용해 모바일 디바이스에 푸시 알림을 전송하려면 다음의 사항이 필요합니다.

- Registration ID – 자세한 내용은 Integrating Your App with ADM을 참조하십시오.
- Client ID – 자세한 내용은 Obtaining ADM Credentials를 참조하십시오.
- Client secret – 자세한 내용은 Obtaining ADM Credentials를 참조하십시오.

ADM 필수 조건의 획득 및 Amazon SNS 모바일 endpoint로의 푸시 알림 메시지 전송에 대한 자세한 내용은 Getting Started with ADM (p. 62)을 참조하십시오.

Getting Started with APNS

This section describes how to obtain the Apple Push Notification Service (APNS) prerequisites (app certificate and app private key) and send a push notification message to an Amazon SNS mobile endpoint. To perform the tasks in this section, you must have an Apple developer account, created an Apple Push Notification service SSL certificate, created an App ID (application identifier), registered your iOS device, and created an iOS Provisioning Profile. For more information, see the Apple Local and Push Notification Programming Guide.

Note

The type of SSL certificate must be either Apple Push Notification service SSL (Sandbox) or Apple Push Notification service SSL (Production). The App ID must not contain wildcard characters and must be enabled for push notifications.

If you already have this information, you can either use the Amazon SNS console to send a message to the mobile endpoint or you can use the Amazon SNS API. For more information about using the Amazon SNS console, see Using Amazon SNS Mobile Push (p. 65). For more information about using the Amazon SNS API, see Sending a Notification Message to a ADM Mobile Endpoint using Amazon SNS API (p. 64).

If you do not already have an iOS app registered with APNS, then you can use the sample iOS app provided by AWS as a template to get started. For more information, see the "To obtain a device token from APNS for your app" procedure later in this topic.

To download an APNS SSL certificate

1. On the Apple Developer web site, click Member Center, click Certificates, Identifiers and Profiles, and then click Certificates.

2. Select the certificate you created for iOS APNS development, click Download, and then save the file, which will have the .cer extension type.

To convert the APNS SSL certificate from .cer format to .pem format

The following steps use the openssl utility.

- At a command prompt, type the following command. Replace `myapnsappcert.cer` with the name of the certificate you downloaded from the Apple Developer web site.

```
openssl x509 -in myapnsappcert.cer -inform DER -out myapnsappcert.pem
```

  The newly created .pem file will be used to configure Amazon SNS for sending mobile push notification messages.

To obtain the application private key

The private key associated with the SSL certificate can be exported from the Keychain Access utility on your Mac computer. This is based on the assumption that you have already imported the .cer file you downloaded from the Apple Developer web site into Keychain Access. You can do this either by copying the .cer file into Keychain Access or double-clicking the .cer file.

1. Open Keychain Access, select Keys, and then highlight your application private key.
2. Click File, click Export Items..., and then enter a name in the Save As: field.
3. Accept the default .p12 file format and then click Save.

   The .p12 file will then be converted to .pem file format.

To convert the application private key from .p12 format to .pem format

- At a command prompt, type the following command. Replace `myapnsappprivatekey.p12` with the name of the private key you exported from Keychain Access.

```
openssl pkcs12 -in myapnsappprivatekey.p12 -out myapnsappprivatekey.pem
-nodes -clcerts
```

  The newly created .pem file will be used to configure Amazon SNS for sending mobile push notification messages.

To verify the certificate and private key by connecting to APNS

You can verify the .pem certificate and private key files by using them to connect to APNS.

- At a command prompt, type the following command. Replace `myapnsappcert.pem` and `myapnsappprivatekey.pem` with the name of your certificate and private key.

```
openssl s_client -connect gateway.sandbox.push.apple.com:2195 -cert myapns
appcert.pem -key myapnsappprivatekey.pem
```

To obtain a device token from APNS for your app

When you register your application with APNS to receive push notification messages, a device token is generated. The following steps describe how to use the sample iOS app provided by AWS to obtain a device token from APNS. You can use this sample iOS app to help you get started with Amazon SNS push notifications.

1. Download and unzip the snsmobilepush.zip file.
2. Navigate to the `AppleMobilePushApp` folder.
3. In Xcode, add the `AppleMobilePushApp` folder, which contains the sample iOS app.
4. Run the app in Xcode. In the output window, you should see the device token displayed, which is similar to the following:

```
Device Token = <examp1e 29z6j5c4 df46f809 505189c4 c83fjcgf 7f6257e9 8542d2jt
 3395kj73>
```

Note

Do not include spaces in the device token when submitting it to Amazon SNS.

You should now have the necessary information from APNS (device token, application private key, and APNS SSL certificate) to send Amazon SNS push notification messages to your mobile endpoint. You can now send a notification to the iOS app on your device by either using the Amazon SNS console or the Amazon SNS API. To use the Amazon SNS console, see Using Amazon SNS Mobile Push (p. 65). To use the Amazon SNS API, see Sending a Notification Message to an APNS Mobile Endpoint using the Amazon SNS API (p. 56).

Sending a Notification Message to an APNS Mobile Endpoint using the Amazon SNS API

This section describes how to use the prerequisite information to send a push notification message to your mobile endpoint using the Amazon SNS API. You add the prerequisite information to the AWS sample file `SNSMobilePush.java`, which is included in the snsmobilepush.zip file.

Note

The following steps use the Eclipse Java IDE. The steps assume you have installed the AWS SDK for Java and you have the AWS security credentials for your AWS account. For more information, see AWS SDK for Java. For more information about credentials, see How Do I Get Security Credentials? in the *AWS General Reference*.

1. Open `SNSMobilePush.java` in Eclipse.
2. Depending on which APNS you are using, uncomment either `sample.demoAppleAppNotification(Platform.APNS_SANDBOX);` or `sample.demoAppleAppNotification(Platform.APNS);`. For example, if you're using APNS_SANDBOX, it should look similar to the following:

```
SNSMobilePush sample = new SNSMobilePush(sns);
// TODO: Uncomment the services you wish to use.
//sample.demoAndroidAppNotification(Platform.GCM);
```

```
//sample.demoKindleAppNotification(Platform.ADM);
//sample.demoAppleAppNotification(Platform.APNS);
sample.demoAppleAppNotification(Platform.APNS_SANDBOX);
```

3. Enter the device token you received from APNS for the value of the device token string. For example, it should look similar to the following:

```
String deviceToken = "ex
amp1e29z6j5c4df46f809505189c4c83fjcgf7f6257e98542d2jt3395kj73";
```

4. Enter the APNS SSL certificate for your application. At the beginning of each new line in your certificate, you must add \n. For example, it should look similar to the following:

```
String certificate = "-----BEGIN CERTIFICATE-----\nMIICiTCCAfICCQD6m7oRw0uXO
jANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC\nVVMxCzAJBgNVBAgTAldBMRAwDgY
DVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6\nb24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRI
wEAYDVQQDEwlUZXN0Q2lsYWMxHzAd\nBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcN
MTEwNDI1MjA0NTIxWhcN\nMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAld
BMRAwDgYD\nVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSB
Db25z\nb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt\nYX
pvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ\n21uUSfw
fEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T\nrDHu
dUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE\nIbb3OhjZn
zcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4\nnUhVVxYUnt
neD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb\nFFBjvS
fpJIlJ00zbhNYS5f6GuoEDmFJl0ZxB
HjJnyp378OD8uTs7fLvjx79LjSTb\nNYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=\n--
---END CERTIFICATE-----";
```

5. Enter the private key for your application. At the beginning of each new line in your certificate, you must add \n. For example, it should look similar to the following:

```
String privateKey = "-----BEG1N RSA PR1VATE KEY-----\nMJICiTCHAfIC
CQD9m7oRw0uXOjANBgkqhkiG7w0BAQUFADCBiDELMAkGA1UEBhMC\nWVMxCzAJBgNVBAgTAldB
MRAwDgYDVQQHEwdTZWF0dGx2MQ8wDQYDVQQKEwZBbWF6\nVVMxCzAJBgNVBAgTAldBMRAwDgY
DVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6\n4MXNchZOFFreg4Rr3Xzhb9Rhv
lIRgsr3wU4/FYai3z96EXAMPLE=\n-----END RSA PR1VATE KEY-----";
```

6. Enter a name for your application. Application names must be made up of only uppercase and lowercase ASCII letters, numbers, underscores, hyphens, and periods, and must be between 1 and 256 characters long. For example, it should look similar to the following:

```
String applicationName = "mypushappname";
```

7. Run the application. You should see output similar to the following in the output window of your IDE:

```
==========================================
Getting Started with Amazon SNS
==========================================

{PlatformApplicationArn: arn:aws:sns:us-west-2:111122223333:app/APNS_SAND
```

```
BOX/mypushappname}
{EndpointArn: arn:aws:sns:us-west-2:111122223333:endpoint/APNS_SANDBOX/push
app/97e9ced9-f136-3893-9d60-775467eafebb}
{"default":"This is the default Message","APNS_SANDBOX":"{ \"aps\" : {
\"alert\" : \"You have got email.\", \"badge\" : 9,\"sound\" :\"default\"}}"}
Published.  MessageId=d65fb4bb-b903-5e37-83d4-feb4818d6da3
```

On your iOS device, you should see a message notification.

Getting Started with GCM

This section describes how to obtain the Google Cloud Messaging for Android (GCM) prerequisites (registration ID, server API key, and Android app registered with GCM) and send a push notification message to an Amazon SNS mobile endpoint. If you already have this information, then you can either use the Amazon SNS console to send a message to the mobile endpoint or you can use the Amazon SNS API. For more information about using the Amazon SNS console, see Using Amazon SNS Mobile Push (p. 65). For more information about using the Amazon SNS API, see Sending a Notification Message to a GCM Mobile Endpoint using the Amazon SNS API (p. 60).

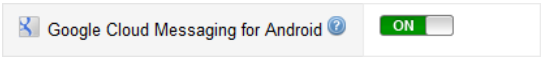
To send an Amazon SNS push notification message to an Android app, you must have a Google API project with the GCM service enabled, created a server API key, and created an Android app that has been registered with GCM, which is in addition to registering your project for the app on the Google APIs Console page.

Note

If you do not already have an Android app registered with GCM, then you can use the sample Android app provided by AWS as a template to get started. For more information, see the procedure below "To obtain a registration ID from GCM for your app".

To verify that you have a Google API project with the GCM service enabled

1. On the Google APIs Console web site, verify that you have an Google API project. If you do not have a project, then see Creating a Google API project in Getting Started with GCM.
2. Click Services, and make sure Google Cloud Messaging for Android is turned on.



To obtain the server API key

To communicate with GCM on your behalf, Amazon SNS uses your server API key.

- On the Google APIs Console web site, click API Access and make note of the server API key with the Key for server apps (with IP locking) label. If you have not yet created a server API key, then click Create new Server key.... This key will be used later in this section to send a push notification to an Amazon SNS mobile endpoint.

To obtain a registration ID from GCM for your app

When you register your app with GCM to receive push notification messages, a registration ID is generated. Amazon SNS uses this value to determine which app and associated device to send mobile push notifications to.

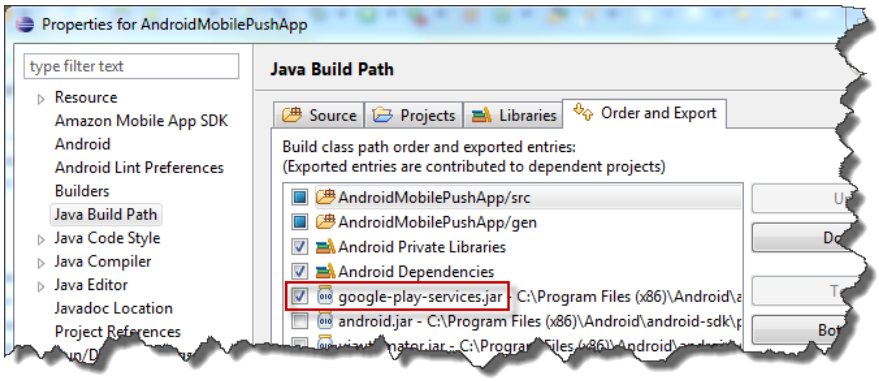
The following steps show how to use the sample Android app provided by AWS to obtain a registration ID from GCM. You can use this sample Android app to help you get started with Amazon SNS push notifications. This sample app requires the Android SDK and the Google Play Services SDK. For more information about these SDKs, see Get the Android SDK and Setup Google Play Services SDK.

1. Download and unzip the snsmobilepush.zip file.
2. Import the `AndroidMobilePushApp` folder into your IDE. In Eclipse, click File, Import, expand the Android folder, click Existing Android Code Into Workspace, click Next, browse to the folder `AndroidMobilePushApp`, click OK, and then click Finish.

   After the sample Android app has been imported into your IDE, you need to add the Project Number for your Google API project to the `strings.xml` file, which is included in the sample Android app.
3. Add the Project Number for your Google API project to the `strings.xml` file. In your IDE, you will find the file included in the values folder, which is a subfolder of res. The string will look similar to the following:

```
<string name="project_number">012345678912</string>
```

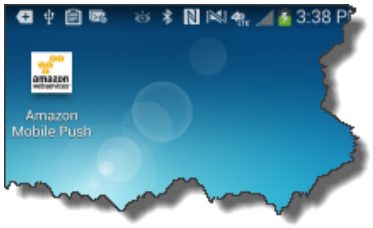
4. Add `google-play-services.jar` and `android.jar` to the Java Build Path. Select `google-play-services.jar` for export, but do not select `android.jar` for export.



5. Run the app to see the registration ID as output to the Android logging system. If you are using Eclipse with the Android ADT plug-in, you can see the registration ID in the LogCat display window. For example, the output containing the registration ID will look similar to the following:

```
06-05 11:50:43.587: V/Registration(14146): Registered, registrationId: =
Examplei7fFachkJ1xjlqT64RaBkcGHochmf1VQAr9k-IB
JtKjp7fedYPzEwT_Pq3Tu0lroqro1cwWJUvgkcPPYcaXCpPWmG3Bqn-
wiqIEzp5zZ7y_jsM0PKPxKhddCzx6paEsyay9Zn3D4wNUJb8m6HXrBf9dqaEw, error = null,
 unregistered = null
```

The installed app will appear on your Android device:



You should now have a registration ID, server API key, and Android app registered with GCM. You can now send a notification to the Android app on your device by either using the Amazon SNS console or the Amazon SNS API. To use the Amazon SNS console, see Using Amazon SNS Mobile Push (p. 65). To use the Amazon SNS API, see Sending a Notification Message to a GCM Mobile Endpoint using the Amazon SNS API (p. 60).

Sending a Notification Message to a GCM Mobile Endpoint using the Amazon SNS API

This section describes how to use the prerequisite information to send a push notification message to your mobile endpoint. You add the gathered prerequisite information to the AWS sample file `SNSMobilePush.java`, which is included in the snsmobilepush.zip file.

Note

The following steps use the Eclipse Java IDE. The steps assume you have installed the AWS SDK for Java and you have the AWS security credentials for your AWS account. For more information, see AWS SDK for Java. For more information about credentials, see How Do I Get Security Credentials? in the *AWS General Reference*.

To add the sample to Eclipse

1. Create a new Java Project in Eclipse.
2. Import the `SNSSamples` folder to the top-level directory of the newly created Java Project. In Eclipse, right-click the name of the Java Project and then click Import, expand General, click File System, click Next, browse to the `SNSSamples` folder, click OK, and then click Finish.

   Note

   In this example, you add your AWS security credentials to the `AwsCredentials.properties` file.

To add the prerequisite information to **`SNSMobilePush.java`**

1. Open `SNSMobilePush.java` in Eclipse and uncomment `sample.demoAndroidAppNotification(Platform.GCM);`. It should look similar to the following:

```
SNSMobilePush sample = new SNSMobilePush(sns);
```

```
// TODO: Uncomment the services you wish to use.
sample.demoAndroidAppNotification(Platform.GCM);
//sample.demoKindleAppNotification(Platform.ADM);
//sample.demoAppleAppNotification(Platform.APNS);
//sample.demoAppleAppNotification(Platform.APNS_SANDBOX);
```

2. Locate the `demoAndroidAppNotification` method and enter the registration ID you received from GCM for the value of the registration ID string. For example, it should look similar to the following:

```
String registrationId = = "EXAMPLE-kLMchcX0v3xOxWVhG6TfdBp...KT2TGkvnKyTvLuS
pzK_qsHgxVB_UpmcUa7Gl6g3EXAMPLE";
```

3. Enter the API key for your application. For example, it should look similar to the following:

```
String serverAPIKey = "EXAMPLExV2lcV2zEKTLNYs625zfk2jh4EXAMPLE";
```

4. Enter a name for your application. Application names must be made up of only uppercase and lowercase ASCII letters, numbers, underscores, hyphens, and periods, and must be between 1 and 256 characters long. For example, it should look similar to the following:

```
String applicationName = "gcmpushapp";
```

5. Run the application. You should see output similar to the following in the output window of your IDE:

```
==========================================
Getting Started with Amazon SNS
==========================================

{PlatformApplicationArn: arn:aws:sns:us-west-2:111122223333:app/GCM/gcmpush
app}
{EndpointArn: arn:aws:sns:us-west-2:111122223333:endpoint/GCM/gcmpush
app/5e3e9847-3183-3f18-a7e8-671c3a57d4b3}
{"default":"This is the default mes
sage","GCM":"{\"delay_while_idle\":true,\"collapse_key\":\"Wel
come\",\"data\":{\"message\":\"Visit Amazon!\",\"url\":\"ht
tp://www.amazon.com/\"},\"time_to_live\":125,\"dry_run\":false}"}
Published.  MessageId=1ca8d7d1-c261-5bfc-8689-9db269c4e46c
```

On your Android device, you should see a message notification appear within the Android app, similar to the following:

Getting Started with ADM

This section describes how to obtain the Amazon Device Messaging (ADM) prerequisites (registration ID, client ID, and client secret) and send a push notification message to an Amazon SNS mobile endpoint.

Note

An API Key is required to use ADM with pre-release or test apps. However, it is not required with a release or production version of your app when you allow Amazon to sign your app on your behalf. For more information, see Getting Your OAuth Credentials and API Key.

If you already have this information, then you can either use the Amazon SNS console to send a message to the mobile endpoint or you can use the Amazon SNS API. For more information about using the Amazon SNS console, see Using Amazon SNS Mobile Push (p. 65). For more information about using the Amazon SNS API, see Sending a Notification Message to a ADM Mobile Endpoint using Amazon SNS API (p. 64)

To send an Amazon SNS push notification message to a Kindle Fire app, you must have an Amazon developer account, set up your development environment, created a Kindle Fire app with Amazon Device Messaging enabled, and registered the app with ADM. To create an Amazon developer account, see Create an Account. For more information about setting up your development environment for developing mobile apps for Kindle Fire tablets, see Setting Up Your Development Environment.

Note

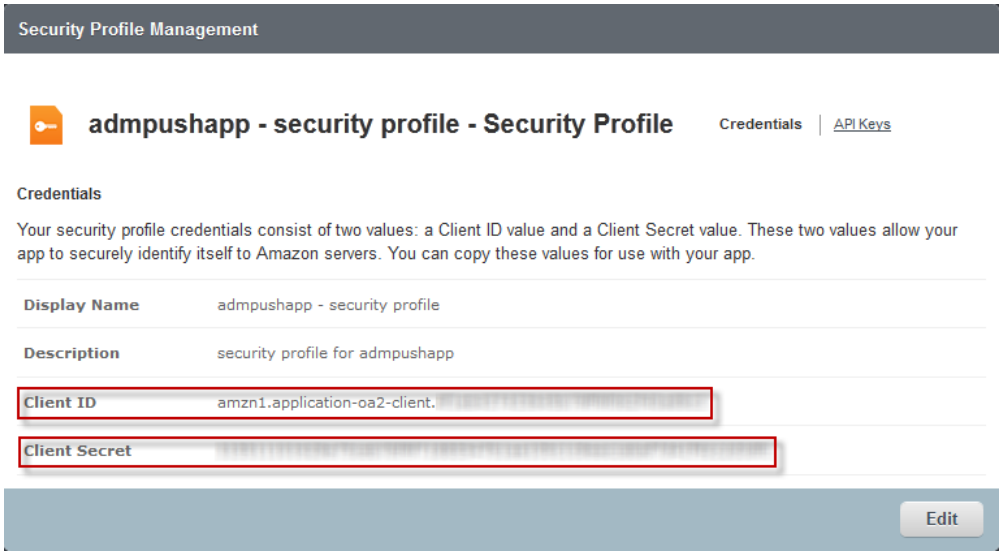
If you do not already have a Kindle app registered with ADM, then you can use the sample Kindle app provided by AWS as a template to get started. For more information, see the procedure below "To obtain a registration ID from ADM for your app".

To verify that you have a Kindle Fire app with the ADM service enabled on the Amazon Mobile App Distribution Portal

1. On the Amazon Mobile App Distribution Portal, click Apps and Services, click the name of your Kindle Fire app, and then click Device Messaging.
2. Verify that ADM is enabled for the app. If your app is not listed on the Amazon Mobile App Distribution Portal, then add it and enable ADM.

To obtain the client ID and client secret

1. On the Amazon Mobile App Distribution Portal, click Apps and Services, click the name of your Kindle Fire app, and then click Security Profile. You should see a security profile associated with your app. If not, click Security Profiles to create a new security profile.
2. Click View Security Profile. Make note of the client ID and client secret, which allows your app to securely identify itself to Amazon servers.



To obtain a registration ID from ADM for your app

The following steps show how to use the sample Kindle app provided by AWS to obtain a registration ID from ADM. You can use this sample Kindle app as an example to help you get started with Amazon SNS push notifications.

1. Download and unzip the snsmobilepush.zip file.
2. Import the `KindleMobilePushApp` folder into your IDE. In Eclipse, click File, Import, expand the Android folder, click Existing Android Code Into Workspace, click Next, browse to the folder `KindleMobilePushApp`, click OK, and then click Finish.

   After the sample Kindle app has been imported into your IDE, you need to add the API Key for your Kindle app to the `strings.xml` file, which is included in the sample Kindle app.
3. Add the API key to the `strings.xml` file. In your IDE you will find the file included in the values folder, which is a subfolder of res. You add the string to the following:

```
<string name="api_key"></string>
```

4. Run the app to see the registration ID as output to the Android logging system. If you are using Eclipse with the Android ADT plug-in, you can see the registration ID in the LogCat display window. For example, the output containing the registration ID will look similar to the following:

```
amzn1.adm-registration.v2.Example...1cwWJUvgkcPPYcaXCpPWmG3Bqn-
```

```
wiqIEzp5zZ7y_jsM0PKPxKhddCzx6paEsyay9Zn3D4wNUJb8m6HXrBf9dqaEw
```

You should now have the necessary information from ADM (registration ID, client ID, and client secret) to send Amazon SNS push notification messages to your mobile endpoint. You can now send a notification to the Kindle Fire app on your device by either using the Amazon SNS console or the Amazon SNS API. To use the Amazon SNS console, see Using Amazon SNS Mobile Push (p. 65). To use the Amazon SNS API, see Sending a Notification Message to a ADM Mobile Endpoint using Amazon SNS API (p. 64).

Sending a Notification Message to a ADM Mobile Endpoint using Amazon SNS API

This section describes how to use the prerequisite information to send a push notification message to your mobile endpoint. You add the gathered prerequisite information to the AWS sample file `SNSMobilePush.java`, which is included in the snsmobilepush.zip file.

Note

The following steps use the Eclipse Java IDE. The steps assume you have installed the AWS SDK for Java and you have the AWS security credentials for your AWS account. For more information, see AWS SDK for Java. For more information about credentials, see How Do I Get Security Credentials? in the *AWS General Reference*.

To add the sample to Eclipse

`SNSMobilePush.java` is included in the mobile push sample.

1. Create a new Java Project in Eclipse.
2. Import the `SNSSamples` folder to the top-level directory of the newly created Java Project. In Eclipse, right-click the name of the Java Project and then click Import, expand General, click File System, click Next, browse to the `SNSSamples` folder, click OK, and then click Finish.

To add the prerequisite information to **`SNSMobilePush.java`**

1. Open `SNSMobilePush.java` in Eclipse and uncomment `sample.demoKindleAppNotification(Platform.ADM);`. It should look similar to the following:

```
SNSMobilePush sample = new SNSMobilePush(sns);
// TODO: Uncomment the services you wish to use.
//sample.demoAndroidAppNotification(Platform.GCM);
sample.demoKindleAppNotification(Platform.ADM);
//sample.demoAppleAppNotification(Platform.APNS);
//sample.demoAppleAppNotification(Platform.APNS_SANDBOX);
```

2. Locate the `demoKindleAppNotification` method and enter the registration ID you received from ADM for the value of the registration ID string. For example, it should look similar to the following:

```
String registrationId = = "amzn1.adm-registration.v2.Example...1cwWJUvgk
cPPYcaXCpPWmG3Bqn-wiqIEzp5zZ7y_jsM0PKPxKhd
dCzx6paEsyay9Zn3D4wNUJb8m6HXrBf9dqaEw";
```

3. Enter the client ID for your application. For example, it should look similar to the following:

```
String clientId = "amzn1.application-oa2-client.EX
AMPLE7423654b79fc9f062fEXAMPLE";
```

4. Enter the client secret for your application. For example, it should look similar to the following:

```
String clientSecret = "EXAMPLE01658e75ceb7bf9f71939647b1aa105c1c8eac
cabaf7d41f68EXAMPLE";
```

5. Enter a name for your application. Application names must be made up of only uppercase and lowercase ASCII letters, numbers, underscores, hyphens, and periods, and must be between 1 and 256 characters long. For example, it should look similar to the following:

```
String applicationName = "admpushapp";
```

6. Run the application. You should see output similar to the following in the output window of your IDE:

```
==========================================
Getting Started with Amazon SNS
==========================================

{PlatformApplicationArn: arn:aws:sns:us-west-2:111122223333:app/ADM/mypush
appname}
{EndpointArn: arn:aws:sns:us-west-2:111122223333:endpoint/ADM/mypushapp
name/97e9ced9-f136-3893-9d60-775467eafebb}
{"default":"This is the default Message","ADM":"{ \"aps\" : { \"alert\" :
\"You have got email.\", \"badge\" : 9,\"sound\" :\"default\"}}"}
Published.  MessageId=b35fb4bz-b503-4e37-83d4-feu4218d6da6
```

On your Kindle device, you should see a message notification appear within the Kindle app.

Using Amazon SNS Mobile Push

This section describes how to use the AWS Management Console with the information described in 필수 조건 (p. 53) to register your mobile app with AWS, add device tokens (also referred to as registration IDs), send a direct message to a mobile device, and send a message to mobile devices subscribed to an Amazon SNS topic.

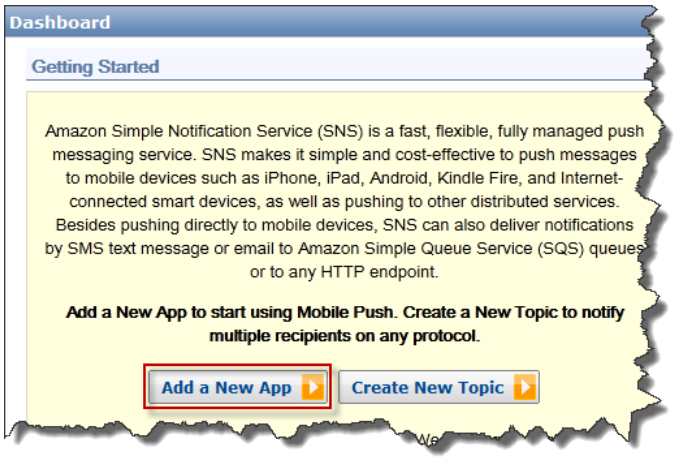
Register Your Mobile App with AWS

For Amazon SNS to send notification messages to mobile endpoints, whether it is direct or with subscriptions to a topic, you first need to register the app with AWS. To register your mobile app with AWS, enter a name to represent your app, select the platform that will be supported, and provide your credentials for the notification service platform. After the app is registered with AWS, the next step is to

create an endpoint for the app and mobile device. The endpoint is then used by Amazon SNS for sending notification messages to the app and device.

To register your mobile app with AWS

1. Go to http://aws.amazon.com/sns/ and click Add a New App.



2. In the Application Name box, enter a name to represent your app.

   Application names must be made up of only uppercase and lowercase ASCII letters, numbers, underscores, hyphens, and periods, and must be between 1 and 256 characters long.
3. In the Push Platform box, select the platform that the app is registered with and then enter the appropriate credentials.

   Note

   If you are using one of the APNS platforms, then you can select Choose File to upload the .p12 file (exported from Keychain Access) to Amazon SNS.

   For detailed instructions on how to acquire the following information, see Getting Started with ADM (p. 62), Getting Started with GCM (p. 58), or Getting Started with ADM (p. 62).

Platform	Credentials
ADM	Client ID—Go to the Amazon Mobile App Distribution Portal, click Apps and Services, click the name of your Kindle Fire app, and then click Security Profile.
	Client Secret—Go to the Amazon Mobile App Distribution Portal, click Apps and Services, click the name of your Kindle Fire app, and then click Security Profile.
APNS	Certificate—Select the password encrypted certificate and private key, as exported from Keychain Access on your Mac computer in the .p12 file format.
	Certificate Password—Enter the password.

Platform	Credentials
APNS_SANDBOX	Certificate—Same as above for APNS.
	Certificate Password—Same as above for APNS.
GCM	API Key—Go to the Google APIs Console web site, click API Access, and make note of the server API key with the Key for server apps (with IP locking) label. If you have not yet created a server API key, then click Create new Server key....

4. After you have entered this information, then click Add New App.

   This registers the app with Amazon SNS, which creates a platform application object for the selected platform and then returns a corresponding PlatformApplicationArn.

Add Device Tokens or Registration IDs

When you first register an app and mobile device with a notification service, such as Apple Push Notification Service (APNS) and Google Cloud Messaging for Android (GCM), device tokens or registration IDs are returned from the notification service. When you add the device tokens or registration IDs to Amazon SNS, they are used with the `PlatformApplicationArn` API to create an endpoint for the app and device. When Amazon SNS creates the endpoint, an *EndpointArn* is returned. The *EndpointArn* is how Amazon SNS knows which app and mobile device to send the notification message to.

You can add device tokens and registration IDs to Amazon SNS using the following methods:

- Manually add a single token to AWS using the AWS Management Console
- Migrate existing tokens from a CSV file to AWS using the AWS Management Console
- Upload several tokens using the `CreatePlatformEndpoint` API
- Register tokens from devices that will install your apps in the future

To manually add a device token or registration ID

1. Go to http://aws.amazon.com/sns/, click Apps, click your app, and then click Add Endpoints.
2. In the Endpoint Token box, enter either the token ID or registration ID, depending on which notification service. For example, with ADM and GCM you enter the registration ID.
3. (Optional) In the User Data box, enter arbitrary information to associate with the endpoint. Amazon SNS does not use this data. The data must be in UTF-8 format and less than 2KB.
4. Finally, click Add Endpoints.

   Now with the endpoint created, you can either send messages directly to a mobile device or send messages to mobile devices that are subscribed to a topic.

To migrate existing tokens from a CSV file to AWS

You can migrate existing tokens contained in a CSV file. The CSV file cannot be larger than 2MB. When migrating several tokens, it is recommended to use the `CreatePlatformEndpoint` API. Each of the tokens in the CSV file must be followed by a newline. For example, your CSV file should look similar to the following:

```
amzn1.adm-registration.v1.XpvSSUk0Rc3hTVVV--TOKEN--KMTlmMWx
wRkxMaDNST2luZz01,"User data with spaces requires quotes"
amzn1.adm-registration.v1.XpvSSUk0Rc3hTVVV--TOKEN--KMTlmMWx
wRkxMaDNST2luZz04,"Data,with,commas,requires,quotes"
amzn1.adm-registration.v1.XpvSSUk0Rc3hTVVV--TOKEN--KMTlmMWx
wRkxMaDNST2luZz02,"Quoted data requires ""escaped"" quotes"
amzn1.adm-registration.v1.XpvSSUk0Rc3hTVVV--TOKEN--KMTlmMWx
wRkxMaDNST2luZz03,"{""key"": ""json is allowed"", ""value"":""endpoint"",
""number"": 1}"
amzn1.adm-registration.v1.XpvSSUk0Rc3hTVVV--TOKEN--KMTlmMWxwRkxMaDNST2luZz05,Sim
pleDataNoQuotes
amzn1.adm-registration.v1.XpvSSUk0Rc3hTVVV--TOKEN--KMTlmMWxwRkxMaDNST2luZz06,"The
 following line has no user data"
amzn1.adm-registration.v1.XpvSSUk0Rc3hTVVV--TOKEN--KMTlmMWxwRkxMaDNST2luZz07
APBTKzPGlCyT6E6oOfpdwLpcRNxQp5vCPFiFeru9oZylc22HvZSwQTDgmmw9WdNlXMerUPxm
pX0w1,"Different token style"
```

1. Go to http://aws.amazon.com/sns/, click Apps, click your app, and then click Add Endpoints.
2. Click Migrate existing tokens over to AWS, click Choose File, select your CSV file, and then click Add Endpoints.

To upload several tokens using the **`CreatePlatformEndpoint`** API

The following steps show how to use the sample Java app provided by AWS to upload several tokens (device tokens or registration IDs) to Amazon SNS. You can use this sample app to help you get started with uploading your existing tokens.

Note

The following steps use the Eclipse Java IDE. The steps assume you have installed the AWS SDK for Java and you have the AWS security credentials for your AWS account. For more information, see AWS SDK for Java. For more information about credentials, see How Do I Get Security Credentials? in the *AWS General Reference*.

1. Download and unzip the snsmobilepush.zip file.
2. Create a new Java Project in Eclipse.
3. Import the `SNSSamples` folder to the top-level directory of the newly created Java Project. In Eclipse, right-click the name of the Java Project and then click Import, expand General, click File System, click Next, browse to the `SNSSamples` folder, click OK, and then click Finish.
4. Download a copy of the OpenCSV library and add it to the Build Path of the `bulkupload` package.
5. Open the `BulkUpload.properties` file contained in the `bulkupload` package.
6. Add the following to `BulkUpload.properties`:
   - The `ApplicationArn` to which you want to add endpoints.
   - The absolute path for the location of your CSV file containing the tokens.

- The names for CSV files (such as `goodTokens.csv` and `badTokens.csv`) to be created for logging the tokens that Amazon SNS parses correctly and those that fail.
- (Optional) The characters to specify the delimiter and quote in the CSV file containing the tokens.

Your completed `BulkUpload.properties` should look similar to the following:

```
applicationarn:arn:aws:sns:us-west-2:111122223333:app/GCM/gcmpushapp
csvfilename:C:\\mytokendirectory\\mytokens.csv
goodfilename:C:\\mylogfiles\\goodtokens.csv
badfilename:C:\\mylogfiles\\badtokens.csv
delimiterchar:'
quotechar:"
```

7. Run the BatchCreatePlatformEndpointSample.java application to upload the tokens to Amazon SNS.

   In this example, the endpoints that were created for the tokens that were uploaded successfully to Amazon SNS would be logged to `goodTokens.csv`, while the malformed tokens would be logged to `badTokens.csv`. In addition, you should see STD OUT logs written to the console of Eclipse, containing content similar to the following:

```
<1>[SUCCESS] The endpoint was created with Arn arn:aws:sns:us-west-
2:111122223333:app/GCM/gcmpushapp/165j2214-051z-3176-b586-138o3d420071
<2>[ERROR: MALFORMED CSV FILE] Null token found in /mytokendirectory/my
tokens.csv
```

To register tokens from devices that will install your apps in the future

You can use one of the following two options:

- Use a proxy server: If your application infrastructure is already set up for your mobile apps to call in and register on each installation, you can continue to use this setup. Your server will act as a proxy and pass the device token to Amazon SNS mobile push notifications, along with any user data you would like to store. For this purpose, the proxy server will connect to Amazon SNS using your AWS credentials and use the `CreatePlatofromEndpoint` API call to upload the token information. The newly created endpoint Amazon Resource Name (ARN) will be returned, which your server can store for making subsequent publish calls to Amazon SNS.
- Use the AWS token vending service: You can also enable your app installed on the mobile device to directly register with Amazon SNS mobile push notifications. Your mobile app will need credentials to create endpoints associated with your Amazon SNS platform application. We recommend using temporary credentials that expire over a period of time. These credentials can be created by implementing a token vending machine (TVM) that uses the AWS Security Token Service. For more information about TVM, see Authenticating Users of AWS Mobile Applications with a Token Vending Machine. For more information about the AWS Security Token Service, see Using Temporary Security Credentials. If you would like to be notified when an app registers with Amazon SNS, you can register to receive an Amazon SNS event that will provide the new endpoint ARN. You can also use the `ListEndpointByPlatformApplication` API to obtain the full list of endpoints registered with Amazon SNS.

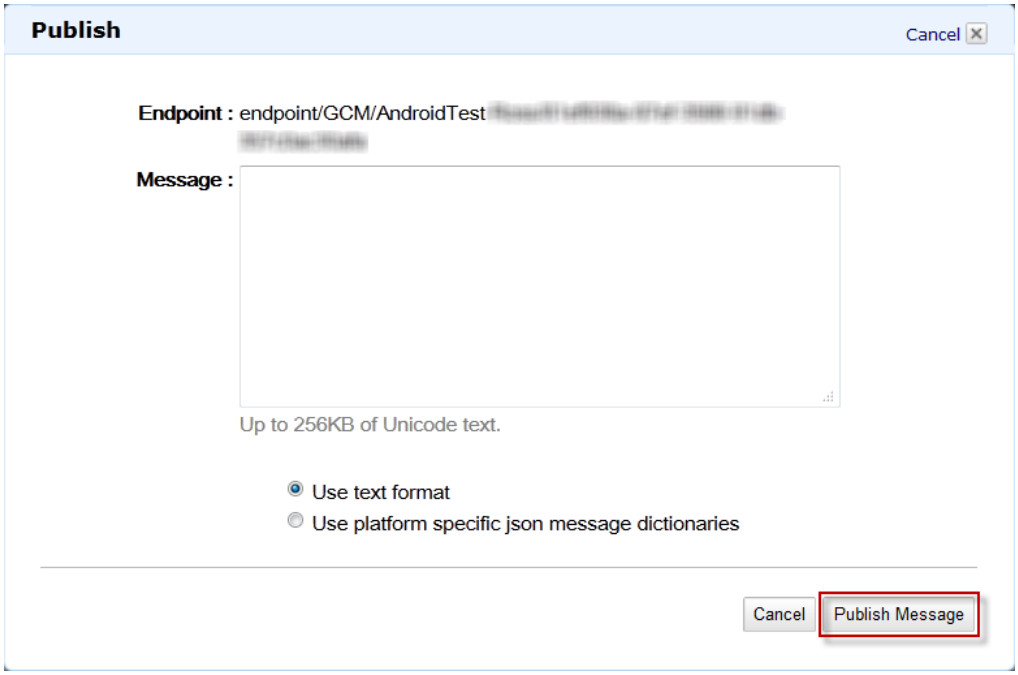
Send a Direct Message to a Mobile Device

You can send Amazon SNS push notification messages directly to an endpoint, which represents an app and mobile device, by completing the following steps.

To send a direct message

1. Go to http:/aws.amazon.com/sns/.
2. In the left Navigation pane, click Apps and click the app that you want to send a message to.
3. On the Application Details screen, select Endpoint Actions and then click Publish.
4. On the Publish dialog box, enter the message to appear in the app on the mobile device and then click Publish.

   The notification message will then be sent from Amazon SNS to the platform notification service, which will then send the message to the app.

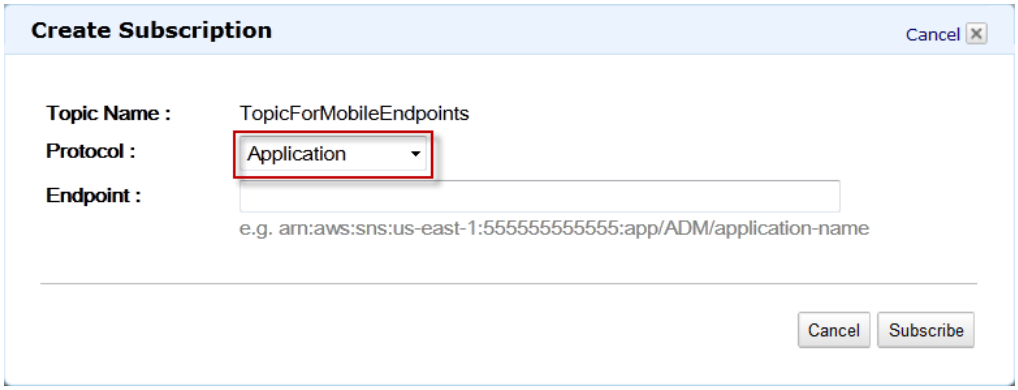


Send Messages to Mobile Devices Subscribed to a Topic

You can also use Amazon SNS to send messages to mobile endpoints subscribed to a topic. The concept is the same as subscribing other endpoint types, such as Amazon SQS, HTTP/S, email, and SMS, to a topic, as described in Amazon Simple Notification Service이란 무엇입니까? (p. 1). The difference is that Amazon SNS communicates through the notification services in order for the subscribed mobile endpoints to receive notifications sent to the topic.

To send to endpoints subscribed to a topic

1. Follow the steps as described in 주제 구독 (p. 6). You just need to select Application in the Protocol drop-down menu and then enter the mobile endpoint Amazon Resource Name (ARN) in the Endpoint box.



2. Follow the steps to publish messages to a topic, as described in 주제 게시 (p. 7), then all mobile endpoints that are subscribed to the topic will be sent the message.

Using Amazon SNS Mobile Push APIs

To use the Amazon SNS mobile push APIs, you must first meet the prerequisites for the push notification service, such as Apple Push Notification Service (APNS) and Google Cloud Messaging for Android (GCM). For more information about the prerequisites, see 필수 조건 (p. 53).

To send a push notification message to a mobile app and device using the APIs, you must first use the `CreatePlatformApplication` action, which returns a *`PlatformApplicationArn`* attribute. The *`PlatformApplicationArn`* attribute is then used by `CreatePlatformEndpoint`, which returns an *`EndpointArn`* attribute. You can then use the *`EndpointArn`* attribute with the `Publish` action to send a notification message to a mobile app and device, or you could use the *`EndpointArn`* attribute with the `Subscribe` action for subscription to a topic.

The following is a list and description of the Amazon SNS mobile push APIs:

API	Description
`CreatePlatformApplication`	Creates a platform application object for one of the supported push notification services, such as APNS and GCM, to which devices and mobile apps may register. Returns a *`PlatformApplicationArn`* attribute, which is used by the `CreatePlatformEndpoint` action. For more information, see CreatePlatformApplication in the Amazon Simple Notification Service API Reference.
`SetPlatformApplicationAttributes`	Sets the attributes of the platform application object. For more information, see SetPlatformApplicationAttributes in the Amazon Simple Notification Service API Reference.

API	Description
`GetPlatformApplicationAttributes`	Retrieves the attributes of the platform application object. For more information, see GetPlatformApplicationAttributes in the Amazon Simple Notification Service API Reference.
`ListPlatformApplications`	Lists the platform application objects for the supported push notification services. For more information, see ListPlatformApplications in the Amazon Simple Notification Service API Reference.
`DeletePlatformApplication`	Deletes a platform application object. For more information, see DeletePlatformApplication in the Amazon Simple Notification Service API Reference.
`CreatePlatformEndpoint`	Creates an endpoint for a device and mobile app on one of the supported push notification services. `CreatePlatformEndpoint` uses the *`PlatformApplicationArn`* attribute returned from the `CreatePlatformApplication` action. The *`EndpointArn`* attribute, which is returned when using `CreatePlatformEndpoint`, is then used with the `Publish` action to send a notification message to a mobile app and device. For more information, see CreatePlatformEndpoint in the Amazon Simple Notification Service API Reference.
`SetEndpointAttributes`	Sets the attributes for an endpoint for a device and mobile app. For more information, see SetEndpointAttributes in the Amazon Simple Notification Service API Reference.
`GetEndpointAttributes`	Retrieves the endpoint attributes for a device and mobile app. For more information, see GetEndpointAttributes in the Amazon Simple Notification Service API Reference.
`ListEndpointsByPlatformApplication`	Lists the endpoints and endpoint attributes for devices and mobile apps in a supported push notification service. For more information, see ListEndpointsByPlatformApplication in the Amazon Simple Notification Service API Reference.
`DeleteEndpoint`	Deletes the endpoint for a device and mobile app on one of the supported push notification services. For more information, see DeleteEndpoint in the Amazon Simple Notification Service API Reference.

API Errors for Amazon SNS Mobile Push

Errors that are returned by the Amazon SNS APIs for mobile push are listed in the following table. For more information about the Amazon SNS APIs for mobile push, see Using Amazon SNS Mobile Push APIs (p. 71).

Error	Description	HTTPS Status Code	Action that Returns this Error
Application Name is null string	The required application name is set to null.	400	CreatePlatformApplication
Platform Name is null string	The required platform name is set to null.	400	CreatePlatformApplication
Platform Name is invalid	An invalid or out-of-range value was supplied for the platform name.	400	CreatePlatformApplication
APNS — Principal is not a valid certificate	An invalid certificate was supplied for the APNS principal, which is the "SSL certificate". For more information, see CreatePlatformApplication in the Amazon Simple Notification Service API Reference.	400	CreatePlatformApplication
APNS — Principal is a valid cert but not in a .pem format	A valid certificate that is not in the .pem format was supplied for the APNS principal, which is the "SSL certificate".	400	CreatePlatformApplication
APNS — Prinicipal is an expired certificate	An expired certificate was supplied for the APNS principal, which is the "SSL certificate".	400	CreatePlatformApplication
APNS — Principal is not an Apple issued certificate	A non-Apple issued certificate was supplied for the APNS principal, which is the "SSL certificate".	400	CreatePlatformApplication
APNS — Principal is not provided	The APNS principal, which is the "SSL certificate", was not provided.	400	CreatePlatformApplication
APNS — Credential is not provided	The APNS credential, which is the "private key", was not provided. For more information, see CreatePlatformApplication in the Amazon Simple Notification Service API Reference.	400	CreatePlatformApplication

Error	Description	HTTPS Status Code	Action that Returns this Error
APNS — Credential are not in a valid .pem format	The APNS credential, which is the "private key", is not in a valid .pem format.	400	CreatePlatformApplication
GCM — serverAPIKey is not provided	The GCM credential, which is the "API key", was not provided. For more information, see CreatePlatformApplication in the Amazon Simple Notification Service API Reference.	400	CreatePlatformApplication
GCM — serverAPIKey is empty	The GCM credential, which is the "API key", is empty.	400	CreatePlatformApplication
GCM — serverAPIKey is a null string	The GCM credential, which is the "API key", is null.	400	CreatePlatformApplication
GCM — serverAPIKey is invalid	The GCM credential, which is the "API key", is invalid.	400	CreatePlatformApplication
ADM — clientsecret is not provided	The required client secret is not provided.	400	
ADM — clientsecret is a null string	The required string for the client secret is null.	400	CreatePlatformApplication
ADM — client_secret is empty string	The required string for the client secret is empty.	400	CreatePlatformApplication
ADM — client_secret is not valid	The required string for the client secret is not valid.	400	CreatePlatformApplication
ADM — client_id is empty string	The required string for the client ID is empty.	400	CreatePlatformApplication
ADM — clientId is not provided	The required string for the client ID is not provided.	400	CreatePlatformApplication
ADM — clientid is a null string	The required string for the client ID is null.	400	CreatePlatformApplication
ADM — client_id is not valid	The required string for the client ID is not valid.	400	CreatePlatformApplication
EventEndpointCreated has invalid ARN format	EventEndpointCreated has invalid ARN format.	400	CreatePlatformApplication
EventEndpointDeleted has invalid ARN format	EventEndpointDeleted has invalid ARN format.	400	CreatePlatformApplication

Error	Description	HTTPS Status Code	Action that Returns this Error
EventEndpointUpdated has invalid ARN format	EventEndpointUpdated has invalid ARN format.	400	CreatePlatformApplication
EventDeliveryAttemptFailure has invalid ARN format	EventDeliveryAttemptFailure has invalid ARN format.	400	CreatePlatformApplication
EventDeliveryFailure has invalid ARN format	EventDeliveryFailure has invalid ARN format.	400	CreatePlatformApplication
EventEndpointCreated is not an existing Topic	EventEndpointCreated is not an existing topic.	400	CreatePlatformApplication
EventEndpointDeleted is not an existing Topic	EventEndpointDeleted is not an existing topic.	400	CreatePlatformApplication
EventEndpointUpdated is not an existing Topic	EventEndpointUpdated is not an existing topic.	400	CreatePlatformApplication
EventDeliveryAttemptFailure is not an existing Topic	EventDeliveryAttemptFailure is not an existing topic.	400	CreatePlatformApplication
EventDeliveryFailure is not an existing Topic	EventDeliveryFailure is not an existing topic.	400	CreatePlatformApplication
Platform ARN is invalid	Platform ARN is invalid.	400	SetPlatformAttributes
Platform ARN is valid but does not belong to the user	Platform ARN is valid but does not belong to the user.	400	SetPlatformAttributes
APNS — Principal is not a valid certificate	An invalid certificate was supplied for the APNS principal, which is the "SSL certificate". For more information, see CreatePlatformApplication in the Amazon Simple Notification Service API Reference.	400	SetPlatformAttributes
APNS — Principal is a valid cert but not in a .pem format	A valid certificate that is not in the .pem format was supplied for the APNS principal, which is the "SSL certificate".	400	SetPlatformAttributes
APNS — Prinicipal is an expired certificate	An expired certificate was supplied for the APNS principal, which is the "SSL certificate".	400	SetPlatformAttributes
APNS — Principal is not an Apple issued certificate	A non-Apple issued certificate was supplied for the APNS principal, which is the "SSL certificate".	400	SetPlatformAttributes

Error	Description	HTTPS Status Code	Action that Returns this Error
APNS — Principal is not provided	The APNS principal, which is the "SSL certificate", was not provided.	400	`SetPlatformAttributes`
APNS — Credential is not provided	The APNS credential, which is the "private key", was not provided. For more information, see CreatePlatformApplication in the Amazon Simple Notification Service API Reference.	400	`SetPlatformAttributes`
APNS — Credential are not in a valid .pem format	The APNS credential, which is the "private key", is not in a valid .pem format.	400	`SetPlatformAttributes`
GCM — serverAPIKey is not provided	The GCM credential, which is the "API key", was not provided. For more information, see CreatePlatformApplication in the Amazon Simple Notification Service API Reference.	400	`SetPlatformAttributes`
GCM — serverAPIKey is a null string	The GCM credential, which is the "API key", is null.	400	`SetPlatformAttributes`
ADM — clientId is not provided	The required string for the client ID is not provided.	400	`SetPlatformAttributes`
ADM — clientid is a null string	The required string for the client ID is null.	400	`SetPlatformAttributes`
ADM — clientsecret is not provided	The required client secret is not provided.	400	`SetPlatformAttributes`
ADM — clientsecret is a null string	The required string for the client secret is null.	400	`SetPlatformAttributes`
EventEndpointUpdated has invalid ARN format	EventEndpointUpdated has invalid ARN format.	400	`SetPlatformAttributes`
EventEndpointDeleted has invalid ARN format	EventEndpointDeleted has invalid ARN format.	400	`SetPlatformAttributes`
EventEndpointUpdated has invalid ARN format	EventEndpointUpdated has invalid ARN format.	400	`SetPlatformAttributes`
EventDeliveryAttemptFailure has invalid ARN format	EventDeliveryAttemptFailure has invalid ARN format.	400	`SetPlatformAttributes`

Error	Description	HTTPS Status Code	Action that Returns this Error
EventDeliveryFailure has invalid ARN format	EventDeliveryFailure has invalid ARN format.	400	SetPlatformAttributes
EventEndpointCreated is not an existing Topic	EventEndpointCreated is not an existing topic.	400	SetPlatformAttributes
EventEndpointDeleted is not an existing Topic	EventEndpointDeleted is not an existing topic.	400	SetPlatformAttributes
EventEndpointUpdated is not an existing Topic	EventEndpointUpdated is not an existing topic.	400	SetPlatformAttributes
EventDeliveryAttemptFailure is not an existing Topic	EventDeliveryAttemptFailure is not an existing topic.	400	SetPlatformAttributes
EventDeliveryFailure is not an existing Topic	EventDeliveryFailure is not an existing topic.	400	SetPlatformAttributes
Platform ARN is invalid	The platform ARN is invalid.	400	GetPlatformApplicationAttributes
Platform ARN is valid but does not belong to the user	The platform ARN is valid, but does not belong to the user.	403	GetPlatformApplicationAttributes
Token specified is invalid	The specified token is invalid.	400	ListPlatformApplications
Platform ARN is invalid	The platform ARN is invalid.	400	ListEndpointsByPlatformApplication
Platform ARN is valid but does not belong to the user	The platform ARN is valid, but does not belong to the user.	404	ListEndpointsByPlatformApplication
Token specified is invalid	The specified token is invalid.	400	ListEndpointsByPlatformApplication
Platform ARN is invalid	The platform ARN is invalid.	400	DeletePlatformApplication
Platform ARN is valid but does not belong to the user	The platform ARN is valid, but does not belong to the user.	403	DeletePlatformApplication
Platform ARN is invalid	The platform ARN is invalid.	400	CreatePlatformEndpoint
Platform ARN is valid but does not belong to the user	The platform ARN is valid, but does not belong to the user.	404	CreatePlatformEndpoint
Token is not specified	The token is not specified.	400	CreatePlatformEndpoint
Token is not of correct length	The token is not the correct length.	400	CreatePlatformEndpoint

Error	Description	HTTPS Status Code	Action that Returns this Error
Customer User data is too large	The customer user data cannot be more than 2048 bytes long in UTF-8 encoding.	400	`CreatePlatformEndpoint`
Endpoint ARN is invalid	The endpoint ARN is invalid.	400	`DeleteEndpoint`
Endpoint ARN is valid but does not belong to the user	The endpoint ARN is valid, but does not belong to the user.	403	`DeleteEndpoint`
Endpoint ARN is invalid	The endpoint ARN is invalid.	400	`SetEndpointAttributes`
Endpoint ARN is valid but does not belong to the user	The endpoint ARN is valid, but does not belong to the user.	403	`SetEndpointAttributes`
Token is not specified	The token is not specified.	400	`SetEndpointAttributes`
Token is not of correct length	The token is not the correct length.	400	`SetEndpointAttributes`
Customer User data is too large	The customer user data cannot be more than 2048 bytes long in UTF-8 encoding.	400	`SetEndpointAttributes`
Endpoint ARN is invalid	The endpoint ARN is invalid.	400	`GetEndpointAttributes`
Endpoint ARN is valid but does not belong to the user	The endpoint ARN is valid, but does not belong to the user.	403	`GetEndpointAttributes`
Target ARN is invalid	The target ARN is invalid.	400	`Publish`
Target ARN is valid but does not belong to the user	The target ARN is valid, but does not belong to the user.	403	`Publish`
Message format is invalid	The message format is invalid.	400	`Publish`
Message size is larger than supported by protocol/end-service	The message size is larger than supported by the protocol/end-service.	400	`Publish`

Amazon SQS 대기열에 Amazon SNS 메시지 전송

Amazon SNS는 Amazon Simple Queue Service(Amazon SQS)와 매우 긴밀히 작동합니다. 두 서비스 모두 개발자에게 다른 혜택을 제공합니다. Amazon SNS를 사용하면 애플리케이션에서 정기적으로 업데이트를 확인하거나 "폴링"할 필요 없이 "푸시" 메커니즘을 통해 다수의 구독자에게 시간이 관건인 메시지를 보낼 수 있습니다. Amazon SQS는 폴링 모델을 통해 메시지를 교환하는데 distributed applications 에 사용되는 메시지 대기열 서비스로, 각 구성 요소가 동시적으로 사용 가능할 필요 없이도 components— 송수신의 디커플링에 사용될 수 있습니다. Amazon SNS와 Amazon SQS를 함께 사용함으로써 즉각적인 이벤트 알림을 필요로 하는 애플리케이션에 메시지를 전송할 수 있고, 다른 애플리케이션에서 나중에 처리할 수 있도록 메시지를 SQS 대기열에 계속 보관할 수도 있습니다.

사용자는 Amazon SNS 주제에 대해 Amazon SQS 대기열을 구독할 때 주제에 대해 메시지를 게시할 수 있으며 Amazon SNS는 구독된 대기열에 Amazon SQS 메시지를 전송할 수 있습니다. Amazon SQS 메시지는 JSON 문서의 메시지에 대한 메타데이터와 함께 주제에 게시된 제목 및 메시지를 포함합니다. Amazon SQS 메시지는 다음의 JSON 문서와 유사합니다.

```
{
  "Type" : "Notification",
  "MessageId" : "63a3f6b6-d533-4a47-aef9-fcf5cf758c76",
  "TopicArn" : "arn:aws:sns:us-east-1:123456789012:MyTopic",
  "Subject" : "Testing publish to subscribed queues",
  "Message" : "Hello world!",
  "Timestamp" : "2012-03-29T05:12:16.901Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEnTrFPa37tnVO0FF9Iau3MGzjlJLRfySEoWz4uZHSj6ycK4ph71Zm
dv0NtJ4dC/El9FOGp3VuvchpaTraNHWhhq/OsN1HVz20zxmF9b88R8GtqjfKB5woZZmz87HiM6CY
DTo3l7LMwFT4VU7ELtyaBBafhPTg9O5CnKkg=",
  "SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotificationSer
vice-f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&Sub
scriptionArn=arn:aws:sns:us-east-1:123456789012:MyTopic:c7fe3a54-ab0e-4ec2-88e0-
db410a0f2bee"
}
```

Note

사용자는 이제 아래에 나열된 단계 대신 Amazon SQS 콘솔을 사용해 Amazon SNS 주제에 대해 Amazon SQS 대기열을 구독하는 한편 프로세스를 간소화할 수 있습니다. 자세한 정보는 Subscribe Queue to Amazon SNS Topic을 참조하십시오.

Amazon SQS 대기열에 메시지를 전송하기 위해 Amazon SNS 주제를 활성화하려면 다음의 단계를 따라야합니다.

1. 메시지를 전송하고자 하는 대기열 및 대기열을 구독하고자 하는 주제의 Amazon Resource Name(ARN)을 파악합니다. (p. 80)
2. Amazon SNS 주제에 대한 `sqs:SendMessage` 권한을 부여하여 대기열로 메시지를 전송할 수 있습니다. (p. 81)
3. Amazon SNS 주제에 대한 대기열을 구독합니다. (p. 83)
4. Amazon SNS 주제를 게시하고 Amazon SQS 대기열에서 메시지를 읽을 수 있도록 IAM 사용자 또는 AWS 계정에 적절한 권한을 부여합니다. (p. 83)
5. 주제에 대한 메시지를 게시하고 대기열에서 메시지를 읽어 테스트합니다. (p. 85)

다른 AWS 계정의 대기열에 메시지를 보내기 위해 주제를 생성하는 방법은 다른 계정에서 Amazon SQS 대기열에 Amazon SNS 메시지 전송 (p. 86)을 참조하십시오.

두 개의 대기열에 메시지를 전송하는 주제를 생성하는 AWS CloudFormation 템플릿을 보려면 Amazon SQS 대기열에 메시지를 전송하는 주제 생성을 위한 AWS CloudFormation Template 사용 (p. 89)을 참조하십시오.

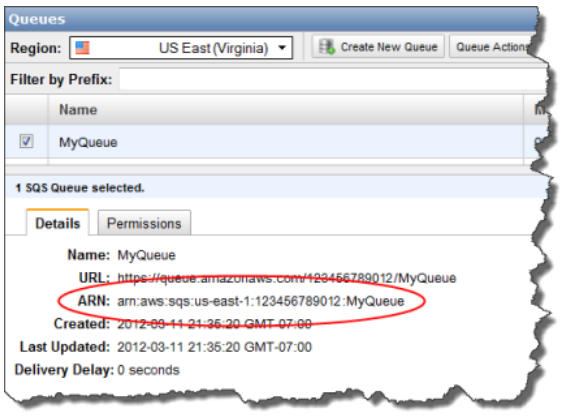
단계 1. 주제 및 대기열의 ARN을 획득합니다.

주제에 대기열을 구독할 때 사용자는 해당 대기열에 대한 ARN 사본이 필요합니다. 이와 마찬가지로 대기열에 메시지를 전송하도록 주제에 권한을 부여할 때 사용자는 주제에 대한 ARN 사본이 필요합니다.

대기열 ARN을 획득하려면 Amazon SQS 콘솔 또는 GetQueueAttributes API 작업을 사용할 수 있습니다.

Amazon SQS 콘솔로부터 대기열 ARN 획득

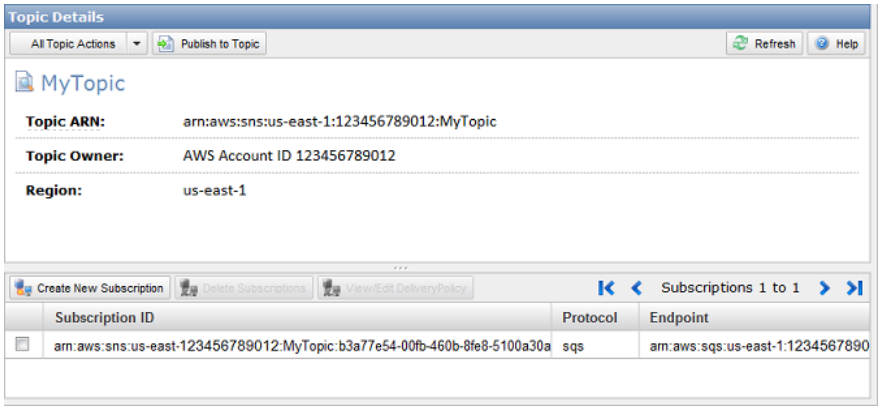
1. Sign in to the AWS Management Console and open the Amazon SQS console at https://console.aws.amazon.com/sqs/.
2. 획득하고자 하는 ARN 대기열 박스를 선택합니다.
3. Details 탭에서 ARN 값을 복사하여 Amazon SNS 주제를 구독하는데 사용할 수 있도록 합니다.



주제 ARN을 획득하기 위해 Amazon SNS 콘솔, sns-get-topic-attributes 명령 또는 GetQueueAttributes API 작업을 사용할 수 있습니다.

Amazon SNS 콘솔로부터 주제 ARN 획득

1. Sign in to the AWS Management Console and open the Amazon SNS console at https://console.aws.amazon.com/sns/.
2. 검색 창에서 획득하고자 하는 ARN 주제를 선택합니다.
3. Topic Details 창에서 Topic ARN 값을 복사하여 대기열에 메시지를 전송하는 Amazon SNS 주제에 대한 권한을 부여하는데 사용합니다.



단계 2. Amazon SQS 대기열에 메시지를 전송하도록 Amazon SNS 주제에 권한을 부여합니다.

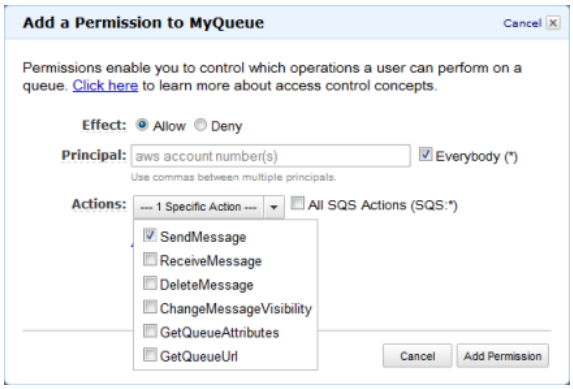
Amazon SNS 주제가 대기열에 메시지를 전송할 수 있으려면 Amazon SNS 주제가 `sqs:SendMessage` 작업을 수행하도록 허용하는 정책을 대기열에 설정해야 합니다.

주제에 대해 대기열을 구독하기 전에 주제 및 대기열이 필요합니다. 주제 또는 대기열을 아직 생성하지 않은 경우, 지금 생성합니다. 자세한 내용은 Amazon Simple Notification Service Getting Started Guide의 *Creating a Topic*을 참조하십시오. 자세한 내용은 Amazon Simple Notification Service Getting Started Guide의 *Creating a Queue*을 참조하십시오.

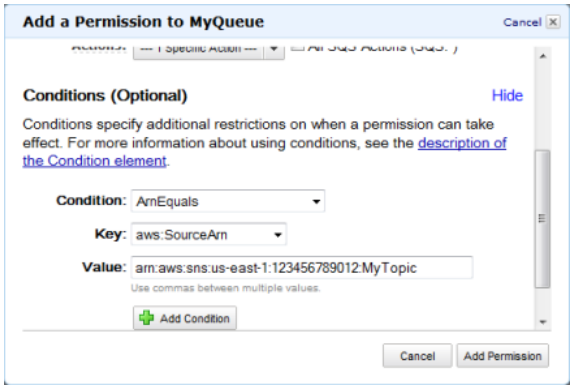
대기열에 정책을 설정하기 위해 Amazon SQS 콘솔 또는 SetQueueAttributes API 작업을 사용할 수 있습니다. 시작하기 전에 대기열로 메시지를 전송하도록 허용하고자 하는 해당 주제의 ARN을 보유하고 있는지 확인해야합니다.

Amazon SQS 콘솔을 사용해 대기열에 SendMessage 정책 설정

1. Sign in to the AWS Management Console and open the Amazon SQS console at https://console.aws.amazon.com/sqs/.
2. 설정하고자 하는 정책의 대기열 박스를 선택합니다.
3. Add a Permission 대화 상자에서 Effect의 Allow를 선택하고, Principal의 Everybody (*) 를 선택한 후 Actions 드롭다운 목록에서 SendMessage를 선택합니다.



4. 주제의 작업을 허용하는 조건을 추가합니다. Add Conditions (optional)을 클릭하고 Condition의 ArnEquals를 선택하고 Key의 aws:SourceArn을 선택하고 Value의 주제 ARN에 붙여넣기 합니다. Add Condition을 클릭합니다. 새 조건은 상자의 하단에 표시되어야 합니다(이를 보기 위해 스크롤을 내려야 할 수도 있음).



5. Add Permission을 클릭합니다.

정책 문서를 생성하고자 할 경우에는 다음과 같이 정책을 생성합니다. 정책은 MyTopic이 MyQueue에 메시지를 전송하도록 허용합니다.

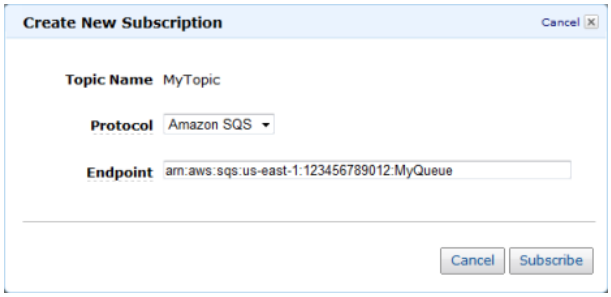
```
{
  "Statement": [
    {
      "Sid": "MySQSPolicy001",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "sqs:SendMessage",
      "Resource": "arn:aws:sqs:us-east-1:123456789012:MyQueue",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:sns:us-east-1:123456789012:MyTopic"
        }
      }
    }
  ]
}
```

단계 3. Amazon SNS 주제에 대한 대기열을 구독합니다.

주제를 통해 메시지를 전송하려면 Amazon SNS 주제에 대한 대기열을 구독해야 합니다. 사용자는 자체 ARN에 의해 해당 대기열을 지정합니다. 주제를 구독하기 위해 Amazon SNS 콘솔, sns-subscribe 명령 또는 Subscribe API 작업을 사용할 수 있습니다. 시작하기 전에 구독하고자 하는 대기열의 ARN을 보유하는지 확인해야 합니다.

Amazon SNS 콘솔을 사용한 주제에 대한 대기열 구독

1. Sign in to the AWS Management Console and open the Amazon SNS console at https://console.aws.amazon.com/sns/.
2. 검색 창에서 주제를 선택합니다.
3. Create New Subscription을 클릭하고 Protocol의 Amazon SQS를 선택하고, Endpoint에 대해 주제가 메시지를 전송할 대기열의 ARN을 붙여넣은 후 Subscribe를 클릭합니다.



4. Subscription request received! 메시지가 표시되면 Close를 클릭합니다.

   구독이 확인되면 새 구독의 Subscription ID는 구독 ID를 표시합니다. 대기열 소유자가 구독을 생성하면 구독은 자동으로 확인되며 거의 즉시 활성화됩니다.

   일반적으로 사용자는 소유한 주제에 대해 사용자 계정의 자체 대기열을 구독합니다. 하지만 소유한 주제에 대해 다른 계정의 대기열을 구독할 수도 있습니다. 구독을 생성한 사용자가 대기열의 소유자가 아닐 경우(예. 계정 A의 사용자가 계정 A의 주제에 대해 계정 B의 대기열을 구독할 경우) 구독이 확인되어야 합니다. 다른 계정에서 대기열 구독 및 구독 확인에 대한 자세한 내용은 다른 계정에서 Amazon SQS 대기열에 Amazon SNS 메시지 전송 (p. 86)를 참고하십시오.

단계 4. 사용자에게 적절한 주제 및 대기열 작업에 대한 권한을 부여합니다.

사용자는 AWS Identity and Access Management(IAM)을 사용해 적절한 사용자만 Amazon SNS 주제를 게시하고 Amazon SQS 대기열에서 메시지를 읽기/삭제하도록 허용해야 합니다. IAM 사용자의 주제 및 대기열 작업 제어에 대한 자세한 내용은 Amazon Simple Notification Service Getting Started Guide의 *Controlling User Access to Your AWS Account* 및 Amazon SQS Developer Guide의 *Controlling User Access to Your AWS Account*를 참조하십시오.

주제 또는 대기열에 대한 액세스를 제어하는 다음의 두 가지 방법이 있습니다.

- IAM 사용자 또는 그룹에 대한 정책을 추가합니다 (p. 84). 사용자에게 주제 또는 대기열에 대한 권한을 부여하는 가장 간단한 방법은 그룹을 생성하고 그룹에 적절한 정책을 추가한 후 사용자를 추가하는 것

입니다. 그룹에서 사용자를 추가하거나 제거하는 것이 각각의 사용자에 대해 설정한 정책을 추적하는 것보다 훨씬 쉽습니다.

- 주제 또는 대기열에 대한 정책을 추가합니다 (p. 84). 또 다른 AWS 계정에 주제 또는 대기열에 대한 권한을 부여하고자 할 경우 사용할 수 있는 유일한 방법은 권한을 부여하고자 하는 AWS 계정과 같은 정책을 추가하는 것입니다.

사용자는 대부분의 경우 첫 번째 방법을 사용해야 합니다(그룹에 정책을 적용하여 해당 그룹에 적절한 사용자를 추가 및 제거함으로써 사용자에 대한 권한을 관리). 또 다른 계정의 사용자에게 권한을 부여해야 할 경우에는 두 번째 방법을 사용해야 합니다.

IAM 사용자 또는 그룹에 대한 정책 추가

IAM 사용자 또는 그룹에 다음의 정책을 추가할 경우 해당 사용자 또는 그룹 구성원에게 MyTopic에 대한 `sns:Publish` 작업을 수행할 권한을 부여할 것입니다.

```
{
  "Statement":[{
      "Sid":"AllowPublishToMyTopic",
      "Effect":"Allow",
      "Action":"sns:Publish",
      "Resource":"arn:aws:sns:us-east-1:123456789012:MyTopic"
    }
  ]
}
```

IAM 사용자 또는 그룹에 다음의 정책을 추가할 경우 해당 사용자 또는 그룹 구성원에게 대기열 MyQueue1 및 MyQueue2에 대한 `sqs:ReceiveMessage` 및 `sqs:DeleteMessage` 작업을 수행할 권한을 부여할 것입니다.

```
{
  "Statement":[{
      "Sid":"AllowReadDeleteMessageOnMyQueue",
      "Effect":"Allow",
      "Action": [
        "sqs:ReceiveMessage",
        "sqs:DeleteMessage"
      ],
      "Resource": [
        "arn:aws:sns:us-east-1:123456789012:MyQueue1",
        "arn:aws:sns:us-east-1:123456789012:MyQueue2"
      ],
    }
  ]
}
```

주제 또는 대기열에 대한 정책 추가

다음의 정책 예제는 또 다른 계정에 주제 및 대기열에 대한 권한을 부여하는 방법을 보여줍니다.

Note

사용자 계정에 있는 리소스에 대한 액세스 권한을 또 다른 AWS 계정에 부여할 때 해당 리소스에 대해 관리자 레벨의 액세스(와일드카드 액세스) 권한을 보유한 IAM 사용자에게도 권한을 부여합니다. 다른 계정의 다른 모든 IAM 사용자는 자동으로 사용자 리소스에 대한 액세스가 거부됩니

다. 해당 AWS 계정의 특정 IAM 사용자에게 사용자의 리소스에 대한 액세스 권한을 부여하고자 할 경우 관리자 레벨의 액세스 권한을 보유한 계정 또는 IAM 사용자는 이러한 IAM 사용자들에게 리소스에 대한 권한을 위임해야 한다. 교차 계정 위임에 대한 자세한 정보는 Using IAM Guide의 *Enabling Cross-Account Access*를 참조하십시오.

계정 123456789012의 주제 MyTopic에 대해 다음의 정책을 추가했을 경우 사용자는 계정 111122223333에 해당 주제에 대한 `sns:Publish` 작업을 수행할 권한을 부여했을 것입니다.

```
{
  "Id":"MyTopicPolicy",
  "Statement":[{
      "Sid":"Allow-publish-to-topic",
      "Effect":"Allow",
      "Principal":{
        "AWS":"111122223333"
      },
      "Action":"sns:Publish",
      "Resource":"arn:aws:sns:us-east-1:123456789012:MyTopic"
    }
  ]
}
```

계정 123456789012의 대기열 MyQueue에 대해 다음의 정책을 추가했을 경우 사용자는 계정 111122223333에 해당 대기열에 대한 `sqs:ReceiveMessage` 및 `sqs:DeleteMessage` 작업을 수행할 권한을 부여했을 것입니다.

```
{
  "Version": "2008-10-17",
  "Id": "MyQueuePolicy",
  "Statement": [
    {
      "Sid": "Allow-Processing-Of-Messages-for-Queue",
      "Effect": "Allow",
      "Principal": {
        "AWS":"111122223333"
      },
      "Action": [
        "sqs:DeleteMessage",
        "sqs:ReceiveMessage"
      ],
      "Resource": [
        "arn:aws:sns:us-east-1:123456789012:MyQueue",
      ]
    }
  ]
}
```

단계 5. 테스트합니다.

주제를 게시하고 주제가 대기열에 전송한 메시지를 확인함으로써 주제의 대기열 구독을 테스트할 수 있습니다.

Amazon SNS 콘솔을 사용하여 주제 게시

1. AWS 계정 또는 주제를 게시할 권한을 가진 IAM 사용자의 자격 증명을 통해 AWS Management Console에 등록하고 https://console.aws.amazon.com/sns/에서 Amazon SNS 콘솔을 엽니다.
2. 검색 창에서 주제를 선택하고 Publish to Topic을 클릭합니다.
3. Subject 상자에서 제목을 입력하고(예. **`Testing publish to queue`**) Message 상자에서 문자를 입력하고(예. **`Hello world!`**) Publish Message를 클릭합니다. 다음의 메시지가 나타납니다. Your message has been successfully published.

Amazon SQS 콘솔을 사용한 주제에서의 메시지 확인

1. AWS 계정 또는 대기열의 메시지를 확인할 권한을 가진 IAM 사용자의 자격 증명을 통해 AWS Management Console에 로그인하고 https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
2. 주제를 구독하는 대기열 박스를 확인합니다.
3. Queue Action 드롭다운 목록에서 View/Delete Messages를 선택하고 Start Polling for Messages를 클릭합니다. Notification 유형으로 된 메시지가 표시됩니다.
4. Body 열에서 More Details를 클릭하십시오. Message Details 상자는 주제에 게시한 제목 및 메시지로 된 JSON 문서를 담고 있습니다. 메시지는 다음의 JSON 문서와 유사합니다.

```
{
  "Type" : "Notification",
  "MessageId" : "63a3f6b6-d533-4a47-aef9-fcf5cf758c76",
  "TopicArn" : "arn:aws:sns:us-east-1:123456789012:MyTopic",
  "Subject" : "Testing publish to subscribed queues",
  "Message" : "Hello world!",
  "Timestamp" : "2012-03-29T05:12:16.901Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEnTrFPa37tnVO0FF9Iau3MGzjlJLRfySEoWz4uZHSj6ycK4ph71Zm
dv0NtJ4dC/El9FOGp3VuvchpaTraNHWhhq/OsN1HVz20zxmF9b88R8GtqjfKB5woZZmz87HiM6CY
DTo3l7LMwFT4VU7ELtyaBBafhPTg9O5CnKkg=",
  "SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotification
Service-f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-east-1.amazonaws.com/?Action=Unsub
scribe&SubscriptionArn=arn:aws:sns:us-east-1:123456789012:MyTopic:c7fe3a54-
ab0e-4ec2-88e0-db410a0f2bee"
}
```

5. Close를 클릭합니다. 대기열에 알림 메시지를 전송하는 주제를 성공적으로 게시했습니다.

다른 계정에서 Amazon SQS 대기열에 Amazon SNS 메시지 전송

다른 계정에서 한 개 이상의 Amazon SQS 대기열 구독과 함께 Amazon SNS 주제에 대한 알림을 게시할 수 있습니다. 동일한 계정에서와 같은 방법으로 주제 및 대기열을 설정합니다(Amazon SQS 대기열에 Amazon SNS 메시지 전송 (p. 79) 참조). 유일한 차이점은 구독 확인을 처리하는 방법이며 주제에 대한 대기열을 구독하는 방법에 따라 다릅니다.

Topics

대기열 소유자 구독 생성

대기열 소유자가 구독을 생성할 경우 구독 확인은 필요하지 않습니다. 대기열은 `Subscribe` 작업이 완료되자마자 주제로부터의 알림 수신을 시작합니다. 대기열 소유자가 주제 소유자의 주제를 구독하려면 주제 소유자는 대기열 소유자 계정에 해당 주제에 대한 `Subscribe` 작업을 호출하도록 권한을 부여해야 합니다. 주제 MyTopic이 계정 123456789012에 추가되면 다음의 정책은 계정 111122223333에 계정 123456789012의 MyTopic에 대한 `sns:Subscribe`을 호출할 수 있는 권한을 부여합니다.

```
{
  "Id":"MyTopicSubscribePolicy",
  "Statement":[{
      "Sid":"Allow-other-account-to-subscribe-to-topic",
      "Effect":"Allow",
      "Principal":{
        "AWS":"111122223333"
      },
      "Action":"sns:Subscribe",
      "Resource":"arn:aws:sns:us-east-1:123456789012:MyTopic"
    }
  ]
}
```

MyTopic에 이 정책이 적용된 후 사용자는 계정 111122223333에 대해 자격 증명을 하고 Amazon SNS 콘솔에 로그인해 주제를 구독할 수 있습니다.

Amazon SQS 콘솔을 사용하여 다른 계정에서 주제에 대한 Amazon SQS 대기열 구독

1. 대기열을 포함하는 AWS 계정 또는 해당 계정의 IAM 사용자 자격 증명을 사용해 AWS Management Console에 로그인하고 https://console.aws.amazon.com/sns/에서 Amazon SNS 콘솔을 엽니다.
2. 주제 및 대기열 모두의 ARN을 보유하는지 확인해야 합니다. 이 두 가지는 구독을 생성할 때 필요합니다.
3. 대기열에 대한 `sqs:SendMessage` 권한 설정을 확인하여 주제로부터 메시지를 수신할 수 있도록 해야 합니다. 자세한 내용은 단계 2. Amazon SQS 대기열에 메시지를 전송하도록 Amazon SNS 주제에 권한을 부여합니다. (p. 81)을 참조하십시오.
4. 검색 창에서 SNS Dashboard를 선택합니다.
5. Additional Actions의 Dashboard에서 Create New Subscription을 클릭합니다.
6. Topic ARN 상자에 주제의 ARN을 입력합니다.
7. Protocol의 경우 Amazon SQS를 선택합니다.
8. Endpoint 상자에 대기열의 ARN을 입력합니다.
9. Subscribe를 클릭합니다.
10. Subscription request received! 메시지를 통해 사용자는 구독을 확인해야 한다고 알립니다. 사용자는 대기열 소유자이므로 구독 확인은 필요하지 않습니다. Close를 클릭합니다. 이제 구독 프로세스가 완료되어 주제에 대해 게시된 알림 메시지가 대기열에 전송될 수 있습니다.

사용자는 AWS 계정 111122223333에 대한 비밀 키 및 액세스 키를 사용해 `sns-subscribe`을 명령하거나 Subscribe API 작업을 호출하여 계정 123456789012의 MyTopic에 대한 Amazon SQS 대기열을 구독할 수도 있습니다. 다음의 sns-subscribe 명령을 통해 계정 123456789012의 주제 MyTopic에 대해 계정 111122223333의 대기열 MyQ를 구독합니다.

```
sns-subscribe arn:aws:sns:us-east-1:123456789012:MyTopic --protocol sqs --end
point arn:aws:sqs:us-east-1:111122223333:MyQ
```

구독을 생성하는 대기열을 소유하지 않은 사용자

구독을 생성하는 대기열을 소유하지 않은 사용자의 경우(예. 계정 A의 주제 소유자가 계정 B의 대기열을 구독 추가) 구독은 확인되어야 합니다.

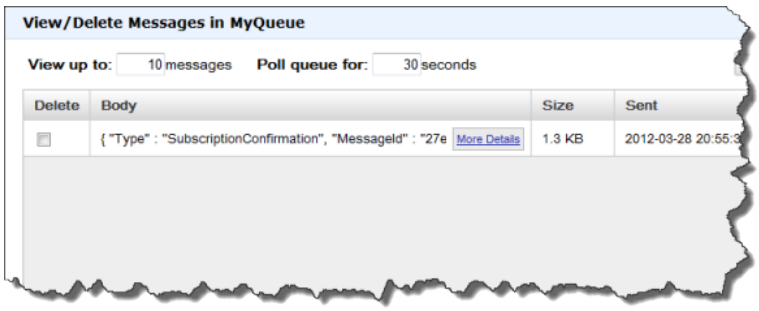
Important

주제를 구독하기 전에 대기열에 대한 `sqs:SendMessage` 권한 설정을 확인하여 주제로부터 메시지를 수신할 수 있도록 해야합니다. 단계 2. Amazon SQS 대기열에 메시지를 전송하도록 Amazon SNS 주제에 권한을 부여합니다. (p. 81)을 참조하십시오.

사용자가 `Subscribe` 작업을 호출할 경우 메시지 유형 `SubscriptionConfirmation`는 대기열로 전송되고 구독은 Pending Confirmation으로 설정된 Subscription ID로 Amazon SNS 콘솔에 표시됩니다. 구독을 확인하려면 대기열로부터의 메시지를 읽을 수 있는 사용자는 메시지의 `SubscribeURL` 값에 지정된 URL을 방문해야 합니다. 구독이 확인되기 전에는 주제에 게시된 알림은 대기열로 전송되지 않습니다. 구독을 확인하기 위해 Amazon SQS 콘솔 또는 ReceiveMessage API 작업을 사용할 수 있습니다.

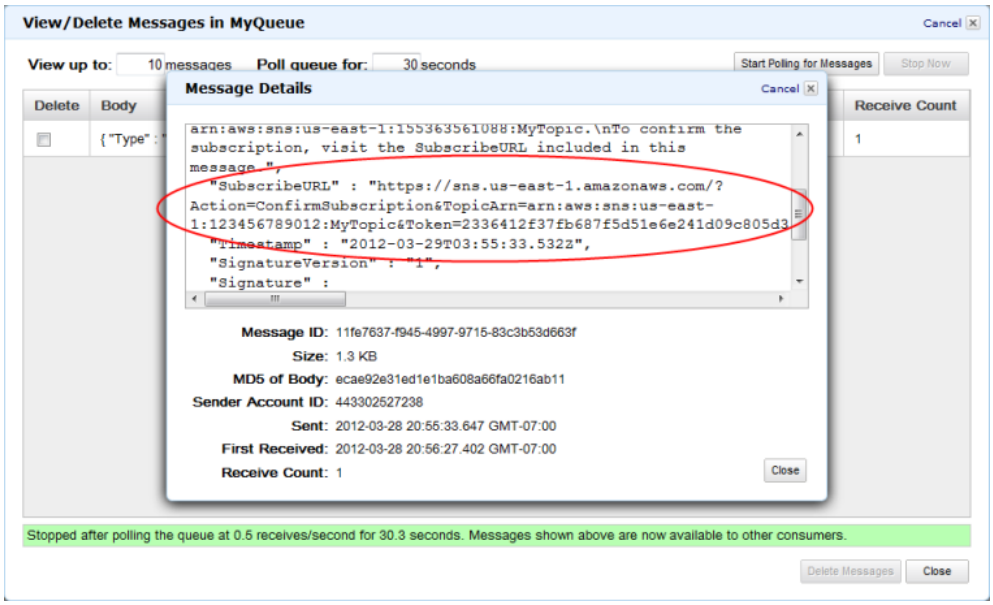
Amazon SQS 콘솔을 사용한 구독 확인

1. Sign in to the AWS Management Console and open the Amazon SQS console at https://console.aws.amazon.com/sqs/.
2. 주제에 대한 대기 중인 구독을 보유한 대기열을 선택합니다.
3. Queue Action 드롭다운 목록에서 View/Delete Messages를 선택하고 Start Polling for Messages를 클릭합니다. SubscriptionConfirmation 유형으로 된 메시지가 표시됩니다.
4. Body 컬럼에서 More Details를 클릭하십시오.



5. 텍스트 상자에서 SubscribeURL 값을 찾아 URL을 복사합니다. 다음의 URL과 유사합니다.

```
https://sns.us-east-1.amazonaws.com/?Action=ConfirmSubscription&Topi
cArn=arn:aws:sns:us-east-1:123456789012:MyTop
```



6. 웹 브라우저에서 주소입력창에 URL을 붙여넣어 해당 URL을 방문합니다. 다음의 XML 문서와 유사한 반응을 보게 됩니다.

```
<ConfirmSubscriptionResponse xmlns="http://sns.amazonaws.com/doc/2010-03-31/">
  <ConfirmSubscriptionResult>
    <SubscriptionArn>arn:aws:sns:us-east-1:123456789012:MyTopic:c7fe3a54-ab0e-4ec2-88e0-db410a0f2bee</SubscriptionArn>
  </ConfirmSubscriptionResult>
  <ResponseMetadata>
    <RequestId>dd266ecc-7955-11e1-b925-5140d02da9af</RequestId>
  </ResponseMetadata>
</ConfirmSubscriptionResponse>
```

이제 Amazon SNS 콘솔의 주제 구독을 확인하면 구독 ARN이 Subscription ID 열의 Pending Confirmation 메시지를 대체함을 보게 됩니다. 구독된 대기열은 주제로부터 메시지를 수신할 준비가 되었습니다.

Amazon SQS 대기열에 메시지를 전송하는 주제 생성을 위한 AWS CloudFormation Template 사용

AWS CloudFormation을 통해 템플릿 파일을 사용하여 AWS 리소스 모음을 단일 유닛으로 생성 및 구성할 수 있습니다. 이 섹션은 대기열을 게시하는 주제를 쉽게 배포하는 예제 템플릿을 보여줍니다. 템플릿은 사용자가 두 개의 대기열을 생성하고, 대기열에 대한 주제를 생성하며, 해당 대기열에 정책을 추가하여 주제가 대기열에 메시지를 전송할 수 있도록 하고, 이러한 리소스에 대한 액세스를 제어하는 IAM 사용자 및 그룹을 생성하는 등의 설정 단계에서 사용됩니다.

AWS CloudFormation 템플릿을 이용한 AWS 리소스 배포에 대한 자세한 내용은 AWS CloudFormation User Guide의 *Get Started*를 참조하십시오.

AWS 계정에서 주제 및 대기열 설정을 위한 AWS CloudFormation 템플릿 사용

예제 템플릿은 하나의 IAM 그룹이 주제를 게시할 권한 및 대기열에서 메시지를 읽을 수 있는 권한을 적절히 부여하여 두 개의 Amazon SQS 대기열에 메시지를 전송할 수 있는 Amazon SNS 주제를 생성합니다. 템플릿은 각 그룹에 추가될 IAM 사용자도 생성합니다.

이 템플릿은 AWS CloudFormation Sample Templates page에서 다운로드 (https://s3.amazonaws.com/cloudformation-templates-us-east-1/SNSToSQS.template) 받을 수 있습니다.

MySNSTopic은 두 개의 Amazon SQS 대기열인 두 개의 구독된 endpoints를 게시하기 위해 설치합니다 (MyQueue1 및 MyQueue2). MyPublishTopicGroup은 IAM 그룹으로 구성원은 Publish API 작업 또는 sns-publish 명령을 사용해 MySNSTopic을 게시할 권한을 갖고 있습니다. 템플릿은 IAM 사용자 MyPublishUser 및 MyQueueUser를 생성하고 이들에게 로그인 프로필 및 액세스 키를 부여합니다. 이 템플릿으로 스택을 생성한 사용자는 로그인 프로필에 입력 매개 변수로 암호를 지정합니다. 템플릿은 MyPublishUserKey 및 MyQueueUserKey를 가진 두 IAM 사용자에 대한 액세스 키를 생성합니다. AddUserToMyPublishTopicGroup이 MyPublishTopicGroup에 MyPublishUser를 추가하여 사용자는 그룹에 할당된 권한을 갖게 됩니다.

MyRDMessageQueueGroup은 구성원이 ReceiveMessage 및 DeleteMessage API 작업을 사용하는 두 Amazon SQS 대기열에서 메시지를 읽고 삭제할 수 있는 권한을 가진 IAM 그룹입니다. AddUserToMyQueueGroup이 MyRDMessageQueueGroup에 MyQueueUser를 추가하여 사용자는 그룹에 할당된 권한을 갖게 됩니다. MyQueuePolicy는 두 대기열에 알림을 게시하도록 MySNSTopic에 대한 권한을 할당합니다.

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "This Template creates an Amazon SNS topic that can send
messages to two Amazon SQS queues with appropriate permissions for one IAM user
 to publish to the topic and another to read messages from the queues. MySNSTopic
 is set up to publish to two subscribed endpoints, which are two Amazon SQS
queues (MyQueue1 and MyQueue2). MyPublishUser is an IAM user that can publish
to MySNSTopic using the Publish API. MyTopicPolicy assigns that permission to
MyPublishUser. MyQueueUser is an IAM user that can read messages from the two
Amazon SQS queues. MyQueuePolicy assigns those permissions to MyQueueUser. It
also assigns permission for MySNSTopic to publish its notifications to the two
 queues. The template creates access keys for the two IAM users with MyPub
lishUserKey and MyQueueUserKey.  Note that you will be billed for the AWS re
sources used if you create a stack from this template.",

  "Parameters" : {
    "MyPublishUserPassword": {
      "NoEcho": "true",
      "Type": "String",
      "Description" : "Password for the IAM user MyPublishUser",
      "MinLength": "1",
      "MaxLength": "41",
      "AllowedPattern" : "[a-zA-Z0-9]*",
      "ConstraintDescription" : "must contain only alphanumeric characters."
    },
    "MyQueueUserPassword": {
      "NoEcho": "true",
      "Type": "String",
      "Description" : "Password for the IAM user MyQueueUser",
```

```
    "MinLength": "1",
    "MaxLength": "41",
    "AllowedPattern" : "[a-zA-Z0-9]*",
    "ConstraintDescription" : "must contain only alphanumeric characters."
  }
},


"Resources" : {
  "MySNSTopic" : {
    "Type" : "AWS::SNS::Topic",
    "Properties" : {
      "Subscription" : [
        {
          "Endpoint" : { "Fn::GetAtt" : ["MyQueue1", "Arn"]},
          "Protocol" : "sqs"
        },
        {
          "Endpoint" : { "Fn::GetAtt" : ["MyQueue2", "Arn"]},
          "Protocol" : "sqs"
        }
      ]
    }
  },
  "MyQueue1" : {
    "Type" : "AWS::SQS::Queue"
  },
  "MyQueue2" : {
    "Type" : "AWS::SQS::Queue"
  },
  "MyPublishUser" : {
   "Type" : "AWS::IAM::User",
   "Properties" : {
     "LoginProfile": {
       "Password": {"Ref" : "MyPublishUserPassword"}
     }
   }
  },
  "MyPublishUserKey" : {
   "Type" : "AWS::IAM::AccessKey",
   "Properties" : {
        "UserName" : {"Ref": "MyPublishUser"}
    }
  },
  "MyPublishTopicGroup" : {
    "Type" : "AWS::IAM::Group",
    "Properties" : {
      "Policies": [
        {
          "PolicyName": "MyTopicGroupPolicy",
          "PolicyDocument": {"Statement":[
            {
              "Effect":"Allow",
        "Action":[
                "sns:Publish"
              ],
              "Resource": {"Ref" : "MySNSTopic"}
            }
```

```
        ]}
      }
    ]
  }
},
"AddUserToMyPublishTopicGroup" : {
  "Type" : "AWS::IAM::UserToGroupAddition",
  "Properties" : {
    "GroupName": {"Ref" : "MyPublishTopicGroup"},
    "Users" : [{ "Ref" : "MyPublishUser" }]
  }
},
"MyQueueUser" : {
 "Type" : "AWS::IAM::User",
 "Properties" : {
   "LoginProfile": {
     "Password": {"Ref" : "MyQueueUserPassword"}
   }
 }
},
"MyQueueUserKey" : {
 "Type" : "AWS::IAM::AccessKey",
 "Properties" : {
      "UserName" : {"Ref": "MyQueueUser"}
  }
},
"MyRDMessageQueueGroup" : {
  "Type" : "AWS::IAM::Group",
  "Properties" : {
    "Policies": [
      {
        "PolicyName": "MyQueueGroupPolicy",
        "PolicyDocument": {"Statement":[
          {
            "Effect":"Allow",
      "Action":[
              "sqs:DeleteMessage",
              "sqs:ReceiveMessage"
            ],
            "Resource":[
              { "Fn::GetAtt" : ["MyQueue1", "Arn"]},
              { "Fn::GetAtt" : ["MyQueue2", "Arn"]}
            ]
          }
        ]}
      }
    ]
  }
},
"AddUserToMyQueueGroup" : {
  "Type" : "AWS::IAM::UserToGroupAddition",
  "Properties" : {
    "GroupName": {"Ref" : "MyRDMessageQueueGroup"},
    "Users" : [{ "Ref" : "MyQueueUser" }]
  }
},
"MyQueuePolicy" : {
  "Type" : "AWS::SQS::QueuePolicy",
```

```
    "Properties" : {
      "PolicyDocument":  {
        "Id":"MyQueuePolicy",
        "Statement" : [
          {
            "Sid":"Allow-SendMessage-To-Both-Queues-From-SNS-Topic",
     "Effect":"Allow",
     "Principal" : {"AWS" : "*"},
      "Action":["sqs:SendMessage"],
     "Resource": "*",
            "Condition": {
              "ArnEquals": {
                "aws:SourceArn": { "Ref" : "MySNSTopic" }
              }
            }
          }
        ]
      },
      "Queues" : [{"Ref" : "MyQueue1"}, {"Ref" : "MyQueue2"}]
    }
  }
},
"Outputs" : {
  "MySNSTopicTopicARN" : {
    "Value" : { "Ref" : "MySNSTopic" }
  },
  "MyQueue1Info" : {
    "Value" : {"Fn::Join" : [
      " ",
      [
        "ARN:",
        { "Fn::GetAtt" : [ "MyQueue1", "Arn" ] },
        "URL:",
        { "Ref" : "MyQueue1" }
      ]
    ]}
  },
  "MyQueue2Info" : {
    "Value" : {"Fn::Join" : [
      " ",
      [
        "ARN:",
        { "Fn::GetAtt" : [ "MyQueue2", "Arn" ] },
        "URL:",
        { "Ref" : "MyQueue2" }
      ]
    ]}
  },
  "MyPublishUserInfo" : {
    "Value" : {"Fn::Join" : [
      " ",
      [
        "ARN:",
        { "Fn::GetAtt" : [ "MyPublishUser", "Arn" ] },
        "Access Key:",
        {"Ref" : "MyPublishUserKey"},
        "Secret Key:",
        {"Fn::GetAtt" : ["MyPublishUserKey", "SecretAccessKey"]}
```

```
        ]
      ]}
    },
    "MyQueueUserInfo" : {
      "Value" : {"Fn::Join" : [
        " ",
        [
          "ARN:",
          { "Fn::GetAtt" : [ "MyQueueUser", "Arn" ] },
          "Access Key:",
          {"Ref" : "MyQueueUserKey"},
          "Secret Key:",
          {"Fn::GetAtt" : ["MyQueueUserKey", "SecretAccessKey"]}
        ]
      ]}
    }
  }
}
```

Amazon SNS를 사용한 SMS 알림 송수신

Amazon Simple Notification Service(Amazon SNS)을 사용하여 SMS 사용 가능한 휴대전화 및 스마트폰에 Short Message Service(SMS) 알림을 송수신할 수 있습니다.

Amazon SNS을 사용하여 SMS 메시지를 전송하려면 표시 이름이 있는 Amazon SNS 주제 중 하나를 선택하여 주제에 대해 메시지를 게시합니다. 표시 이름의 첫 10 문자가 문자 메시지 접두사의 앞부분으로 사용되므로 주제는 지정된 표시 이름을 보유해야 합니다. SMS 메시지는 최대 ASCII 140자 또는 Unicode 70자 입니다. Amazon SNS는 발신하는 모든 SMS 메시지에 표시 이름 접두사를 포함하므로 표시 이름 접두사와 메시지 페이로드의 합은 ASCII 140자 또는 Unicode 70자를 초과할 수 없습니다. Amazon SNS는 이 제한을 초과하는 메시지의 끝을 자릅니다.

Amazon SNS를 사용하여 SMS 메시지를 수신하려면 Amazon SNS 주제 구독 시 SMS 프로토콜 설정을 선택합니다. 전체 메시지 접두사는 표시 이름과 > 문자를 구성됩니다. 예를 들어, 주제의 표시 이름이 `MyTopic`이고 전송된 메시지 페이로드가 `Hello World!`이라면 전달된 메시지는 다음의 예와 같이 표시됩니다.

```
MYTOPIC>Hello World!
```

Note

표시 이름은 대/소문자로 구분하지 않으며 Amazon SNS는 SMS 메시지의 표시 이름을 대문자로 전환합니다.

이메일 등의 다른 알림 유형과 함께 SMS 알림을 사용할 수 있습니다. 예를 들어, Amazon CloudWatch를 사용하여 AWS 애플리케이션을 모니터링하는 경우 Amazon SNS 주제와 연관된 Amazon CloudWatch 경보를 생성할 수 있습니다. 그러고 나서 이메일 및 SMS를 통해 주제를 구독하고 이메일뿐 아니라 SMS 가능한 디바이스로도 알림을 수신할 수 있습니다.

SMS 및 이메일 알림 단일 메시지의 사용을 촉진하기 위해 Amazon SNS가 메시지에 메시지 본문과 제목이 포함되어 있는지 여부를 확인합니다. 메시지 본문만으로 구성된 메시지를 게시할 경우 SMS 및 이메일 구독자는 각 프로토콜의 최대 제한 크기로(SMS의 경우 140자, 이메일의 경우 8KiB) 동일한 메시지를 수신합니다. 메시지가 140자 보다 긴 경우 SMS 메시지의 끝은 잘립니다.

메시지 페이로드가 140자 이상일 때 SMS 메시지의 끝 잘림 현상을 방지하려면 제목 및 메시지 페이로드로 구성된 메시지를 게시합니다. 제목 및 메시지 페이로드로 구성된 메시지의 경우 Amazon SNS는 SMS

구독자들에게 제목만 전송하지만 이메일 구독자에게는 제목 및 메시지를 모두 전송합니다. 이로써 이메일 알림을 최대 8KiB까지 길게 전송할 수 있으며 모바일 디바이스에 SMS 메시지로 제목줄을 전송할 수도 있습니다.

Note

SMS 알림은 현재 미국 내 전화번호를 지원합니다. SMS 메시지는 US East (Northern Virginia) Region에서 생성된 주제에서만 전송될 수 있습니다. 하지만 다른 어느 지역에서나 US East (Northern Virginia) Region에서 생성한 주제에 대한 메시지를 게시할 수 있습니다.

Amazon SNS는 SMS 메시지를 송수신하기 위해 단축 번호 `30304`를 사용합니다.

필수 조건

- Sign up for Amazon SNS – 계정이 없는 경우 AWS 계정을 생성합니다.
  자세한 내용은 시작하기 전 (p. 4)을 참조하십시오.
- Create an Amazon SNS topic – 주제가 없는 경우 Amazon SNS 주제를 생성합니다.
  자세한 내용은 주제 생성 (p. 4)을 참조하십시오.

필수 조건 작업을 모두 완료한 후에 사용자는 다음의 절차를 통해 Amazon SNS로 SMS 메시지를 게시 및 수신할 수 있습니다.

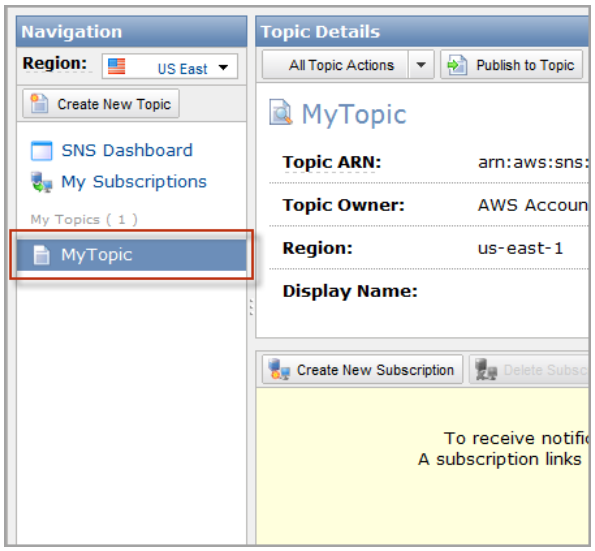
Amazon SNS를 사용한 SMS 메시지 송수신 절차

작업 1: 주제 표시 이름 지정

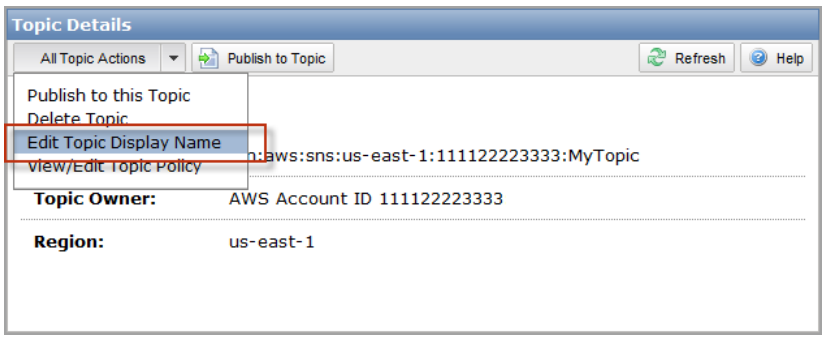
주제에 대해 SMS 메시지를 게시하려면 주제에 표시 이름을 지정해야 합니다.

주제에 표시 이름 지정

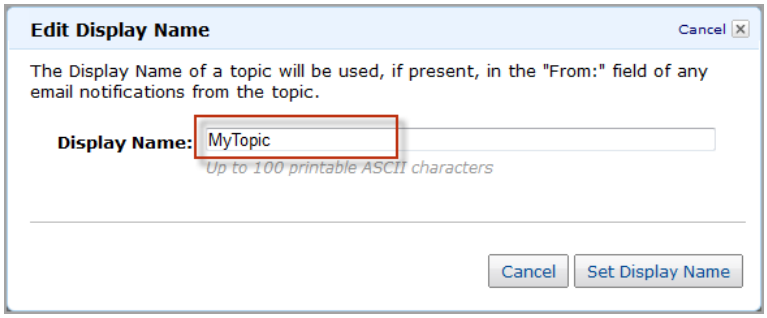
1. Sign in to the AWS Management Console and open the Amazon SNS console at https://console.aws.amazon.com/sns/.
2. 검색 창에서 주제를 선택합니다.
   다음의 예에서는 주제 이름을 *MyTopic*으로 합니다.



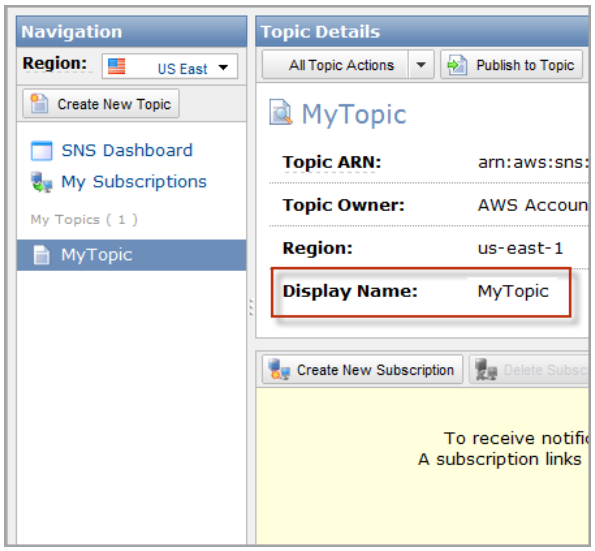
3. All Topic Actions 드롭다운 목록에서 Edit Topic Display Name을 선택합니다.



4. Display Name 상자에 표시 이름을 입력하고 Set Display Name을 클릭합니다.



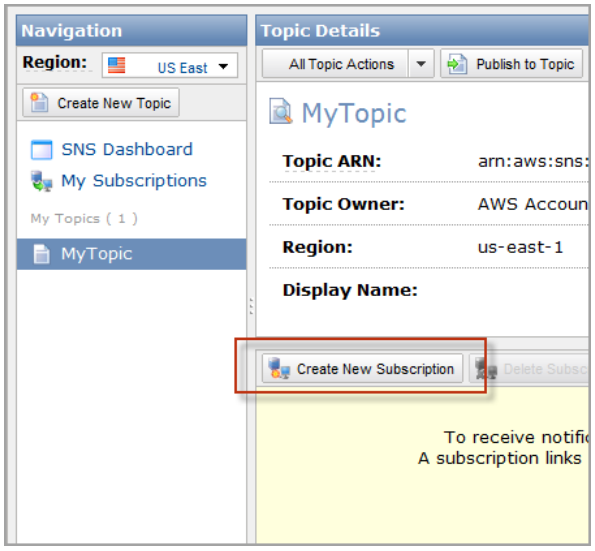
새 주제 표시 이름이 Topic Details페이지에 표시됩니다.



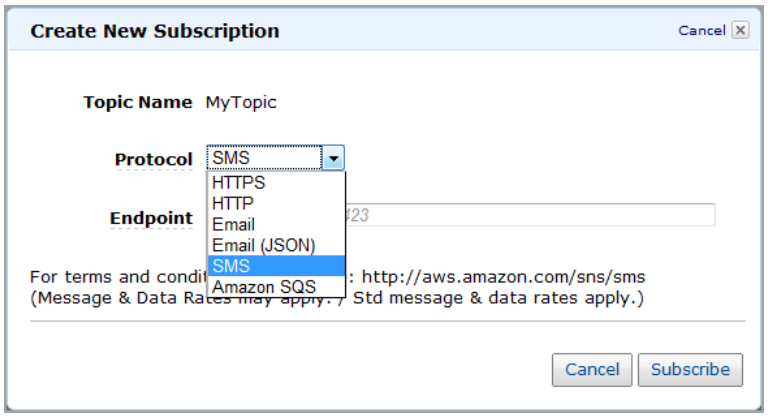
작업 2: SMS 프로토콜을 이용한 주제 구독

SMS 프로토콜을 이용한 Amazon SNS 주제 구독

1. Sign in to the AWS Management Console and open the Amazon SNS console at https://console.aws.amazon.com/sns/.
2. 검색 창에서 주제를 선택하고 Topic Details 창에서 Create New Subscription을 클릭합니다.



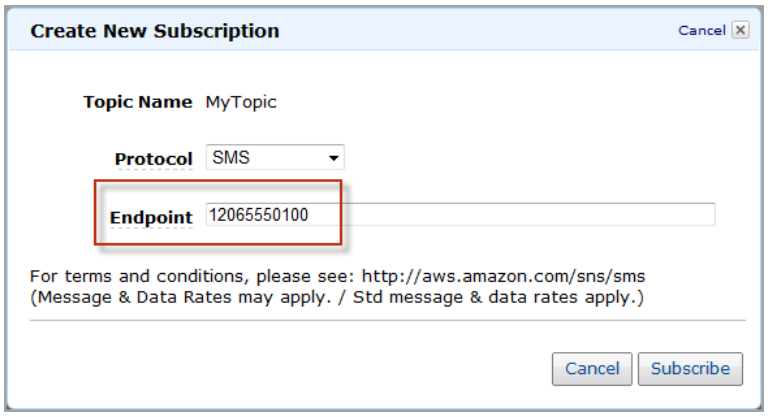
3. Protocol 드롭다운 목록에서 SMS를 선택합니다.



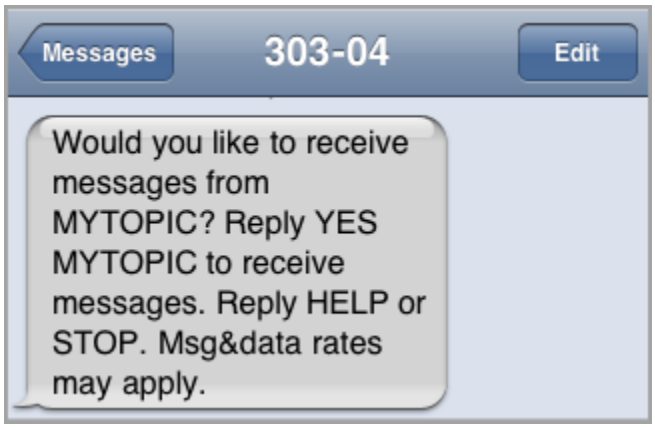
4. Endpoint 상자에 SMS 수신 가능한 디바이스의 전화 번호를 입력하고 Subscribe를 클릭합니다.

   Note

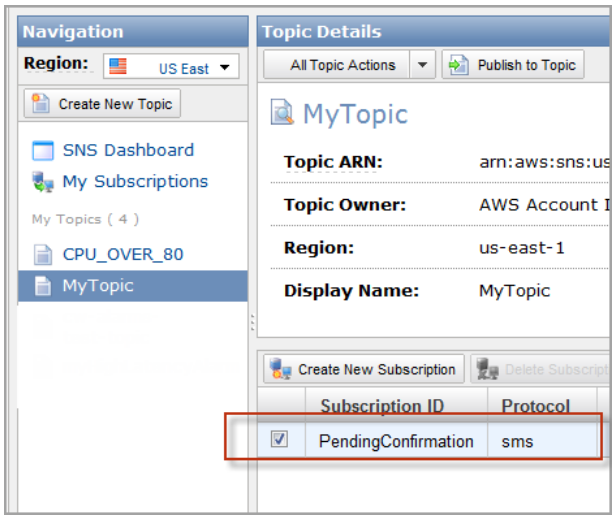
   번호만 사용합니다. 대시, 공백, 괄호를 사용하지 마십시오.



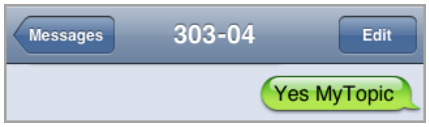
Amazon SNS는 입력한 번호의 SMS 수신 가능한 디바이스에 확인 문자 메시지를 전송합니다.



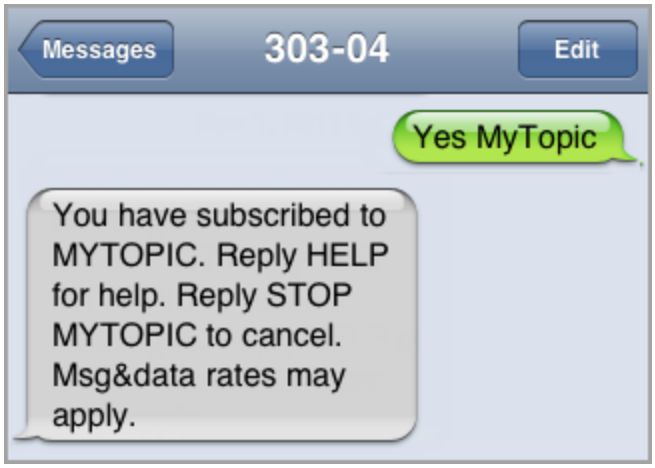
SMS 수신 가능한 디바이스가 구독을 확인할 때까지 AWS Management Console에서 구독은 PendingConfirmation으로 나열됩니다.



5. 확인 문자 메시지에 적극적으로 응답하기 위해 이전 단계에서 입력한 번호의 SMS 수신 가능한 디바이스를 사용합니다. 예를 들어 다음의 문자 메시지는 `MyTopic` Amazon SNS 주제에 대한 구독을 확인합니다.



Amazon SNS는 구독 확인 메시지로 응답합니다.

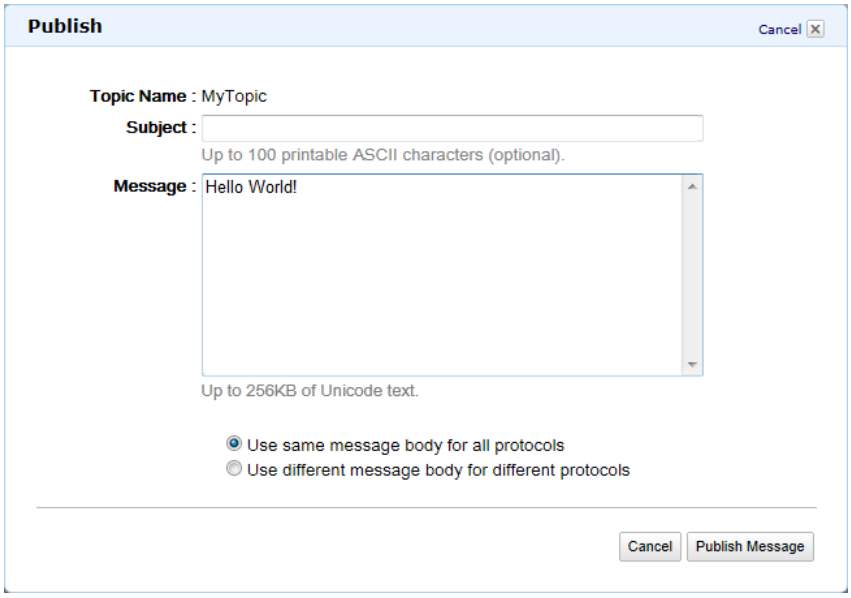


작업 3: 메시지 게시

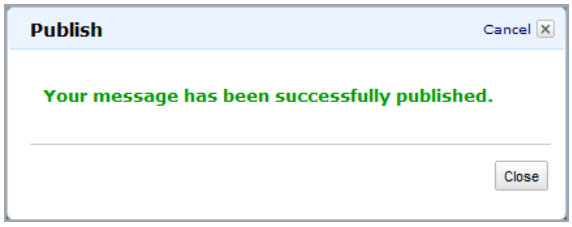
주제에 대한 메시지 게시

1. Sign in to the AWS Management Console and open the Amazon SNS console at https://console.aws.amazon.com/sns/.
2. 검색 창에서 주제를 선택합니다.

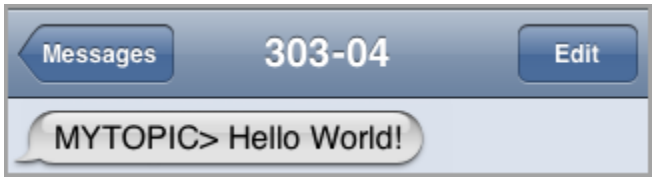
3. Publish to Topic을 클릭합니다.
4. Message 상자에 문자를 입력합니다.
   Amazon SNS는 Message 상자에 입력한 문자를 SMS 구독자들에게 전송합니다(Subject 상자에 문자를 입력하지 않을 경우). 다음의 예는 문자 `Hello World!`로 메시지를 생성합니다.



5. Subject 상자에 문자를 입력합니다(구독자에게 보낼 이메일용으로 Message 상자를 사용하고자 하는 경우).
   Subject 상자에 문자를 포함할 경우 SMS 메시지는 Message 상자의 문자보다는 제목 문자로 구성됩니다. 그러나 모든 이메일 구독자는 제목 및 메시지 본문 모두를 수신합니다. 따라서 사용자는 하나의 게시된 메시지를 사용해 제목을 사용한 짧은 SMS 메시지 및 메시지 페이로드를 사용한 긴 이메일 메시지를 전송할 수 있습니다.
6. Publish Message를 클릭합니다.
   Amazon SNS는 확인 대화 상자를 표시합니다.



SMS 메시지는 SMS 수신 가능한 디바이스에 표시됩니다.

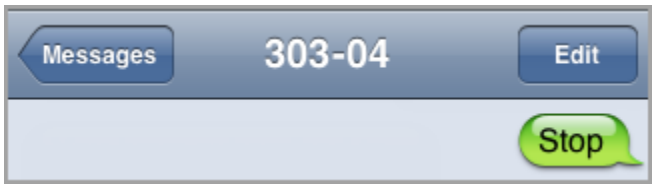


작업 4: SMS 구독 취소

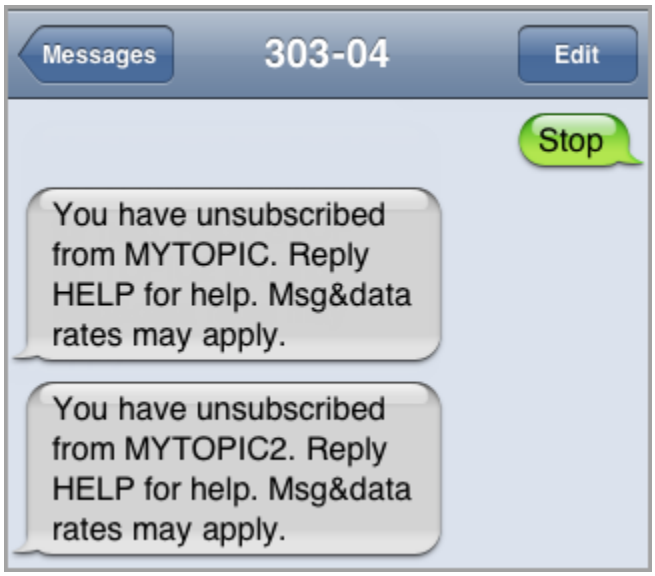
주제의 SMS 구독 취소를 위한 몇 가지 옵션이 있습니다. 단축 번호 30304에 대한 `STOP` 또는 `QUIT`에 응답함으로써 모든 SMS 메시지의 수신을 중지할 수 있습니다. 특정 주제의 구독을 취소하려면 `STOP <TOPICNAME>`를 담고 있는 SMS 메시지를 단축 번호 `30304`으로 전송합니다(주제의 표시 이름은 `<TOPICNAME>`임). AWS Management Console 또는 Query API Unsubscribe 작업을 통해서도 구독을 취소할 수 있습니다.

Amazon SNS의 모든 SMS 메시지 수신 중단

- SMS 수신 가능한 디바이스를 사용하여 `STOP` 또는 `QUIT` 메시지를 단축 번호 `30304`에 전송합니다. 예를 들어, 다음의 문자 메시지는 이 디바이스가 Amazon SNS에 보유한 구독을 모두 취소합니다.

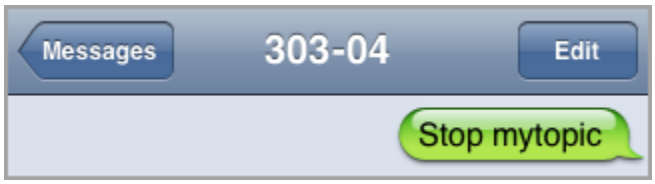


Amazon SNS는 각 주제에 대한 확인 메시지로 응답합니다.

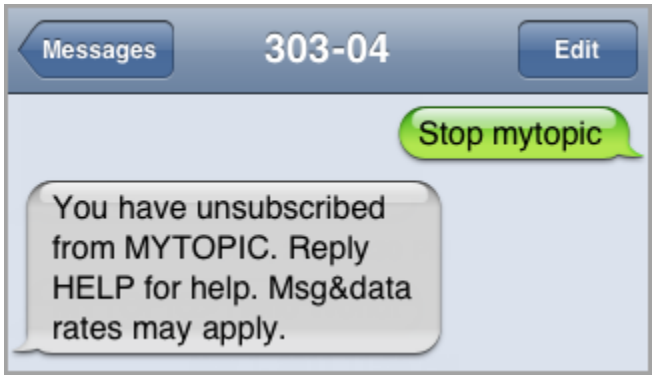


특정 주제의 SMS 메시지 수신 중단

- SMS 수신 가능한 디바이스를 사용하여 `STOP <TOPICNAME>`를 담고 있는 SMS 메시지를 단축 번호 `30304`로 전송합니다(주제의 표시 이름은 `<TOPICNAME>`임). 예를 들어, 다음의 SMS 메시지는 mytopic이라는 이름의 주제에 대한 구독을 취소합니다.



Amazon SNS는 확인 메시지로 응답합니다.



Amazon SNS 메시지를 HTTP/HTTPS Endpoint로 전송

Amazon SNS를 사용하여 하나 이상의 HTTP endpoint 또는 HTTPS endpoint에 알림 메시지를 전송할 수 있습니다. 주제에 대한 endpoint를 구독할 때 사용자는 주제에 대한 알림을 게시할 수 있으며 Amazon SNS는 HTTP POST 요청을 전송하는 한편 구독된 endpoint에 알림 콘텐츠를 전송합니다. 사용자는 endpoint 구독 시 Amazon SNS가 endpoint에 POST 요청을 전송하기 위해 HTTP를 사용하는지 또는 HTTPS를 사용하는지 여부를 선택합니다. 요청은 JSON 문서의 알림에 대한 메타데이터와 함께 주제에 게시된 제목 및 메시지를 포함합니다. 요청은 다음의 HTTP POST 요청과 유사합니다. 요청 본문의 HTTP 헤더 및 JSON 형식에 대한 세부 정보는 HTTP/HTTPS 헤더 (p. 132) 및 HTTP/HTTPS 알림 JSON 형식 (p. 135)을 참조하십시오.

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: da41e39f-ea4d-435a-b922-c6aae3915ebe
x-amz-sns-topic-arn: arn:aws:sns:us-east-1:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-east-1:123456789012:MyTopic:2bcfbf39-
05c3-41de-beaa-fcfcc21c8f55
Content-Length: 761
Content-Type: text/plain; charset=UTF-8
Host: ec2-50-17-44-49.compute-1.amazonaws.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "da41e39f-ea4d-435a-b922-c6aae3915ebe",
  "TopicArn" : "arn:aws:sns:us-east-1:123456789012:MyTopic",
  "Subject" : "test",
  "Message" : "test message",
  "Timestamp" : "2012-04-25T21:49:25.719Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLElDMXvB8r9R83tGoNn0ecwd5UjllzsvS
vbItzfaMpN2nk5HVSw7XnOn/49IkxDKz8YrlH2qJXj2iZB0Zo2O71c4qQk1fMUDi3LG
pij7RCW7AW9vYYsSqIKRnFS94ilu7NFhUzLiieYr4BKHpdTmdD6c0esKEYBpabxDSc=",
  "SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotificationSer
vice-f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&Sub
```

```
scriptionArn=arn:aws:sns:us-east-1:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-
fcfcc21c8f55"
}
```

HTTP endpoint 또는 HTTPS endpoint에 메시지를 전송하기 위해 Amazon SNS 주제를 활성화하려면 다음의 단계를 따라야합니다.

단계 1: Amazon SNS 메시지를 처리하도록 endpoint를 준비합니다.

사용자는 주제에 대한 HTTP endpoint 또는 HTTPS endpoint를 구독하기 전에 HTTP endpoint 또는 HTTPS endpoint가 구독 확인 및 알림 메시지를 전송하기 위해 Amazon SNS가 사용하는 HTTP POST 요청을 처리할 수 있는 능력이 있는지 확인해야 합니다. 일반적으로 이는 Amazon SNS에서 HTTP 요청을 처리하는 웹 애플리케이션의 생성 및 배포(예. endpoint host가 Linux, Apache 및 Tomcat을 실행할 경우 Java servlet)를 의미합니다. HTTP endpoint 구독 시 Amazon SNS는 이를 구독 확인 요청에 전송합니다. 사용자가 구독을 생성할 때 Amazon SNS는 이 요청을 전송하므로 endpoint는 이 요청을 수신하고 처리하도록 준비되어야 합니다. Amazon SNS는 사용자가 구독을 확인하기 전에는 endpoint에 알림을 전송하지 않습니다. 구독을 확인하고 나면 Amazon SNS는 구독하는 주제에 대해 게시 작업이 수행될 때 endpoint에 알림을 전송합니다.

구독 확인 및 알림 메시지 처리를 위한 endpoint 설정

1. 사용자의 코드는 Amazon SNS가 endpoint에 전송한 HTTP POST 요청의 HTTP 헤더를 읽어야 합니다. 사용자의 코드는 Amazon SNS가 사용자에게 전송한 메시지 유형을 나타내는 헤더 필드 `x-amz-sns-message-type`을 찾아야 합니다. 헤더를 확인함으로써 사용자는 HTTP 요청의 본문을 분석하지 않고도 메시지 유형을 결정할 수 있습니다. 처리해야 할 두 가지 유형 `SubscriptionConfirmation`와 `Notification`가 있습니다. `UnsubscribeConfirmation` 메시지는 주제에서 구독이 삭제된 때에만 사용됩니다.

   HTTP 헤더에 대한 세부 정보는 HTTP/HTTPS 헤더 (p. 132)를 참조하십시오. 다음의 HTTP POST 요청은 구독 확인 메시지의 예입니다.

```
POST / HTTP/1.1
x-amz-sns-message-type: SubscriptionConfirmation
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-east-1:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-east-1:123456789012:MyTopic:2bcf
bf39-05c3-41de-beaa-fcfcc21c8f55
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: example.com
```

```
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "SubscriptionConfirmation",
  "MessageId" : "165545c9-2a5c-472c-8df2-7ff2be2b3b1b",
  "Token" :
"2336412f37fb687f5d51e6e241d09c805a6a57808d712f794c5f6a8866b1276ab0a74ba6f3ab71854285dad248b09eece2917f1f02d69c582af
bacc99c583a916b9981dd2728f4ae6fdb82efd087cc3b7849e05798d2d2785c03b0879594eeac82c01f235d0e717736",

  "TopicArn" : "arn:aws:sns:us-east-1:123456789012:MyTopic",
  "Message" : "You have chosen to subscribe to the topic arn:aws:sns:us-
east-1:123456789012:MyTopic.\nTo confirm the subscription, visit the Sub
scribeURL included in this message.",
  "SubscribeURL" : "https://sns.us-east-1.amazonaws.com/?Action=ConfirmSub
scription&TopicArn=arn:aws:sns:us-east-1:123456789012:MyTop
ic&Token=2336412f37fb687f5d51e6e241d09c805a6a57808d712f794c5f6a8866b1276ab0a74ba6f3ab71854285dad248b09eece2917f1f02d69c582af
bacc99c583a916b9981dd2728f4ae6fdb82efd087cc3b7849e05798d2d2785c03b0879594eeac82c01f235d0e717736",

  "Timestamp" : "2012-04-26T20:45:04.751Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEpH+DcEwjAPg8O9mY8dReBSwksfg2S7WKQcikcNK
WLQjwu6A4VbeS0QHVCkhRS7fUQvi2egU3N858fiTDN6bkkOxYDVrY0Ad8L10Hs3zH81mtnPk5uvvol
IC1CXGu43obcgFxeL3khZl8IKvO61GWB6jI9b5+gLPoBc1Q=",
  "SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotification
Service-f3ecfb7224c7233fe7bb5f59f96de52f.pem"
  }
```

2. 사용자의 코드는 HTTP POST 요청 본문의 JSON 문서를 분석하여 Amazon SNS 메시지를 구성하는 이름/값 쌍을 읽어야 합니다. JSON 분석을 사용해 제어 문자의 escaped 상태를 ASCII 문자 값으로 변환 처리합니다(예. \n을 줄바꿈 문자로 변환). 사용자는 Jackson JSON Processor와 같은 기존 JSON 분석(http://wiki.fasterxml.com/JacksonHome)을 사용하거나 자체적으로 쓸 수 있습니다. 제목 및 메시지 필드의 문자를 유효한 JSON으로 보내려면 Amazon SNS는 제어 문자를 JSON 문서에 포함될 수 있는 escaped 상태로 변환해야 합니다. endpoint에 전송된 POST 요청의 본문에 있는 JSON 문서를 수신하면 사용자는 escaped 문자를 다시 원본 문자 값으로 변환해야 합니다. 서명은 원본 형식의 메시지 및 제목을 서명할 스트링의 일부로 사용하기 때문에 이는 알림의 서명을 확인하고자 할 경우 아주 중요합니다.
3. 또는 Amazon SNS가 보낸 알림의 신뢰성, 구독 확인 또는 구독 해지 확인 메시지를 확인할 수 있습니다. Endpoint는 Amazon SNS 메시지에 포함된 정보를 사용해 서명을 재생성할 수 있으며 사용자는 Amazon SNS가 메시지로 보낸 서명을 자신의 서명과 비교함으로써 메시지의 콘텐츠를 확인할 수 있습니다. 메시지의 서명 확인에 대한 자세한 내용은 Amazon SNS 메시지의 서명 확인 (p. 125)을 참조하십시오.
4. 사용자의 코드는 헤더 필드 `x-amz-sns-message-type`가 지정하는 유형에 기초하여 HTTP 요청의 본문에 담긴 JSON 문서를 읽고 메시지를 처리해야 합니다. 다음은 두 가지 주요 메시지 유형 처리 지침입니다.

   SubscriptionConfirmation
   : *`SubscribeURL`*의 값을 읽고 해당 URL을 방문합니다. 구독을 확인하고 endpoint에서 알림 수신을 시작하려면 *`SubscribeURL`* URL을 방문해야 합니다(예. URL에 HTTP GET 요청을 전송). *`SubscribeURL`*를 확인하려면 이전 단계의 HTTP 요청 예제를 확인합니다. *`SubscriptionConfirmation`*메시지의 형식에 대한 자세한 내용은 HTTP/HTTPS 구독 확인 JSON 형식 (p. 133)을 참조하십시오. URL을 방문하면 다음의 XML 문서와 같은 응답을 받게 됩니다. 문서는 `ConfirmSubscriptionResult` 요소 내 endpoint에 대한 구독 ARN을 되돌립니다.

```
<ConfirmSubscriptionResponse xmlns="http://sns.amazonaws.com/doc/2010-
03-31/">
  <ConfirmSubscriptionResult>
    <SubscriptionArn>arn:aws:sns:us-east-1:123456789012:MyTopic:2bcfbf39-
05c3-41de-beaa-fcfcc21c8f55</SubscriptionArn>
  </ConfirmSubscriptionResult>
  <ResponseMetadata>
    <RequestId>075ecce8-8dac-11e1-bf80-f781d96e9307</RequestId>
  </ResponseMetadata>
  </ConfirmSubscriptionResponse>
```

사용자는 *SubscribeURL* 방문 외에도 ConfirmSubscription 작업과 함께 *Token*(*SubscriptionConfirmation* 메시지의 해당 값으로 설정된)을 사용해 구독을 확인할 수 있습니다. 주제 소유자 및 구독 소유자만 endpoint를 구독 해지할 수 있도록 허용하고자 하는 경우 AWS 서명을 사용해 *ConfirmSubscription* 작업을 호출합니다.

Notification

`Subject` 및 `Message`에 대한 값을 읽어 주제에 게시된 알림 정보를 획득합니다.

`Notification` 메시지 형식에 대한 세부 정보는 HTTP/HTTPS 헤더 (p. 132)를 참조하십시오. 다음의 HTTP POST 요청은 endpoint example.com에 전송된 알림 메시지의 예입니다.

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324
x-amz-sns-topic-arn: arn:aws:sns:us-east-1:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-east-1:123456789012:MyTop
ic:c9135db0-26c4-47ec-8998-413945fb5a96
Content-Length: 773
Content-Type: text/plain; charset=UTF-8
Host: example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324",
  "TopicArn" : "arn:aws:sns:us-east-1:123456789012:MyTopic",
  "Subject" : "My First Message",
  "Message" : "Hello world!",
  "Timestamp" : "2012-05-02T00:54:06.655Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEw6JRNwm1LFQL4ICB0bnXrdB8ClRMTQFGBqwLp
GbM78tJ4etTwC5zU7O3tS6tGpey3ejedNdOJ+1fkIp9F2/LmNVKb5aFlYq+9rk9ZiPph5YlL
mWsDcyC5T+Sy9/umic5S0UQc2PEtgdpVBahwNOdMW4JPwk0kAJJztnc=",
  "SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotific
ationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-east-1.amazonaws.com/?Action=Unsub
scribe&SubscriptionArn=arn:aws:sns:us-east-1:123456789012:MyTop
ic:c9135db0-26c4-47ec-8998-413945fb5a96"
  }
```

5. Endpoint가 적절한 상태 코드로 Amazon SNS의 HTTP POST 메시지에 응답하도록 확인해야 합니다. 연결은 15초 내에 끊어집니다. endpoint가 연결 시간이 끊기기 전까지 응답하지 않거나 endpoint가 상태 코드를 200~4*xx* 범주 외로 되돌릴 경우 Amazon SNS는 메시지 전송이 실패한 것으로 간주합니다.

6. Amazon SNS로부터의 메시지 전송 재시도를 처리할 수 있도록 코드를 확인해야 합니다. Amazon SNS는 endpoint에서 성공적인 응답을 수신하지 않을 경우 메시지를 다시 전송하는 시도를 합니다. 이는 구독 확인 메시지를 포함한 모든 메시지에 적용됩니다. 기본 설정에서 메시지의 초기 전송이 실패할 경우 Amazon SNS는 최대 3회 재시도를 실시하며 실패한 시도 사이에 설정된 지연 시간은 20초 입니다. 메시지 요청 시간은 15초 입니다. 따라서 시간 지연으로 인한 메시지 전송 실패의 경우 Amazon SNS는 이전 전송 시도의 약 35초 후에 재시도를 합니다. 사용자가 기본 전송 정책을 선호하지 않을 경우 endpoint에서 다른 전송 정책을 설정할 수 있습니다.

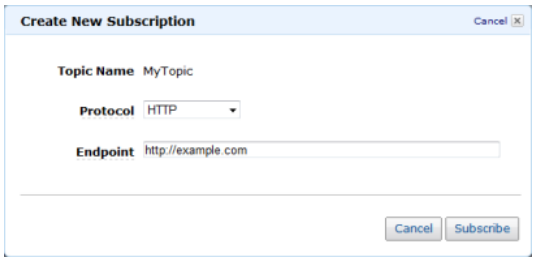
   다시 말하자면, Amazon SNS는 전송이 실패한 이후에만 재전송을 시도합니다. 사용자는 `x-amz-sns-message-id` 헤더 필드를 사용하여 메시지를 파악할 수 있습니다. 사용자는 수신 메시지와 처리한 메시지의 ID를 비교함으로써 메시지가 재전송을 시도하는지 여부를 판단할 수 있습니다.
7. HTTPS endpoint를 구독할 경우 endpoint가 신뢰할 수 있는 Certificate Authority(CA)로부터의 서버 인증서를 보유하는지 확인해야 합니다. Amazon SNS는 Amazon SNS가 신뢰하는 CA가 승인한 서버 인증서를 보유한 HTTPS endpoint에만 메시지를 보냅니다. 신뢰할 수 있는 CA 목록은 HTTPS Endpoint에 대해 Amazon SNS가 인정하는 Certificate Authorities(CA) (p. 121)를 참조하십시오.
8. Amazon SNS 메시지를 수신하기 위해 생성한 코드를 배포합니다. endpoint를 구독할 때 endpoint는 적어도 구독 확인 메시지를 받을 준비가 되어야 합니다.

단계 2: Amazon SNS 주제에 대한 HTTP/HTTPS endpoint 구독

주제를 통해 메시지를 HTTP endpoint 또는 HTTPS endpoint에 전송하려면 사용자는 Amazon SNS 주제에 대한 endpoint를 구독해야 합니다. 사용자는 해당 URL을 사용하여 endpoint를 지정합니다. 주제를 구독하기 위해 Amazon SNS 콘솔, sns-subscribe 명령 또는 Subscribe API 작업을 사용할 수 있습니다. 시작하기 전에 사용자는 구독하고자 하는 endpoint의 URL을 보유하는지 확인해야 하며 단계 1에서 설명한 대로 endpoint가 확인 및 알림 메시지를 수신할 수 있도록 준비되어야 합니다.

Amazon SNS 콘솔을 이용한 주제에 대한 HTTP endpoint 또는 HTTPS endpoint 구독

1. Sign in to the AWS Management Console and open the Amazon SNS console at https://console.aws.amazon.com/sns/.
2. 검색 창에서 주제를 선택합니다.
3. Create New Subscription을 클릭하고, Protocol의 HTTP 또는 HTTPS를 선택하고, Endpoint에 대해 주제가 메시지를 전송할 endpoint의 URL을 붙여넣은 후 Subscribe를 클릭합니다.



4. Subscription request received! 메시지가 표시되면 Close를 클릭합니다.

   새 구독의 Subscription ID는 PendingConfirmation에 표시됩니다. 구독 확인 시 Subscription ID는 Subscription ID에 표시됩니다.

3단계: 구독 확인

Endpoint를 구독한 후 Amazon SNS는 endpoint에 구독 확인 메시지를 전송합니다. 사용자는 이미 Step 1 (p. 105)에서 설명한 작업을 수행하는 endpoint에 배포될 코드를 보유해야 합니다. 구체적으로 endpoint의 코드는 구독 확인 메시지로부터의 *SubscribeURL* 값을 검색하고 *SubscribeURL*이 자체적으로 지정한 위치를 방문하거나 사용자가 수동으로 *SubscribeURL*(예. 웹 브라우저)를 방문할 수 있도록 해야 합니다. Amazon SNS는 구독이 확인되기 전에는 endpoint에 메시지를 보내지 않습니다. *SubscribeURL* 방문 시, 응답은 구독에 대한 ARN을 지정하는 요소 *SubscriptionArn*를 담고 있는 XML 문서를 포함합니다. 사용자는 Amazon SNS 콘솔을 사용하여 구독 확인을 검증할 수도 있습니다. Subscription ID는 구독에 대한 ARN을 표시합니다(구독을 처음 추가했을 때 확인한 *PendingConfirmation* 값 대신).

단계 4: 구독의 전송 재시도 정책 설정(옵션)

기본 설정에서 메시지의 초기 전송이 실패할 경우 Amazon SNS는 최대 3회 재시도를 실시하며 실패한 시도 사이에 설정된 지연 시간은 20초 입니다. 단계 1 (p. 105)에서 논의되었듯 endpoint는 재시도되는 메시지를 처리할 수 있는 코드를 보유해야 합니다. 사용자는 주제 또는 구독에 대한 전송 정책을 설정함으로써 Amazon SNS가 전송 실패 메시지를 재전송하는 빈도 및 간격을 제어할 수 있습니다. 전송 정책은 주제 또는 특정 구독에 대해 설정할 수 있습니다.

단계 5: 사용자에게 주제를 게시할 권한 부여(옵션)

기본 설정에서 주제 소유자는 주제를 게시할 권한을 갖습니다. 다른 사용자 및 애플리케이션을 활성화하여 주제를 게시하려면 AWS Identity and Access Management(IAM)을 사용하여 주제에 대한 게시 권한을 부여해야 합니다. IAM 사용자의 Amazon SNS 작업 권한 부여에 대한 자세한 내용은 Controlling User Access to Your AWS Account를 참조하십시오.

주제에 대한 액세스를 제어하는 다음의 두 가지 방법이 있습니다.

- IAM 사용자 또는 그룹에 정책을 추가합니다. 사용자에게 주제에 대한 권한을 부여하는 가장 간단한 방법은 그룹을 생성하고 그룹에 적절한 정책을 추가한 후 사용자를 추가하는 것입니다. 그룹에서 사용자를 추가하거나 제거하는 것이 각각의 사용자에 대해 설정한 정책을 추적하는 것보다 훨씬 쉽습니다.
- 주제에 정책을 추가합니다. 또 다른 AWS 계정에 주제에 대한 권한을 부여하고자 할 경우 사용할 수 있는 유일한 방법은 권한을 부여하고자 하는 AWS 계정과 같은 정책을 추가하는 것입니다.

사용자는 대부분의 경우 첫 번째 방법을 사용해야 합니다(그룹에 정책을 적용하여 해당 그룹에 적절한 사용자를 추가 및 제거함으로써 사용자에 대한 권한을 관리). 또 다른 계정의 사용자에게 권한을 부여해야 할 경우에는 두 번째 방법을 사용합니다.

IAM 사용자 또는 그룹에 다음의 정책을 추가할 경우 해당 사용자 또는 그룹 구성원에게 MyTopic에 대한 `sns:Publish` 작업을 수행할 권한을 부여할 것입니다.

```
{
  "Statement":[{
      "Sid":"AllowPublishToMyTopic",
      "Effect":"Allow",
      "Action":"sns:Publish",
      "Resource":"arn:aws:sns:us-east-1:123456789012:MyTopic"
    }
  ]
}
```

다음의 정책 예제는 또 다른 계정에 주제에 대한 권한을 부여하는 방법을 나타냅니다.

Note

사용자 계정에 있는 리소스에 대한 액세스 권한을 또 다른 AWS 계정에 부여할 때 해당 리소스에 대해 관리자 레벨의 액세스(와일드카드 액세스) 권한을 보유한 IAM 사용자에게도 권한을 부여합니다. 다른 계정의 다른 모든 IAM 사용자는 자동으로 사용자 리소스에 대한 액세스가 거부됩니다. 해당 AWS 계정의 특정 IAM 사용자에게 사용자의 리소스에 대한 액세스 권한을 부여하고자 할 경우 관리자 레벨의 액세스 권한을 보유한 계정 또는 IAM 사용자는 이러한 IAM 사용자들에게 리소스에 대한 권한을 위임해야 한다. 교차 계정 위임에 대한 자세한 정보는 Using IAM Guide의 *Enabling Cross-Account Access*를 참조하십시오.

계정 123456789012의 주제 MyTopic에 대해 다음의 정책을 추가했을 경우 사용자는 계정 111122223333에 해당 주제에 대한 `sns:Publish` 작업을 수행할 권한을 부여했을 것입니다.

```
{
  "Id":"MyTopicPolicy",
  "Statement":[{
      "Sid":"Allow-publish-to-topic",
      "Effect":"Allow",
      "Principal":{
        "AWS":"111122223333"
      },
      "Action":"sns:Publish",
      "Resource":"arn:aws:sns:us-east-1:123456789012:MyTopic"
    }
  ]
}
```

단계 6: HTTP/HTTPS endpoint에 메시지 전송

주제를 게시함으로써 주제의 구독에 메시지를 전송할 수 있습니다. 주제를 게시하려면 Amazon SNS 콘솔인 sns-publish 명령 또는 Publish API를 을 사용할 수 있습니다.

단계 1 (p. 105)을 따랐다면 endpoint에 배포한 코드는 알림을 처리해야 합니다.

Amazon SNS 콘솔을 사용하여 주제 게시

1. AWS 계정 또는 주제를 게시할 권한을 가진 IAM 사용자의 자격 증명을 통해 AWS Management Console에 등록하고 https://console.aws.amazon.com/sns/에서 Amazon SNS 콘솔을 엽니다.
2. 검색 창에서 주제를 선택하고 Publish to Topic을 클릭합니다.
3. Subject 상자에서 제목을 입력하고(예. **`Testing publish to my endpoint`**) Message 상자에서 문자를 입력하고(예. **`Hello world!`**) Publish Message를 클릭합니다. 다음의 메시지가 나타납니다. Your message has been successfully published.

Setting Amazon SNS Delivery Retry Policies for HTTP/HTTPS Endpoints

Topics

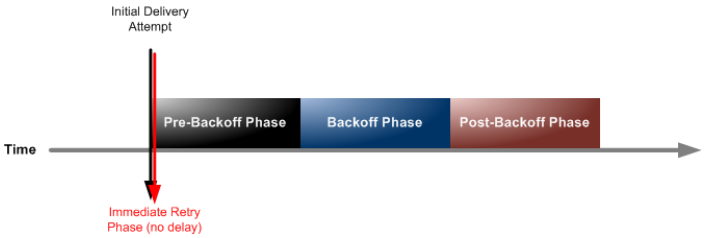
- Applying Delivery Policies to Topics and Subscriptions (p. 112)

A successful Amazon SNS delivery to an HTTP/HTTPS endpoint sometimes requires more than one attempt. This can be the case, for example, if the web server that hosts the subscribed endpoint is down for maintenance or is experiencing heavy traffic. If an initial delivery attempt doesn't result in a successful response from the subscriber, Amazon SNS attempts to deliver the message again. We call such an attempt a *retry*. In other words, a retry is an attempted delivery that occurs after the initial delivery attempt.

Amazon SNS only attempts a retry after a failed delivery attempt. Amazon SNS considers the following situations as a failed delivery attempt.

- HTTP status in the range 500-599.
- HTTP status outside the range 200-599.
- A request timeout (15 seconds). Note that if a request timeout occurs, the next retry will occur at the specified interval after the timeout. For example, if the retry interval is 20 seconds and a request times out, the start of the next request will be 35 seconds after the start of the request that timed out.
- Any connection error such as connection timeout, endpoint unreachable, bad SSL certificate, etc.

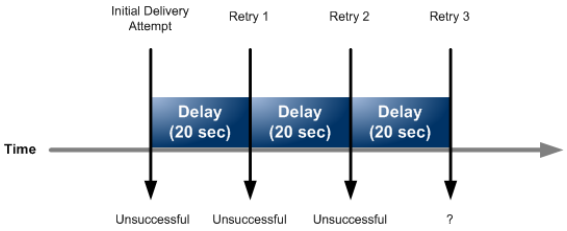
You can use delivery policies to control not only the total number of retries, but also the time delay between each retry. You can specify up to 100 total retries distributed among four discrete phases.



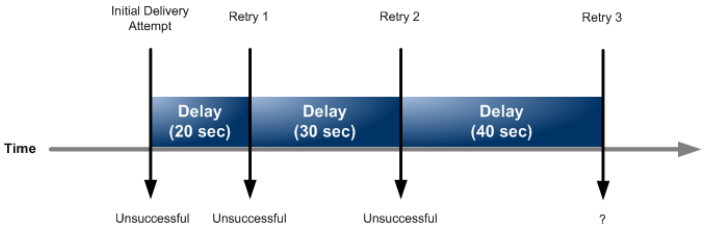
1. Immediate Retry Phase (p. 116)—Also called the *no delay phase*, this phase occurs immediately after the initial delivery attempt. The value you set for Retries with no delay determines the number of retries immediately after the initial delivery attempt. There is no delay between retries in this phase.
2. Pre-Backoff Phase (p. 117)—The pre-backoff phase follows the immediate retry phase. Use this phase to create a set of retries that occur before a backoff function applies to the retries. Use the Minimum delay retries setting to specify the number of retries in the Pre-Backoff Phase. You can control the time delay between retries in this phase by using the Minimum delay setting.
3. Backoff Phase (p. 118)—This phase is called the backoff phase because you can control the delay between retries in this phase using the retry backoff function. Set the Minimum delay and the Maximum delay, and then select a Retry backoff function to define how quickly the delay increases from the minimum delay to the maximum delay.
4. Post-Backoff Phase (p. 120)—The post-backoff phase follows the backoff phase. Use the Maximum delay retries setting to specify the number of retries in the post-backoff phase. You can control the time delay between retries in this phase by using the Maximum delay setting.

The backoff phase is the most commonly used phase. If no delivery policies are set, the default is to retry three times in the backoff phase, with a time delay of 20 seconds between each retry. The default value for both the Minimum delay and the Maximum delay is 20. The default number of retries is 3, so the default

retry policy calls for a total of 3 retries with a 20 second delay between each retry. The following diagram shows the delay associated with each retry.



To see how the retry backoff function affects the time delay between retries, you can set the maximum delay to 40 seconds and leave the remaining settings at their default values. With this change, your delivery policy now specifies 3 retries during the backoff phase, a minimum delay of 20 seconds, and a maximum delay of 40 seconds. Because the default backoff function is linear, the delay between messages increases at a constant rate over the course of the backoff phase. Amazon SNS attempts the first retry after 20 seconds, the second retry after 30 seconds, and the final retry after 40 seconds. The following diagram shows the delay associated with each retry.



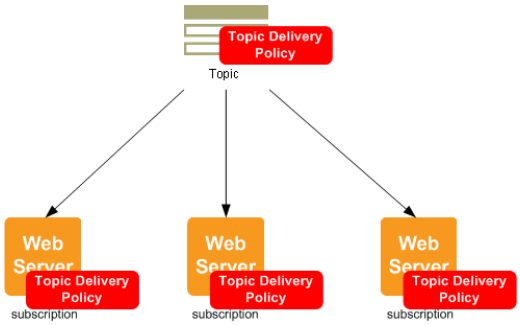
The maximum lifetime of a message in the system is one hour. This one hour limit cannot be extended by a delivery policy.

Note

Only HTTP and HTTPS subscription types are supported by delivery policies. Support for other Amazon SNS subscription types (e.g., email, Amazon SQS, and SMS) is not currently available.

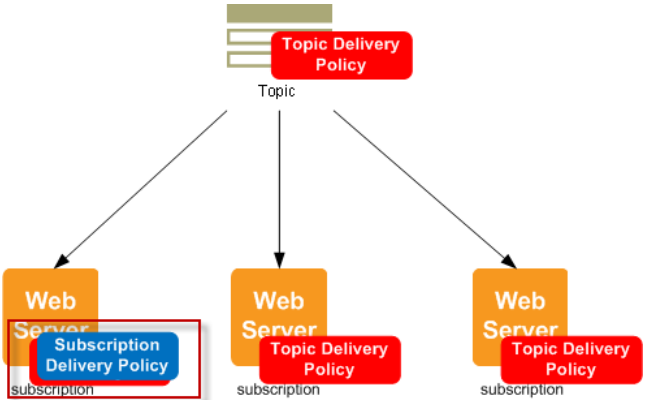
Applying Delivery Policies to Topics and Subscriptions

You can apply delivery policies to Amazon SNS topics. If you set a delivery policy on a topic, the policy applies to all of the topic's subscriptions. The following diagram illustrates a topic with a delivery policy that applies to all three subscriptions associated with that topic.

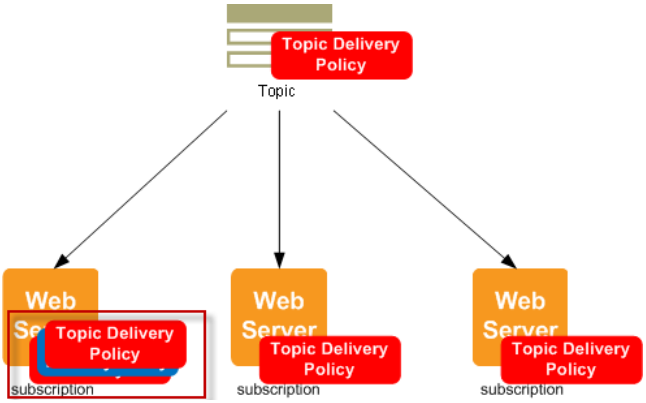


You can also apply delivery policies to individual subscriptions. If you assign a delivery policy to a subscription, the subscription-level policy takes precedence over the topic-level delivery policy. In the

following diagram, one subscription has a subscription-level delivery policy whereas the two other subscriptions do not.



In some cases, you might want to ignore all subscription delivery policies so that your topic's delivery policy applies to all subscriptions even if a subscription has set its own delivery policy. To configure Amazon SNS to apply your topic delivery policy to all subscriptions, click Ignore subscription override in the View/Edit Topic Delivery Policies dialog box. The following diagram shows a topic-level delivery policy that applies to all subscriptions, even the subscription that has its own subscription delivery policy because subscription-level policies have been specifically ignored.



Setting the Maximum Receive Rate

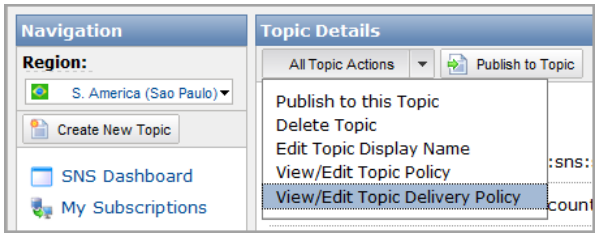
You can set the maximum number of messages per second that Amazon SNS sends to a subscribed endpoint by setting the Maximum receive rate setting. Amazon SNS holds messages that are awaiting delivery for up to an hour. Messages held for more than an hour are discarded.

- To set a maximum receive rate that applies to all of a topic's subscriptions, apply the setting at the topic level using the View/Edit Topic Delivery Policy dialog box. For more information, see To set the maximum receive rate for a topic (p. 113).
- To set a maximum receive rate that applies to a specific subscription, apply the setting at the subscription level using the View/Edit Subscription Delivery Policy dialog box. For more information, see To set the maximum receive rate for a subscription (p. 114).

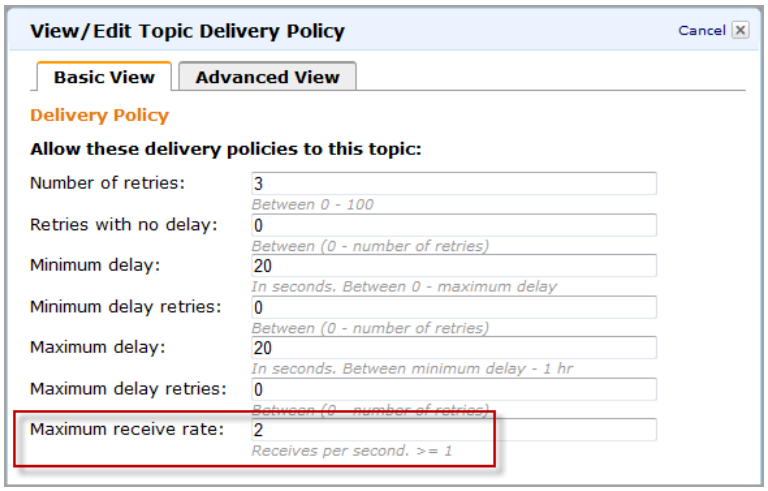
To set the maximum receive rate for a topic

1. Sign in to the AWS Management Console and open the Amazon SNS console at https://console.aws.amazon.com/sns/.

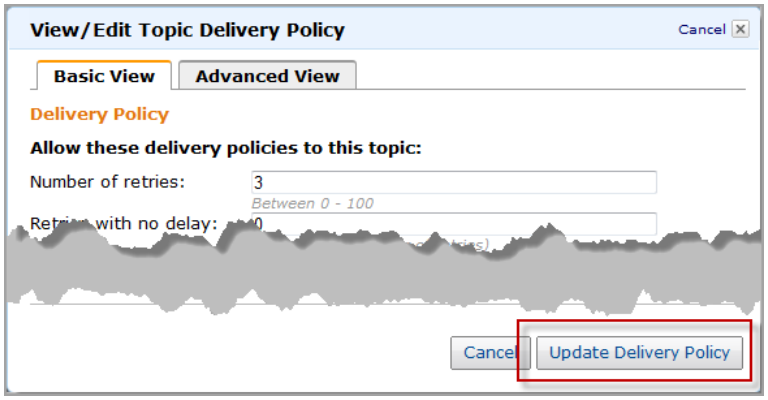
2. Select a topic in the Navigation pane, and in the Topic Details pane select View/Edit Topic Delivery Policy from the All Topic Actions list.



3. Type an integer value (e.g., 2) in the Maximum receive rate box.

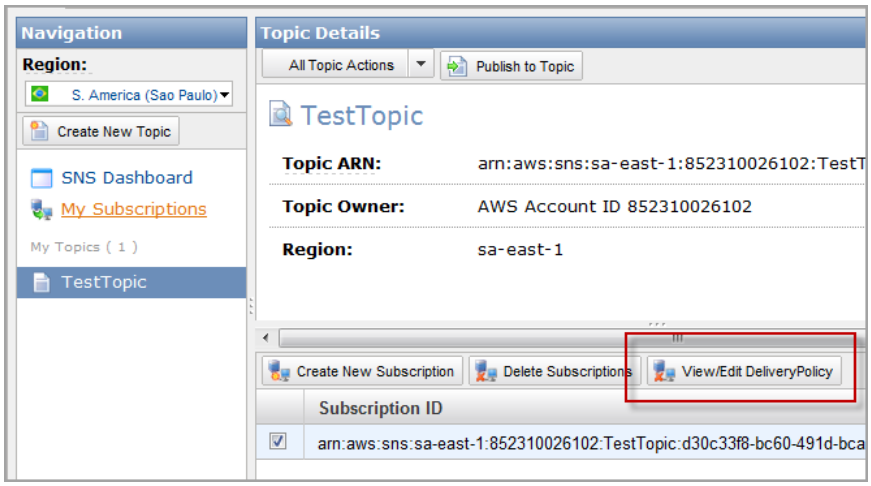


4. Click Update Delivery Policy to save your changes.

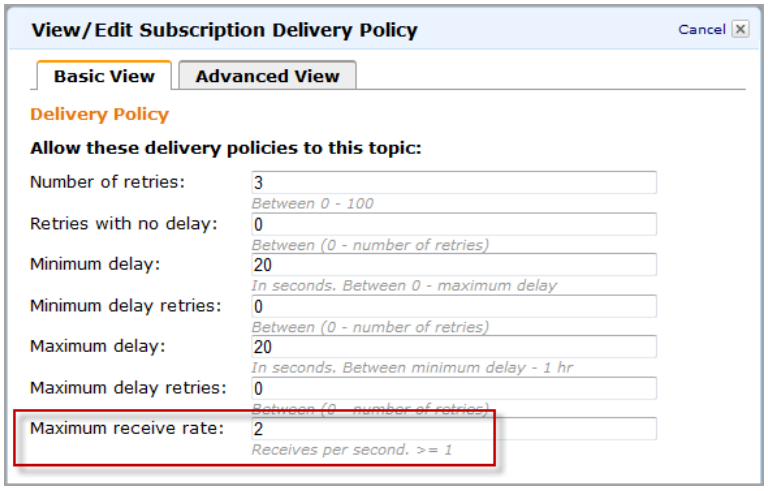


To set the maximum receive rate for a subscription

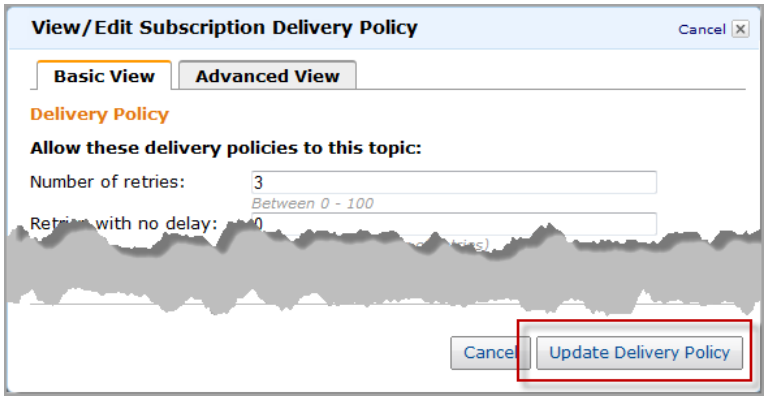
1. Sign in to the AWS Management Console and open the Amazon SNS console at https://console.aws.amazon.com/sns/.
2. Select a topic in the Navigation pane.
3. In the Topic Details pane, select a subscription and click View/Edit Delivery Policy.



4. Type an integer value (e.g., 2) in the Maximum receive rate box.



5. Click Update Delivery Policy to save your changes.

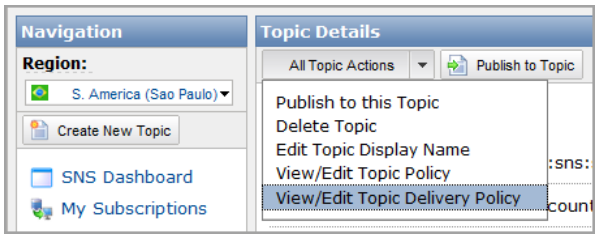


Immediate Retry Phase

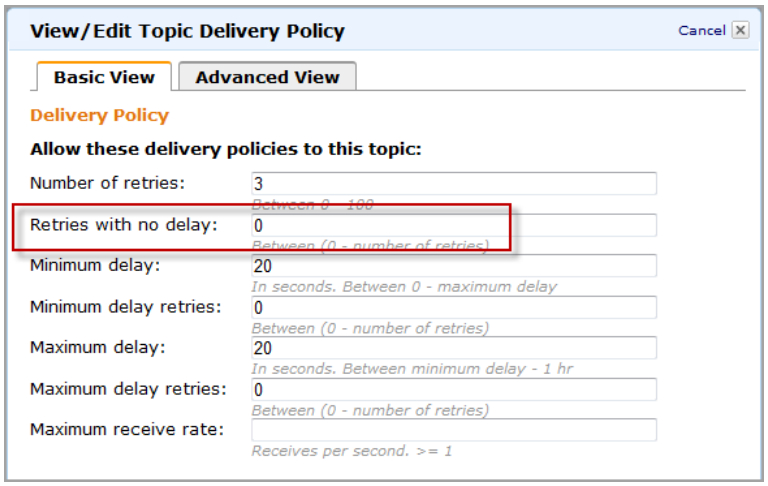
The immediate retry phase occurs directly after the initial delivery attempt. This phase is also known as the No Delay phase because it happens with no time delay between the retries. The default number of retries for this phase is 0.

To set the number of retries in the immediate retry phase

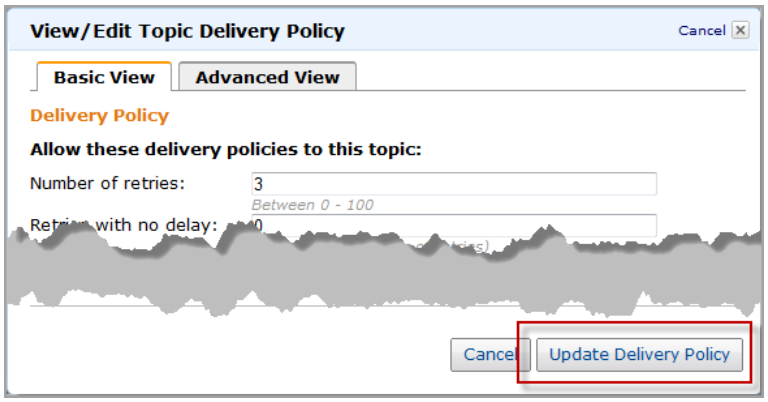
1. Sign in to the AWS Management Console and open the Amazon SNS console at https://console.aws.amazon.com/sns/.
2. Select a topic in the Navigation pane, and in the Topic Details pane select View/Edit Topic Delivery Policy from the All Topic Actions list.



3. Type an integer value in the Retries with no delay box.



4. Click Update Delivery Policy to save your changes.

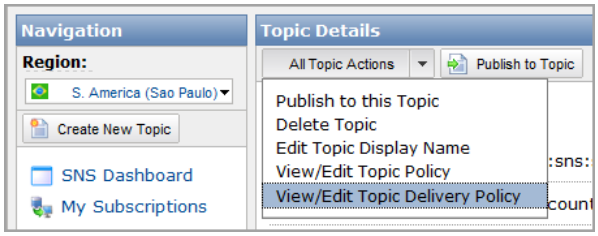


Pre-Backoff Phase

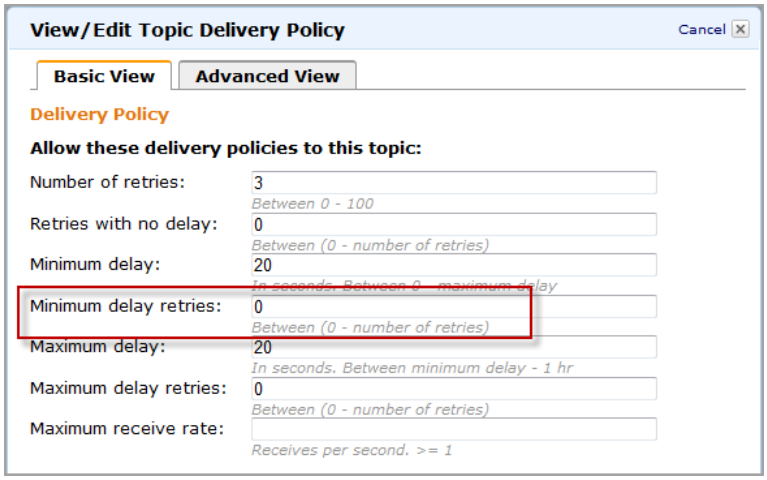
The pre-backoff phase follows the immediate retry phase. Use this phase if you want to create a set of one or more retries that happen before the backoff function affects the delay between retries. In this phase, the time between retries is constant and is equal to the setting that you choose for the Minimum delay. The Minumum delay setting affects retries in two phases—it applies to all retries in the pre-backoff phase and serves as the initial time delay for retries in the backoff phase. The default number of retries for this phase is 0.

To set the number of retries in the pre-backoff phase

1. Sign in to the AWS Management Console and open the Amazon SNS console at https://console.aws.amazon.com/sns/.
2. Select a topic in the Navigation pane, and in the Topic Details pane select View/Edit Topic Delivery Policy from the All Topic Actions list.

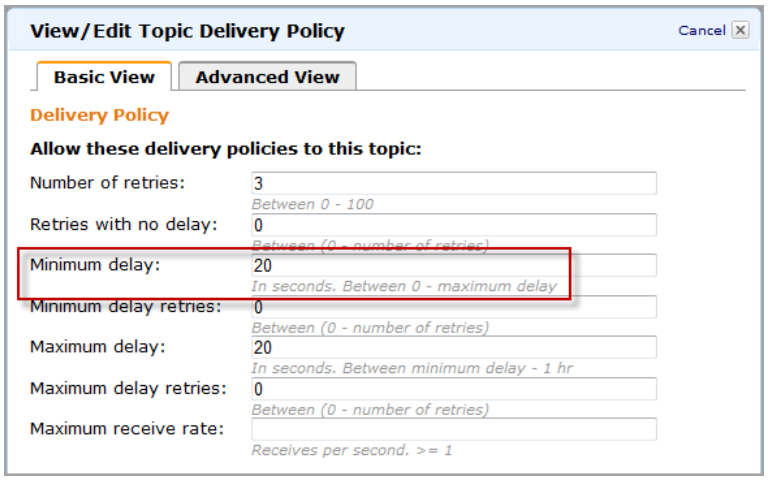


3. Type an integer value in the Minimum delay retries box.

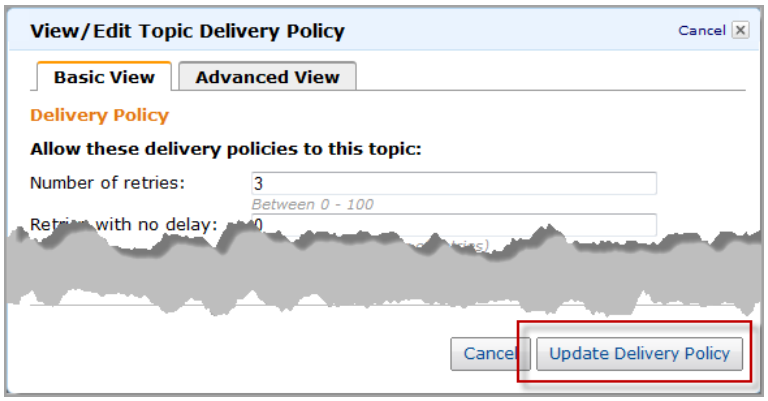


4. Type an integer value in the Minimum delay box to set the delay between messages in this phase.

   The value you set must be less than or equal to the value you set for Maximum delay.



5. Click Update Delivery Policy to save your changes.



Backoff Phase

The backoff phase is the only phase that applies by default. You can control the number of retries in the backoff phase using Number of retries.

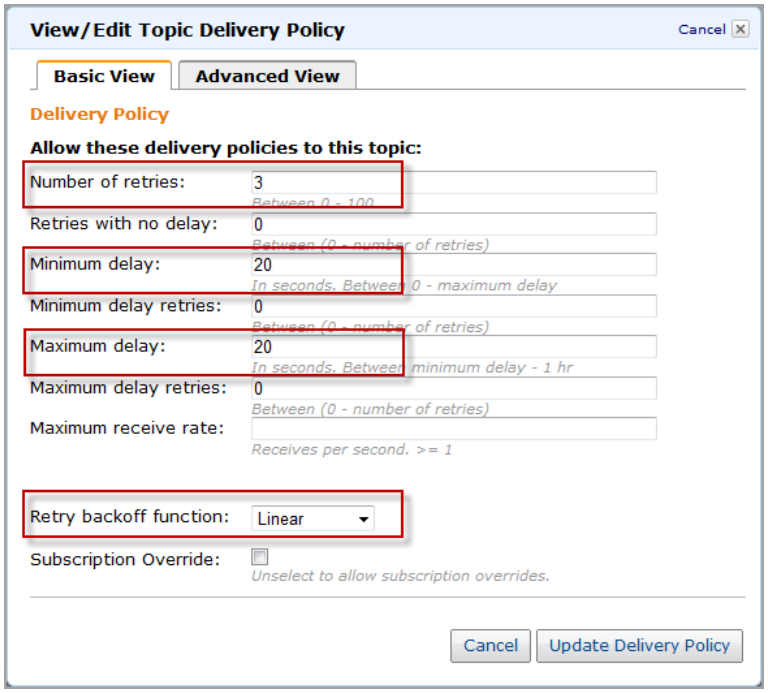
Important

The value you choose for Number of retries represents the total number of retries, including the retries you set for Retries with no delay, Minimum delay retries, and Maximum delay retries.

You can control the frequency of the retries in the backoff phase with three parameters.

- Minimum delay—The minimum delay defines the delay associated with the first retry attempt in the backoff phase.
- Maximum delay—The maximum delay defines the delay associated with the final retry attempt in the backoff phase.
- Retry backoff function—The retry backoff function defines the algorithm that Amazon SNS uses to calculate the delays associated with all of the retry attempts between the first and last retries in the backoff phase.

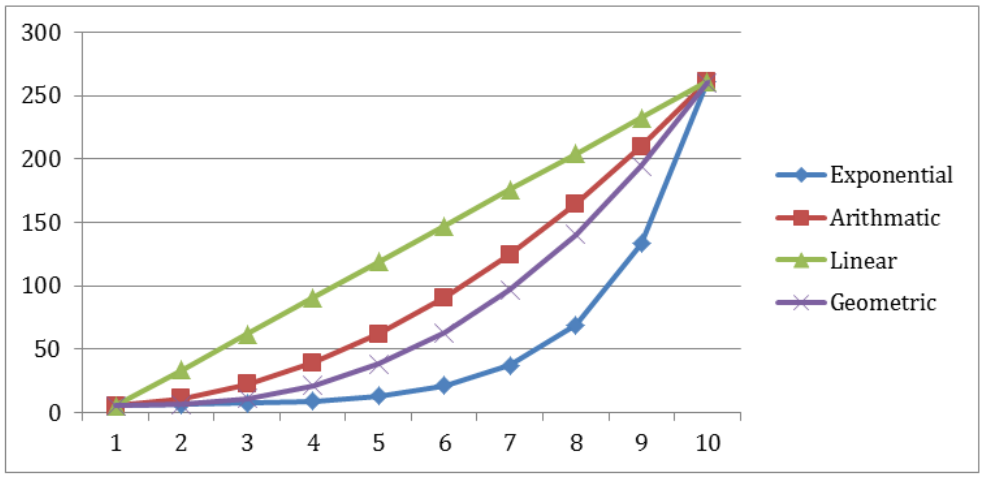
The following screen shot shows the Amazon SNS console fields that pertain to the backoff phase.



You can choose from four retry backoff functions.

- Linear
- Arithmetic
- Geometric
- Exponential

The following screen shot shows how each retry backoff function affects the delay associated with messages during the backoff period. The vertical axis represents the delay in seconds associated with each of the 10 retries. The horizontal axis represents the retry number. The minimum delay is 5 seconds, and the maximum delay is 260 seconds.

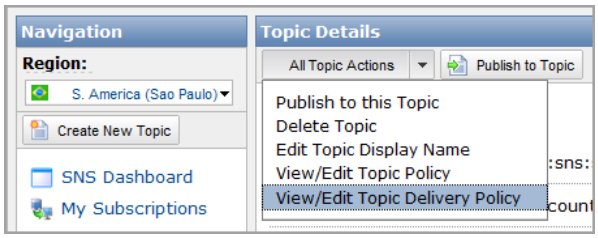


Post-Backoff Phase

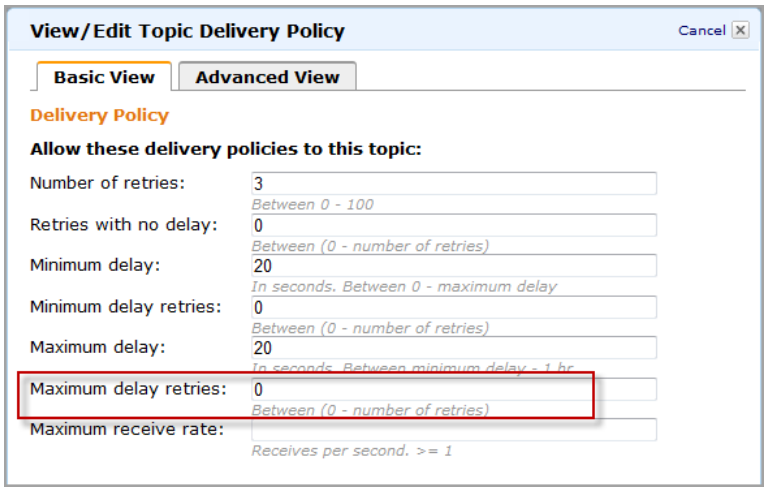
The post-backoff phase is the final phase. Use this phase if you want to create a set of one or more retries that happen after the backoff function affects the delay between retries. In this phase, the time between retries is constant and is equal to the setting that you choose for the Maximum delay. The Maximum delay setting affects retries in two phases—it applies to all retries in the post-backoff phase and serves as the final time delay for retries in the backoff phase. The default number of retries for this phase is 0.

To set the number of retries in the post-backoff phase

1. Sign in to the AWS Management Console and open the Amazon SNS console at https://console.aws.amazon.com/sns/.
2. Select a topic in the Navigation pane, and in the Topic Details pane select View/Edit Topic Delivery Policy from the All Topic Actions list.

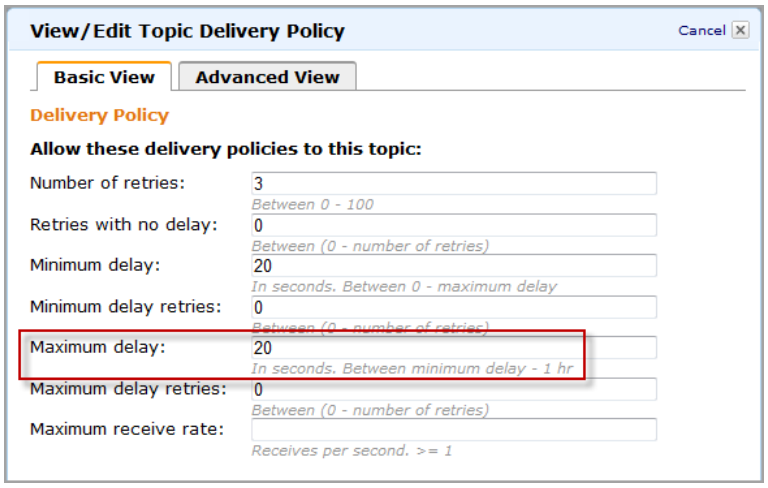


3. Type an integer value in the Maximum delay retries box.

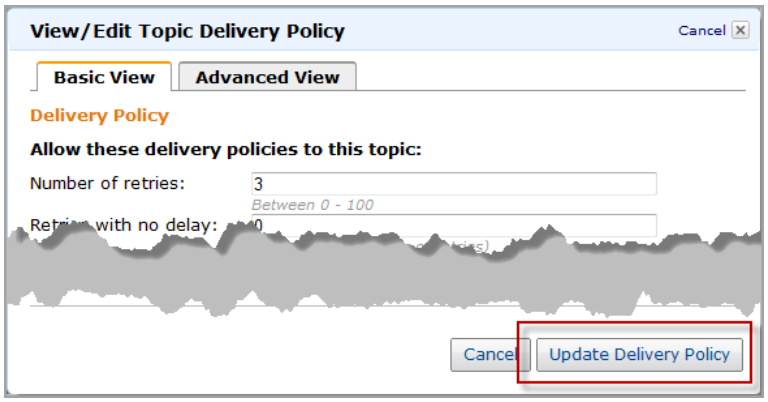


4. Type an integer value in the Maximum delay box to set the delay between messages in this phase.

   The value you set must be greater than or equal to the value you set for Minimum delay.



5. Click Update Delivery Policy to save your changes.



HTTPS Endpoint에 대해 Amazon SNS가 인정하는 Certificate Authorities(CA)

주제에 대한 HTTPS endpoint를 구독할 경우 endpoint는 신뢰할 수 있는 Certificate Authority(CA)에 의해 서명한 서버 인증서를 보유해야 합니다. Amazon SNS는 Amazon SNS가 인정하는 신뢰할 수 있는 CA에서 서명한 인증서를 보유한 HTTPS endpoints에만 메시지를 전송합니다. Amazon SNS는 다음의 CA를 인정합니다.

```
digicertassuredidrootca, Jan 7, 2008, trustedCertEntry,
        Certificate fingerprint (MD5):
87:CE:0B:7B:2A:0E:49:00:E1:58:71:9B:37:A8:93:72
        trustcenterclass2caii, Jan 7, 2008, trustedCertEntry,
        Certificate fingerprint (MD5):
CE:78:33:5C:59:78:01:6E:18:EA:B9:36:A0:B9:2E:23
        thawtepremiumserverca, Dec 2, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
A6:6B:60:90:23:9B:3F:2D:BB:98:6F:D6:A7:19:0D:46
        swisssignsilverg2ca, Aug 13, 2008, trustedCertEntry,
```

```
        Certificate fingerprint (MD5):
E0:06:A1:C9:7D:CF:C9:FC:0D:C0:56:75:96:D8:62:13
        swisssignplatinumg2ca, Aug 13, 2008, trustedCertEntry,
        Certificate fingerprint (MD5):
C9:98:27:77:28:1E:3D:0E:15:3C:84:00:B8:85:03:E6
        equifaxsecureebusinessca1, Jul 18, 2003, trustedCertEntry,
        Certificate fingerprint (MD5):
64:9C:EF:2E:44:FC:C6:8F:52:07:D0:51:73:8F:CB:3D
        thawteserverca, Dec 2, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
EE:FE:61:69:65:6E:F8:9C:C6:2A:F4:D7:2B:63:EF:A2
        utnuserfirstclientauthemailca, May 2, 2006, trustedCertEntry,
        Certificate fingerprint (MD5):
D7:34:3D:EF:1D:27:09:28:E1:31:02:5B:13:2B:DD:F7
        thawtepersonalfreemailca, Dec 2, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
53:4B:1D:17:58:58:1A:30:A1:90:F8:6E:5C:F2:CF:65
        utnuserfirsthardwareca, May 2, 2006, trustedCertEntry,
        Certificate fingerprint (MD5):
4C:56:41:E5:0D:BB:2B:E8:CA:A3:ED:18:08:AD:43:39
        entrustevca, Apr 28, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
D6:A5:C3:ED:5D:DD:3E:00:C1:3D:87:92:1F:1D:3F:E4
        certumca, Feb 10, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
2C:8F:9F:66:1D:18:90:B1:47:26:9D:8E:86:82:8C:A9
        addtrustclass1ca, May 2, 2006, trustedCertEntry,
        Certificate fingerprint (MD5):
1E:42:95:02:33:92:6B:B9:5F:C0:7F:DA:D6:B2:4B:FC
        entrustrootcag2, Jun 22, 2010, trustedCertEntry,
        Certificate fingerprint (MD5):
4B:E2:C9:91:96:65:0C:F4:0E:5A:93:92:A0:0A:FE:B2
        equifaxsecureca, Jul 18, 2003, trustedCertEntry,
        Certificate fingerprint (MD5):
67:CB:9D:C0:13:24:8A:82:9B:B2:17:1E:D1:1B:EC:D4
        quovadisrootca3, Jun 9, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
31:85:3C:62:94:97:63:B9:AA:FD:89:4E:AF:6F:E0:CF
        quovadisrootca2, Jun 9, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
5E:39:7B:DD:F8:BA:EC:82:E9:AC:62:BA:0C:54:00:2B
        digicerthighassuranceevrootca, Jan 7, 2008, trustedCertEntry,
        Certificate fingerprint (MD5):
D4:74:DE:57:5C:39:B2:D3:9C:85:83:C5:C0:65:49:8A
        secomvalicertclass1ca, May 1, 2008, trustedCertEntry,
        Certificate fingerprint (MD5):
65:58:AB:15:AD:57:6C:1E:A8:A7:B5:69:AC:BF:FF:EB
        equifaxsecureglobalebusinessca1, Jul 18, 2003, trustedCertEntry,
        Certificate fingerprint (MD5):
8F:5D:77:06:27:C4:98:3C:5B:93:78:E7:D7:7D:9B:CC
        geotrustuniversalca, Dec 3, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
92:65:58:8B:A2:1A:31:72:73:68:5C:B4:A5:7A:07:48
        thawteprimaryrootcag3, Nov 24, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
FB:1B:5D:43:8A:94:CD:44:C6:76:F2:43:4B:47:E7:31
        verisignclass3ca, Dec 2, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
```

```
EF:5A:F1:33:EF:F1:CD:BB:51:02:EE:12:14:4B:96:C4
        deutschetelekomrootca2, Nov 6, 2008, trustedCertEntry,
        Certificate fingerprint (MD5):
74:01:4A:91:B1:08:C4:58:CE:47:CD:F0:DD:11:53:08
        utnuserfirstobjectca, May 2, 2006, trustedCertEntry,
        Certificate fingerprint (MD5):
A7:F2:E4:16:06:41:11:50:30:6B:9C:E3:B4:9C:B0:C9
        geotrustprimaryca, Nov 24, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
02:26:C3:01:5E:08:30:37:43:A9:D0:7D:CF:37:E6:BF
        verisignclass1ca, Dec 2, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
86:AC:DE:2B:C5:6D:C3:D9:8C:28:88:D3:8D:16:13:1E
        baltimorecodesigningca, May 10, 2002, trustedCertEntry,
        Certificate fingerprint (MD5):
90:F5:28:49:56:D1:5D:2C:B0:53:D4:4B:EF:6F:90:22
        baltimorecybertrustca, May 10, 2002, trustedCertEntry,
        Certificate fingerprint (MD5):
AC:B6:94:A5:9C:17:E0:D7:91:52:9B:B1:97:06:A6:E4
        starfieldclass2ca, Jan 20, 2005, trustedCertEntry,
        Certificate fingerprint (MD5):
32:4A:4B:BB:C8:63:69:9B:BE:74:9A:C6:DD:1D:46:24
        camerfirmachamberscommerceca, Oct 10, 2008, trustedCertEntry,
        Certificate fingerprint (MD5):
B0:01:EE:14:D9:AF:29:18:94:76:8E:F1:69:33:2A:84
        ttelesecglobalrootclass3ca, Feb 10, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
CA:FB:40:A8:4E:39:92:8A:1D:FE:8E:2F:C4:27:EA:EF
        verisignclass3g5ca, Nov 24, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
CB:17:E4:31:67:3E:E2:09:FE:45:57:93:F3:0A:FA:1C
        trustcenteruniversalcai, Jan 7, 2008, trustedCertEntry,
        Certificate fingerprint (MD5):
45:E1:A5:72:C5:A9:36:64:40:9E:F5:E4:58:84:67:8C
        ttelesecglobalrootclass2ca, Feb 10, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
2B:9B:9E:E4:7B:6C:1F:00:72:1A:CC:C1:77:79:DF:6A
        verisignclass3g3ca, Mar 25, 2004, trustedCertEntry,
        Certificate fingerprint (MD5):
CD:68:B6:A7:C7:C4:CE:75:E0:1D:4F:57:44:61:92:09
        certumtrustednetworkca, Feb 10, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
D5:E9:81:40:C5:18:69:FC:46:2C:89:75:62:0F:AA:78
        certplusclass3pprimaryca, May 27, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
E1:4B:52:73:D7:1B:DB:93:30:E5:BD:E4:09:6E:BE:FB
        verisignclass3g2ca, Mar 25, 2004, trustedCertEntry,
        Certificate fingerprint (MD5):
A2:33:9B:4C:74:78:73:D4:6C:E7:C1:F3:8D:CB:5C:E9
        globalsignr3ca, Aug 17, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
C5:DF:B8:49:CA:05:13:55:EE:2D:BA:1A:C3:3E:B0:28
        utndatacorpsgcca, May 2, 2006, trustedCertEntry,
        Certificate fingerprint (MD5):
B3:A5:3E:77:21:6D:AC:4A:C0:C9:FB:D5:41:3D:CA:06
        secomscrootca2, Aug 17, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
6C:39:7D:A4:0E:55:59:B2:3F:D6:41:B1:12:50:DE:43
```

```
        secomscrootca1, May 1, 2008, trustedCertEntry,
        Certificate fingerprint (MD5):
F1:BC:63:6A:54:E0:B5:27:F5:CD:E7:1A:E3:4D:6E:4A
        gtecybertrustglobalca, May 10, 2002, trustedCertEntry,
        Certificate fingerprint (MD5):
CA:3D:D3:68:F1:03:5C:D0:32:FA:B8:2B:59:E8:5A:DB
        verisignuniversalrootca, Nov 24, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
8E:AD:B5:01:AA:4D:81:E4:8C:1D:D1:E1:14:00:95:19
        trustcenterclass4caii, Jan 7, 2008, trustedCertEntry,
        Certificate fingerprint (MD5):
9D:FB:F9:AC:ED:89:33:22:F4:28:48:83:25:23:5B:E0
        globalsignr2ca, Aug 1, 2007, trustedCertEntry,
        Certificate fingerprint (MD5):
94:14:77:7E:3E:5E:FD:8F:30:BD:41:B0:CF:E7:D0:30
        certplusclass2primaryca, May 27, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
88:2C:8C:52:B8:A2:3C:F3:F7:BB:03:EA:AE:AC:42:0B
        digicertglobalrootca, Jan 7, 2008, trustedCertEntry,
        Certificate fingerprint (MD5):
79:E4:A9:84:0D:7D:3A:96:D7:C0:4F:E2:43:4C:89:2E
        globalsignca, Mar 26, 2008, trustedCertEntry,
        Certificate fingerprint (MD5):
3E:45:52:15:09:51:92:E1:B7:5D:37:9F:B1:87:29:8A
        thawteprimaryrootca, Nov 24, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
8C:CA:DC:0B:22:CE:F5:BE:72:AC:41:1A:11:A8:D8:12
        geotrustglobalca, Jul 18, 2003, trustedCertEntry,
        Certificate fingerprint (MD5):
F7:75:AB:29:FB:51:4E:B7:77:5E:FF:05:3C:99:8E:F5
        soneraclass2ca, Mar 28, 2006, trustedCertEntry,
        Certificate fingerprint (MD5):
A3:EC:75:0F:2E:88:DF:FA:48:01:4E:0B:5C:48:6F:FB
        verisigntsaca, Aug 13, 2008, trustedCertEntry,
        Certificate fingerprint (MD5):
7F:66:7A:71:D3:EB:69:78:20:9A:51:14:9D:83:DA:20
        quovadisrootca, Jun 9, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
27:DE:36:FE:72:B7:00:03:00:9D:F4:F0:1E:6C:04:24
        soneraclass1ca, Mar 28, 2006, trustedCertEntry,
        Certificate fingerprint (MD5):
33:B7:84:F5:5F:27:D7:68:27:DE:14:DE:12:2A:ED:6F
        valicertclass2ca, Jan 20, 2005, trustedCertEntry,
        Certificate fingerprint (MD5):
A9:23:75:9B:BA:49:36:6E:31:C2:DB:F2:E7:66:BA:87
        comodoaaaca, May 2, 2006, trustedCertEntry,
        Certificate fingerprint (MD5):
49:79:04:B0:EB:87:19:AC:47:B0:BC:11:51:9B:74:D0
        aolrootca2, Mar 26, 2008, trustedCertEntry,
        Certificate fingerprint (MD5):
D6:ED:3C:CA:E2:66:0F:AF:10:43:0D:77:9B:04:09:BF
        keynectisrootca, Jun 8, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
CC:4D:AE:FB:30:6B:D8:38:FE:50:EB:86:61:4B:D2:26
        addtrustqualifiedca, May 2, 2006, trustedCertEntry,
        Certificate fingerprint (MD5):
27:EC:39:47:CD:DA:5A:AF:E2:9A:01:65:21:A9:4C:BB
        aolrootca1, Jan 17, 2008, trustedCertEntry,
```

```
        Certificate fingerprint (MD5):
14:F1:08:AD:9D:FA:64:E2:89:E7:1C:CF:A8:AD:7D:5E
        verisignclass2g3ca, Mar 25, 2004, trustedCertEntry,
        Certificate fingerprint (MD5):
F8:BE:C4:63:22:C9:A8:46:74:8B:B8:1D:1E:4A:2B:F6
        addtrustexternalca, May 2, 2006, trustedCertEntry,
        Certificate fingerprint (MD5):
1D:35:54:04:85:78:B0:3F:42:42:4D:BF:20:73:0A:3F
        verisignclass2g2ca, Mar 25, 2004, trustedCertEntry,
        Certificate fingerprint (MD5):
2D:BB:E5:25:D3:D1:65:82:3A:B7:0E:FA:E6:EB:E2:E1
        geotrustprimarycag3, Nov 24, 2009, trustedCertEntry,
        Certificate fingerprint (MD5):
B5:E8:34:36:C9:10:44:58:48:70:6D:2E:83:D4:B8:05
        swisssigngoldg2ca, Aug 13, 2008, trustedCertEntry,
        Certificate fingerprint (MD5):
24:77:D9:A8:91:D1:3B:FA:88:2D:C2:FF:F8:CD:33:93
        entrust2048ca, Jun 22, 2010, trustedCertEntry,
        Certificate fingerprint (MD5):
EE:29:31:BC:32:7E:9A:E6:E8:B5:F7:51:B4:34:71:90
        gtecybertrust5ca, May 10, 2002, trustedCertEntry,
        Certificate fingerprint (MD5):
7D:6C:86:E4:FC:4D:D1:0B:00:BA:22:BB:4E:7C:6A:8E
        camerfirmachambersignca, Oct 10, 2008, trustedCertEntry,
        Certificate fingerprint (MD5):
9E:80:FF:78:01:0C:2E:C1:36:BD:FE:96:90:6E:08:F3
        camerfirmachambersca, Oct 10, 2008, trustedCertEntry,
        Certificate fingerprint (MD5):
5E:80:9E:84:5A:0E:65:0B:17:02:F3:55:18:2A:3E:D7
        godaddyclass2ca, Jan 20, 2005, trustedCertEntry,
        Certificate fingerprint (MD5):
91:DE:06:25:AB:DA:FD:32:17:0C:BB:25:17:2A:84:67
        entrustsslca, Jan 9, 2003, trustedCertEntry,
        Certificate fingerprint (MD5):
DF:F2:80:73:CC:F1:E6:61:73:FC:F5:42:E9:C5:7C:EE
        verisignclass1g3ca, Mar 25, 2004, trustedCertEntry,
        Certificate fingerprint (MD5):
B1:47:BC:18:57:D1:18:A0:78:2D:EC:71:E8:2A:95:73
        secomevrootca1, May 1, 2008, trustedCertEntry,
        Certificate fingerprint (MD5):
22:2D:A6:01:EA:7C:0A:F7:F0:6C:56:43:3F:77:76:D3
        verisignclass1g2ca, Mar 25, 2004, trustedCertEntry,
        Certificate fingerprint (MD5):
DB:23:3D:F9:69:FA:4B:B9:95:80:44:73:5E:7D:41:83
        godaddysecurecertificationauthority, Jul 19, 2010, trustedCertEntry,
        Certificate fingerprint (MD5):
D5:DF:85:B7:9A:52:87:D1:8C:D5:0F:90:23:2D:B5:34
```

Amazon SNS 메시지의 서명 확인

또는 Amazon SNS가 보낸 알림의 신뢰성, 구독 확인 또는 구독 해지 확인 메시지를 확인할 수 있습니다. 사용자의 endpoint는 Amazon SNS 메시지에 담긴 정보를 사용해 서명할 스트링 및 서명을 재생성하므로

사용자는 메시지 콘텐츠로부터 재생성한 서명과 Amazon SNS가 메시지로 보낸 서명을 비교함으로써 메시지의 콘텐츠를 확인할 수 있습니다.

예를 들어 Amazon SNS 메시지를 처리하고 서명을 확인하는 Java servlet 코드는 Amazon SNS Endpoint Java Servlet에 대한 코드 예제 (p. 127)를 참조하십시오.

Amazon SNS 메시지의 서명 확인

1. Amazon SNS가 endpoint에 전송한 HTTP POST 요청 본문의 JSON 문서에서 이름/값 쌍을 추출합니다. 일부 이름/값 쌍의 값을 사용하여 서명할 스트링을 생성합니다. Amazon SNS 메시지의 서명을 확인할 때 escaped control 문자를 *`Message`* 및 *`Subject`* 값의 원본 문자로 변환하는 것은 무척 중요합니다. 이러한 값은 서명할 스트링의 일부로 사용 시 원본 형식이어야 합니다. JSON 문서를 분석하는 방법에 대한 자세한 내용은 단계 1: Amazon SNS 메시지를 처리하도록 endpoint를 준비합니다. (p. 105)를 참조하십시오.

   *`SignatureVersion`*는 서명 버전을 알려줍니다. 사용자는 서명 버전에서 서명을 생성하는 방법에 따르는 요구 사항을 결정할 수 있습니다. Amazon SNS 알림의 경우 Amazon SNS는 현재 서명 버전 1 및 서명 버전 4를 지원합니다. 이 섹션은 서명 버전 1을 사용하여 서명을 생성하는 단계를 제공합니다. 서명 버전 4에 대한 자세한 내용은Signature Version 4 Signing Process를 참조하십시오.
2. Amazon SNS가 메시지에 서명하는데 사용한 X509 인증서를 획득합니다. *`SigningCertURL`* 값은 메시지에 대한 전자 서명을 생성하는데 사용한 X509 인증서의 위치를 나타냅니다. 이 위치에서 인증서를 검색합니다.
3. 인증서로부터 공개 키를 추출합니다. *`SigningCertURL`*에 의해 지정된 인증서로부터의 공개 키는 메시지의 신뢰성 및 무결성을 확인하는데 사용합니다.
4. 메시지 유형을 결정합니다. 서명할 스트링 형식은 *`Type`* 값에 의해 지정된 메시지 유형에 따릅니다.
5. 서명할 스트링을 생성합니다. 서명할 스트링은 메시지에서 지정된 이름/값 쌍의 줄바꿈 문자로 구분지어진 목록입니다. 각각의 이름/값 쌍은 이름, 줄바꿈 문자, 값, 줄바꿈 문자 순으로 표기됩니다. 이름/값 쌍은 바이트, sort 순으로 나열됩니다.

   메시지 유형에 따라 서명할 스트링은 다음의 이름/값 쌍을 보유해야 합니다.

   Notification
   : 알림 메시지는 다음의 이름/값 쌍을 포함해야 합니다.

```
Message
MessageId
Subject (if included in the message)
Timestamp
TopicArn
Type
```

   다음의 예는 `Notification`에 대해 서명할 스트링입니다.

```
Message
My Test Message
MessageId
4d4dc071-ddbf-465d-bba8-08f81c89da64
Subject
My subject
Timestamp
2012-06-05T04:37:04.321Z
TopicArn
arn:aws:sns:us-east-1:123456789012:s4-MySNSTopic-1G1WEFCOXTC0P
Type
Notification
```

SubscriptionConfirmation 및 UnsubscribeConfirmation

`SubscriptionConfirmation` 및 `UnsubscribeConfirmation` 메시지는 다음의 이름/값 쌍을 포함해야 합니다.

```
Message
MessageId
SubscribeURL
Timestamp
Token
TopicArn
Type
```

다음의 예는 `SubscriptionConfirmation`에 대해 서명할 스트링입니다.

```
MessageId
3d891288-136d-417f-bc05-901c108273ee
SubscribeURL
https://sns.us-east-1.amazonaws.com/?Action=ConfirmSubscription&Topi
cArn=arn:aws:sns:us-east-1:123456789012:s4-MySNSTopic-1G1WEF
COXTC0P&Token=2336412f37fb687f5d51e6e241d09c8058323f60b964268bfe08ce35640228c208a66d3621bd9f7b012918cf
d
[illegible]
Timestamp
2012-06-03T19:25:13.719Z
Token
2336412f37fb687f5d51e6e241d09c8058323f60b964268bfe08ce35640228c208a66d3621bd9f7b012918cf
d
[illegible]
TopicArn
arn:aws:sns:us-east-1:123456789012:s4-MySNSTopic-1G1WEFCOXTC0P
Type
SubscriptionConfirmation
```

6. Base64 형식으로부터의 *`Signature`* 값을 디코딩합니다. 메시지는 Base64로 인코딩된 *`Signature`* 값의 서명을 전송합니다. 서명 값을 계산한 서명과 비교하기 전에 Base64 형식으로부터의 *`Signature`* 값을 디코딩하여 동일한 형식을 사용하여 값을 비교해야 합니다.
7. Amazon SNS 메시지의 파생 해시 값을 생성합니다. Amazon SNS 메시지를 서명을 생성하는데 사용한 동일한 해시 기능에 정규 형식으로 제출합니다.
8. Amazon SNS 메시지의 확인 해시 값을 생성합니다. 확인 해시 값은 공개 키 값(단계 3의) 사용하여 Amazon SNS 메시지로 전송된 서명 암호를 푼 결과입니다.
9. Amazon SNS 메시지의 신뢰성과 무결성을 확인합니다. 파생 해시 값(단계 7의)과 확인 해시 값(단계 8의)을 비교합니다. 값이 동일하다면 수신자는 메시지가 전송하는 동안 수정되지 않았으며 Amazon SNS에서 기원했음을 확신합니다. 값이 동일하지 않다면 수신자는 이를 신뢰하지 말아야 합니다.

Amazon SNS Endpoint Java Servlet에 대한 코드 예제

다음의 코드 조각은 Amazon SNS HTTP POST 요청을 처리하는 Java servlet 예제에서 기인합니다.

다음의 방법은 Java servlet에서 Amazon SNS로부터의 HTTP POST 요청 핸들러의 예를 구현합니다.

```
 protected void doPost(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException, SecurityException{
  //Get the message type header.
  String messagetype = request.getHeader("x-amz-sns-message-type");
  //If message doesn't have the message type header, don't process it.
  if (messagetype == null)
   return;

    // Parse the JSON message in the message body
    // and hydrate a Message object with its contents
    // so that we have easy access to the name/value pairs
    // from the JSON message.
    Scanner scan = new Scanner(request.getInputStream());
    StringBuilder builder = new StringBuilder();
    while (scan.hasNextLine()) {
      builder.append(scan.nextLine());
    }
  Message msg = readMessageFromJson(builder.toString());

    // The signature is based on SignatureVersion 1.
    // If the sig version is something other than 1,
    // throw an exception.
    if (msg.getSignatureVersion().equals("1")) {
    // Check the signature and throw an exception if the signature verification
 fails.
    if (isMessageSignatureValid(msg))
    log.info(">>Signature verification succeeded");
    else {
    log.info(">>Signature verification failed");
    throw new SecurityException("Signature verification failed.");
    }
    }
    else {
   log.info(">>Unexpected signature version. Unable to verify signature.");
      throw new SecurityException("Unexpected signature version. Unable to
verify signature.");
    }

    // Process the message based on type.
  if (messagetype.equals("Notification")) {
   //TODO: Do something with the Message and Subject.
   //Just log the subject (if it exists) and the message.
   String logMsgAndSubject = ">>Notification received from topic " +
msg.getTopicArn();
   if (msg.getSubject() != null)
    logMsgAndSubject += " Subject: " + msg.getSubject();
   logMsgAndSubject += " Message: " + msg.getMessage();
   log.info(logMsgAndSubject);
  }
    else if (messagetype.equals("SubscriptionConfirmation"))
  {
       //TODO: You should make sure that this subscription is from the topic
you expect. Compare topicARN to your list of topics
       //that you want to enable to add this endpoint as a subscription.

       //Confirm the subscription by going to the subscribeURL location
       //and capture the return value (XML message body as a string)
       Scanner sc = new Scanner(new URL(msg.getSubscribeURL()).openStream());
```

```
       StringBuilder sb = new StringBuilder();
       while (sc.hasNextLine()) {
         sb.append(sc.nextLine());
       }
       log.info(">>Subscription confirmation (" + msg.getSubscribeURL() +")
Return value: " + sb.toString());
       //TODO: Process the return value to ensure the endpoint is subscribed.
  }
    else if (messagetype.equals("UnsubscribeConfirmation")) {
      //TODO: Handle UnsubscribeConfirmation message.
      //For example, take action if unsubscribing should not have occurred.
      //You can read the SubscribeURL from this message and
      //re-subscribe the endpoint.
      log.info(">>Unsubscribe confirmation: " + msg.getMessage());
  }
    else {
      //TODO: Handle unknown message type.
      log.info(">>Unknown message type.");
    }
  log.info(">>Done processing message: " + msg.getMessageId());
 }
```

다음의 Java 방법 예제는 요청 본문에서 전송된 데이터를 담고 있는 `Message` 객체의 정보를 사용하여 서명을 생성하고 `Message` 객체에서도 읽히는 메시지의 Base64로 인코딩된 원본 서명 대비 서명을 확인합니다.

```
 private static boolean isMessageSignatureValid(Message msg) {
        try {
          URL url = new URL(msg.getSigningCertURL());
          InputStream inStream = url.openStream();
          CertificateFactory cf = CertificateFactory.getInstance("X.509");
          X509Certificate cert = (X509Certificate)cf.generateCertificate(in
Stream);
          inStream.close();

          Signature sig = Signature.getInstance("SHA1withRSA");
          sig.initVerify(cert.getPublicKey());
          sig.update(getMessageBytesToSign(msg));
          return sig.verify(Base64.decodeBase64(msg.getSignature()));
        }
        catch (Exception e) {
          throw new SecurityException("Verify method failed.", e);
        }
   }
```

다음의 Java 방법 예제는 Amazon SNS 메시지에 대해 서명할 스트링을 생성합니다. `getMessageBytesToSign` 방법은 메시지 유형에 기초하여 적절한 스트링-투-싸인 방법을 호출하고 서명할 스트링을 바이트 배열로 실행합니다. `buildNotificationStringToSign` 및 `buildSubscriptionStringToSign` 방법은 Amazon SNS 메시지의 서명 확인 (p. 125)에 설명된 형식에 기초하여 서명할 스트링을 생성합니다.

```
 private static byte [] getMessageBytesToSign (Message msg) {
  byte [] bytesToSign = null;
  if (msg.getType().equals("Notification"))
   bytesToSign = buildNotificationStringToSign(msg).getBytes();
```

```
  else if (msg.getType().equals("SubscriptionConfirmation") || msg.get
Type().equals("UnsubscribeConfirmation"))
   bytesToSign = buildSubscriptionStringToSign(msg).getBytes();
  return bytesToSign;
 }

  //Build the string to sign for Notification messages.
  public static String buildNotificationStringToSign( Message msg) {
    String stringToSign = null;

    //Build the string to sign from the values in the message.
    //Name and values separated by newline characters
    //The name value pairs are sorted by name
    //in byte sort order.
    stringToSign = "Message\n";
    stringToSign += msg.getMessage() + "\n";
    stringToSign += "MessageId\n";
    stringToSign += msg.getMessageId() + "\n";
    if (msg.getSubject() != null) {
      stringToSign += "Subject\n";
     stringToSign += msg.getSubject() + "\n";
    }
    stringToSign += "Timestamp\n";
    stringToSign += msg.getTimestamp() + "\n";
    stringToSign += "TopicArn\n";
    stringToSign += msg.getTopicArn() + "\n";
    stringToSign += "Type\n";
    stringToSign += msg.getType() + "\n";
    return stringToSign;
  }

  //Build the string to sign for SubscriptionConfirmation
  //and UnsubscribeConfirmation messages.
  public static String buildSubscriptionStringToSign(Message msg) {
   String stringToSign = null;
   //Build the string to sign from the values in the message.
   //Name and values separated by newline characters
   //The name value pairs are sorted by name
   //in byte sort order.
   stringToSign = "Message\n";
   stringToSign += msg.getMessage() + "\n";
   stringToSign += "MessageId\n";
   stringToSign += msg.getMessageId() + "\n";
   stringToSign += "SubscribeURL\n";
   stringToSign += msg.getSubscribeURL() + "\n";
   stringToSign += "Timestamp\n";
   stringToSign += msg.getTimestamp() + "\n";
   stringToSign += "Token\n";
   stringToSign += msg.getToken() + "\n";
   stringToSign += "TopicArn\n";
   stringToSign += msg.getTopicArn() + "\n";
   stringToSign += "Type\n";
   stringToSign += msg.getType() + "\n";
   return stringToSign;
  }
```

부록: 메시지 및 JSON 형식

Amazon SNS은 다음의 형식을 사용합니다.

Topics

HTTP/HTTPS 헤더

Amazon SNS가 구독확인, 알림 또는 구독 해지 확인 메시지를 HTTP/HTTPS endpoint로 전송할 때 Amazon SNS 지정 헤더 값과 함께 POST 메시지를 전송합니다. 이러한 헤더 값을 사용하면 `Type` 값을 읽기 위해 JSON 메시지 본문을 분석할 필요 없이 메시지 유형을 파악할 수 있습니다.

x-amz-sns-message-type
: 메시지의 유형입니다. 가능한 값은 *`SubscriptionConfirmation`*, *`Notification`* 및 *`UnsubscribeConfirmation`* 입니다.

x-amz-sns-message-id
: 범용 고유 식별자(Universally Unique Identifier)로 게시되는 각 메시지마다 고유합니다. 재시도 하는 동안 Amazon SNS가 재전송하는 알림의 경우 원본 메시지의 메시지 ID가 사용됩니다.

x-amz-sns-topic-arn
: 이 메시지가 게시된 주제에 대한 Amazon Resource Name(ARN)입니다.

x-amz-sns-subscription-arn
: 이 endpoint에 대한 구독의 ARN입니다.

다음의 HTTP POST 헤더는 HTTP endpoint에 대한 SubscriptionConfirmation 메시지 헤더의 예입니다.

```
POST / HTTP/1.1
x-amz-sns-message-type: SubscriptionConfirmation
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-east-1:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-east-1:123456789012:MyTopic:2bcfbf39-
05c3-41de-beaa-fcfcc21c8f55
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
```

HTTP/HTTPS 구독 확인 JSON 형식

사용자가 HTTP/HTTPS endpoint를 구독한 후에 Amazon SNS는 구독 확인 메시지를 HTTP/HTTPS endpoint에 전송합니다. 이 메시지는 구독을 확인하기 위해 방문해야 하는 *SubscribeURL* 값을 담고 있습니다(또는, *Token* 값을 ConfirmSubscription과 함께 사용할 수 있습니다). Amazon SNS는 구독이 확인되기 전에는 알림을 이 endpoint로 전송하지 않습니다.

구독 확인 메시지는 다음의 이름/값 쌍을 갖는 JSON 문서를 포함하는 메시지 본문으로 된 POST 메시지입니다.

Message
: 메시지를 설명하는 문자열입니다. 구독 확인의 경우 이 문자열은 다음과 같습니다.

```
You have chosen to subscribe to the topic arn:aws:sns:us-east-
1:123456789012:MyTopic.\nTo confirm the subscription, visit the SubscribeURL
 included in this message.
```

MessageId
: 범용 고유 식별자(Universally Unique Identifier)로 게시되는 각 메시지마다 고유합니다. 재시도 하는 동안 Amazon SNS가 재전송하는 메시지의 경우 원본 메시지의 메시지 ID가 사용됩니다.

Signature
: Message, MessageId, Type, Timestamp, TopicArn 값을 가진 Base64 인코딩 “SHA1withRSA” 서명입니다.

SignatureVersion
: 사용한 Amazon SNS 서명의 버전입니다.

SigningCertURL
: 메시지에 서명하기 위해 사용된 인증서의 URL입니다.

SubscribeURL
: 구독 확인을 위해 방문해야 하는 URL입니다. 또는, 그 대신 *Token*을 ConfirmSubscription 작업으로 사용하여 구독을 확인하면 됩니다.

Timestamp
: 구독 확인이 전송된 시간(GMT)입니다.

Token
: 구독 확인을 위해 ConfirmSubscription 작업에 사용할 수 있는 값입니다. 또는, 간단히 *SubscribeURL*을 방문하면 됩니다.

TopicArn
: 이 메시지가 구독된 주제에 대한 Amazon Resource Name(ARN)입니다.

Type
: 메시지의 유형입니다. 구독 확인의 경우 유형은 *SubscriptionConfirmation*입니다.

다음의 HTTP POST 메시지는 HTTP endpoint에 대한 SubscriptionConfirmation 메시지의 예입니다.

```
POST / HTTP/1.1
x-amz-sns-message-type: SubscriptionConfirmation
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-east-1:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-east-1:123456789012:MyTopic:2bcfbf39-
05c3-41de-beaa-fcfcc21c8f55
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
```

```
{
  "Type" : "SubscriptionConfirmation",
  "MessageId" : "165545c9-2a5c-472c-8df2-7ff2be2b3b1b",
  "Token" :
"23641 2f37fb87f5b1e6e241d09c85a6a57b30d712f794cc5f6a98866b9276bdb0a74fba6f3eb71854e25d6ad242b09eece2947f1f02b69c582af
bacc99c583a916b9981dd2728f4ae6fdb82efd087cc3b7849e05798d2d2785c03b0879594eeac82c01f235d0e717736",

  "TopicArn" : "arn:aws:sns:us-east-1:123456789012:MyTopic",
  "Message" : "You have chosen to subscribe to the topic arn:aws:sns:us-east-
1:123456789012:MyTopic.\nTo confirm the subscription, visit the SubscribeURL
included in this message.",
  "SubscribeURL" : "https://sns.us-east-1.amazonaws.com/?Action=ConfirmSubscrip
tion&TopicArn=arn:aws:sns:us-east-1:123456789012:MyTop
ic&Token=23641 2f37fb87f5b1e6e241d09c85a6a57b30d712f794cc5f6a98866b9276bdb0a74fba6f3eb71854e25d6ad242b09eece2947f1f02b69c582af
bacc99c583a916b9981dd2728f4ae6fdb82efd087cc3b7849e05798d2d2785c03b0879594eeac82c01f235d0e717736",

  "Timestamp" : "2012-04-26T20:45:04.751Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEpH+DcEwjAPg8O9mY8dReBSwksfg2S7WKQcikcNK
WLQjwu6A4VbeS0QHVCkhRS7fUQvi2egU3N858fiTDN6bkkOxYDVrY0Ad8L10Hs3zH81mtnPk5uvvol
IC1CXGu43obcgFxeL3khZl8IKvO61GWB6jI9b5+gLPoBc1Q=",
  "SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotificationSer
vice-f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}
```

구독 확인에 대한 JSON 형식의 정의를 확인하려면 https://sns.us-east-1.amazonaws.com/doc/2010-03-31/SubscriptionConfirmation.json에서 다음의 JSON 파일을 다운로드하십시오.

HTTP/HTTPS 알림 JSON 형식

Amazon SNS이 구독된 HTTP 또는 HTTPS endpoint에 알림을 전송할 때 endpoint에 전송된 POST 메시지는 다음의 이름/값 쌍으로 된 JSON 문서를 구성하는 메시지 본문을 보유합니다.

Message
: 알림이 주제에 게시되었을 때 지정되는 메시지 값입니다.

MessageId
: 범용 고유 식별자(Universally Unique Identifier)로 게시되는 각 메시지마다 고유합니다. 재시도 하는 동안 Amazon SNS가 재전송하는 알림의 경우 원본 메시지의 메시지 ID가 사용됩니다.

Signature
: Message, MessageId, Subject(있는 경우), Type, Timestamp, TopicArn 값을 가진 Base64 인코딩 "SHA1withRSA" 서명입니다.

SignatureVersion
: 사용한 Amazon SNS 서명의 버전입니다.

SigningCertURL
: 메시지에 서명하기 위해 사용된 인증서의 URL입니다.

Subject
: 알림이 주제에 게시되었을 때 제목 매개 변수이며, 옵션 매개 변수입니다. 제목이 지정되지 않을 경우에는 이 이름/값 쌍은 본 JSON 문서에 표시되지 않습니다.

Timestamp
: 알림이 게시된 시간(GMT)입니다.

TopicArn
: 이 메시지가 게시된 주제에 대한 Amazon Resource Name(ARN)입니다.

Type
: 메시지의 유형입니다. 알림의 경우 유형은 *`Notification`*입니다.

UnsubscribeURL
: 이 주제에서 endpoint를 구독 해지하는데 사용하는 URL입니다. 이 URL을 방문하면 Amazon SNS는 endpoint를 구독 해지하고 이 endpoint로 전송하는 알림을 중지합니다.

다음의 HTTP POST 메시지는 HTTP endpoint에 대한 Notification 메시지의 예입니다.

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324
x-amz-sns-topic-arn: arn:aws:sns:us-east-1:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-east-1:123456789012:MyTopic:c9135db0-
26c4-47ec-8998-413945fb5a96
Content-Length: 773
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324",
  "TopicArn" : "arn:aws:sns:us-east-1:123456789012:MyTopic",
  "Subject" : "My First Message",
  "Message" : "Hello world!",
  "Timestamp" : "2012-05-02T00:54:06.655Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEw6JRNwm1LFQL4ICB0bnXrdB8ClRMTQFGBqwLp
```

```
GbM78tJ4etTwC5zU7O3tS6tGpey3ejedNdOJ+1fkIp9F2/LmNVKb5aFlYq+9rk9ZiPph5YlLmWsD
cyC5T+Sy9/umic5S0UQc2PEtgdpVBahwNOdMW4JPwk0kAJJztnc=",
  "SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotificationSer
vice-f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&Sub
scriptionArn=arn:aws:sns:us-east-1:123456789012:MyTopic:c9135db0-26c4-47ec-8998-
413945fb5a96"
}
```

알림에 대한 JSON 형식의 정의를 확인하려면 https://sns.us-east-1.amazonaws.com/doc/2010-03-31/Notification.json에서 다음의 JSON 파일을 다운로드하십시오.

HTTP/HTTPS 구독 해지 확인 JSON 형식

주제에서 HTTP/HTTPS endpoint가 구독 해지된 후 Amazon SNS는 endpoint에 구독 해지 확인 메시지를 전송합니다.

구독 해지 확인 메시지는 다음의 이름/값 쌍을 갖는 JSON 문서를 포함하는 메시지 본문으로 된 POST 메시지입니다.

Message
: 메시지를 설명하는 문자열입니다. 구독 해지 확인의 경우 이 문자열은 다음과 같습니다.

```
You have chosen to deactivate subscription arn:aws:sns:us-east-
1:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55.\nTo cancel this
 operation and restore the subscription, visit the SubscribeURL included in
 this message.
```

MessageId
: 범용 고유 식별자(Universally Unique Identifier)로 게시되는 각 메시지마다 고유합니다. 재시도 하는 동안 Amazon SNS가 재전송하는 메시지의 경우 원본 메시지의 메시지 ID가 사용됩니다.

Signature
: Message, MessageId, Type, Timestamp, TopicArn 값을 가진 Base64 인코딩 “SHA1withRSA” 서명입니다.

SignatureVersion
: 사용한 Amazon SNS 서명의 버전입니다.

SigningCertURL
: 메시지에 서명하기 위해 사용된 인증서의 URL입니다.

SubscribeURL
: 구독 재확인을 위해 방문해야 하는 URL입니다. 또는, 그 대신 *Token*을 ConfirmSubscription 작업으로 사용하여 구독을 재확인하면 됩니다.

Timestamp
: 구독 해지 확인이 전송된 시간(GMT)입니다.

Token
: 구독 재확인을 위해 ConfirmSubscription 작업에 사용할 수 있는 값입니다. 또는, 간단히 *SubscribeURL*을 방문하면 됩니다.

TopicArn
: 이 endpoint가 구독 해지된 주제에 대한 Amazon Resource Name(ARN)입니다.

Type
: 메시지의 유형입니다. 구독 해지 확인의 경우 유형은 *UnsubscribeConfirmation*입니다.

다음의 HTTP POST 메시지는 HTTP endpoint에 대한 UnsubscribeConfirmation 메시지의 예입니다.

```
POST / HTTP/1.1
x-amz-sns-message-type: UnsubscribeConfirmation
x-amz-sns-message-id: 47138184-6831-46b8-8f7c-afc488602d7d
x-amz-sns-topic-arn: arn:aws:sns:us-east-1:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-east-1:123456789012:MyTopic:2bcfbf39-
05c3-41de-beaa-fcfcc21c8f55
Content-Length: 1399
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
```

```
{
  "Type" : "UnsubscribeConfirmation",
  "MessageId" : "47138184-6831-46b8-8f7c-afc488602d7d",
  "Token" : "2336412f37fb687f5d51e6e241d09c805a5a57b30d712f7948a98bac386ed
fe3e10314e873973b3e0a3c09119b722dedf2b5e31c59b13ed
bb26417c19f109351e6f2169efa9085ffe97e10535f4179ac1a0390b0f541f209c190f9ae23219ed6c470453e06c19b5ba9fcbb27daeb7c7",

  "TopicArn" : "arn:aws:sns:us-east-1:123456789012:MyTopic",
  "Message" : "You have chosen to deactivate subscription arn:aws:sns:us-east-
1:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55.\nTo cancel this
operation and restore the subscription, visit the SubscribeURL included in this
 message.",
  "SubscribeURL" : "https://sns.us-east-1.amazonaws.com/?Action=ConfirmSubscrip
tion&TopicArn=arn:aws:sns:us-east-1:123456789012:MyTop
ic&Token=2336412f37fb687f5d51e6e241d09c805a5a57b30d712f7948a98bac386ed
fe3e10314e873973b3e0a3c09119b722dedf2b5e31c59b13ed
bb26417c19f109351e6f2169efa9085ffe97e10535f4179ac1a0390b0f541f209c190f9ae23219ed6c470453e06c19b5ba9fcbb27daeb7c7",

  "Timestamp" : "2012-04-26T20:06:41.581Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEHXgJmXqnqsHTlqOCk7TIZsnk8zpJJoQbr8leD+8kAHcke3ClC4VPOvd
pZo9s/vR9GOznKab6sjGxE8uwqDI9HwpDm8lGxSlFGuwCruWeecnt7MdJCNh0XK4XQCbtGoXB762ePJ
faSWi9tYwzW65zAFU04WkNBkNsIf60=",
  "SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotificationSer
vice-f3ecfb7224c7233fe7bb5f59f96de52f.pem"
  }
```

구독 해지 확인에 대한 JSON 형식의 정의를 확인하려면 https://sns.us-east-1.amazonaws.com/doc/2010-03-31/UnsubscribeConfirmation.json에서 다음의 JSON 파일을 다운로드하십시오.

SetSubscriptionAttributes 전송 정책 JSON 형식

SetSubscriptionAttributes 작업에 요청을 전송하고 `DeliveryPolicy`의 값에 대해 AttributeName 매개 변수를 설정할 경우, AttributeValue 매개 변수의 값은 유효한 JSON 객체이어야 합니다. 예를 들어, 다음의 예는 최대 5회 재시도하도록 전송 정책을 설정합니다.

```
http://sns.us-east-1.amazonaws.com/
?Action=SetSubscriptionAttributes
&SubscriptionArn=arn%3Aaws%3Asns%3Aus-east-1%3A123456789012%3AMy-Top
ic%3A80289ba6-0fd4-4079-afb4-ce8c8260f0ca
&AttributeName=DeliveryPolicy
&AttributeValue={"healthyRetryPolicy":{"numRetries":5}}
...
```

AttributeValue 매개 변수의 값에 다음의 JSON 형식을 사용합니다.

```
{
  "healthyRetryPolicy" : {
    "minDelayTarget" :  <int>,
    "maxDelayTarget" : <int>,
    "numRetries" : <int>,
    "numMaxDelayRetries" : <int>,
    "backoffFunction" : "<linear|arithmetic|geometric|exponential>"
  },
  "throttlePolicy" : {
        "maxReceivesPerSecond" : <int>
  }
}
```

SetSubscriptionAttribute 작업에 대한 자세한 내용은 Amazon Simple Notification Service API Reference의 *SetSubscriptionAttributes*를 참조하십시오.

SetTopicAttributes 전송 정책 JSON 형식

SetTopicAttributes 작업에 요청을 전송하고 `DeliveryPolicy`의 값에 대해 AttributeName 매개 변수를 설정할 경우, AttributeValue 매개 변수의 값은 유효한 JSON 객체이어야 합니다. 예를 들어, 다음의 예는 최대 5회 재시도하도록 전송 정책을 설정합니다.

```
http://sns.us-east-1.amazonaws.com/
?Action=SetTopicAttributes
&TopicArn=arn%3Aaws%3Asns%3Aus-east-1%3A123456789012%3AMy-Topic
&AttributeName=DeliveryPolicy
&AttributeValue={"http":{"defaultHealthyRetryPolicy":{"numRetries":5}}}
...
```

AttributeValue 매개 변수의 값에 다음의 JSON 형식을 사용합니다.

```
{
  "http" : {
    "defaultHealthyRetryPolicy" : {
      "minDelayTarget":  <int>,
      "maxDelayTarget": <int>,
      "numRetries": <int>,
      "numMaxDelayRetries": <int>,
      "backoffFunction": "<linear|arithmetic|geometric|exponential>"
    },
    "disableSubscriptionOverrides" : <boolean>,
    "defaultThrottlePolicy" : {
      "maxReceivesPerSecond" : <int>
    }
  }
}
```

SetSubscriptionAttribute 작업에 대한 자세한 내용은 Amazon Simple Notification Service API Reference의 *SetTopicAttributes*를 참조하십시오.

부록: 라지 페이로드 및 원시 메시지 전송

이제 Amazon SNS와 Amazon SQS를 사용하여 최대 256KB(262,144바이트) 크기의 라지 페이로드 메시지를 전송할 수 있습니다. 라지 페이로드(64KB-256KB 크기의 메시지)를 전송하려면 AWS Signature Version 4(SigV4) signing을 지원하는 AWS SDK를 사용해야 합니다. SigV4가 AWS SDK를 지원하는지 여부를 검증하려면 SDK 릴리스 정보를 확인하십시오.

라지 페이로드 전송 외에도, Amazon SNS를 사용해 Amazon SQS endpoints 또는 HTTP/S endpoints에 전송된 메시지에 대한 원시 메시지 전송이 가능합니다. 이는 endpoint가 원시 메시지 전송이 선택되지 않을 경우 Amazon SNS 메타데이터에 대해 생성된 JSON 형식을 처리할 필요성을 제거합니다. 예를 들어 Amazon SQS endpoint에 대한 원시 메시지 전송을 활성화할 때, Amazon SNS 메타데이터는 포함되지 않으며 게시된 메시지는 구독된 Amazon SQS endpoint에 그대로 전송됩니다. HTTP/S endpoint에 대한 원시 메시지 전송을 활성화할 때, 메시지는 추가 HTTP 헤더 `x-amz-raw-message`(*`true`*의 값으로 된)를 포함하며 메시지가 JSON 형식이 아닌 원시로 게시될 것을 알릴 것입니다. 따라서 endpoint는 무엇이 전송되고 있는지 이해하며 구독은 JSON에서 원시 전송으로 보다 쉽게 전환될 수 있습니다.

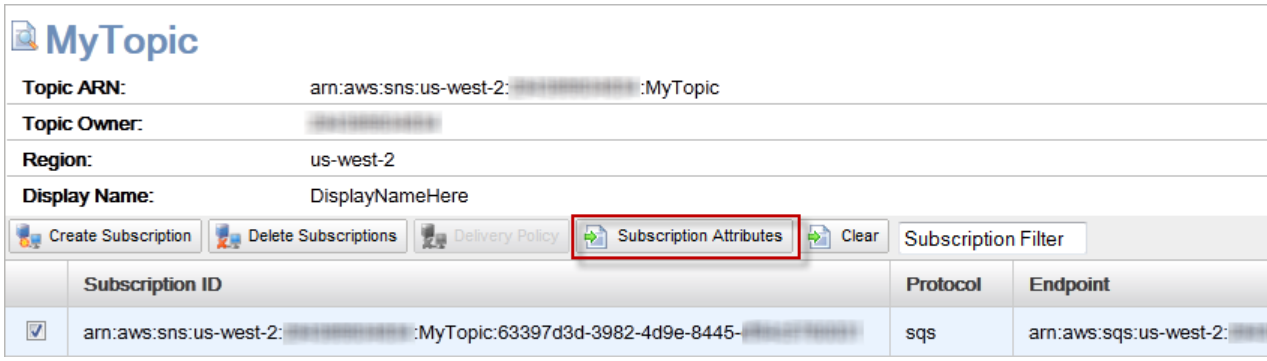
AWS SDK 중 하나를 사용하여 원시 메시지를 활성화하려면 `SetSubscriptionAttribute` 작업을 사용해야 하며 *`RawMessageDelivery`* 속성을 *`true`* 값으로 구성해야 합니다. 기본 값은 *`false`*입니다.

AWS Management Console을 사용한 원시 메시지 전송 활성화

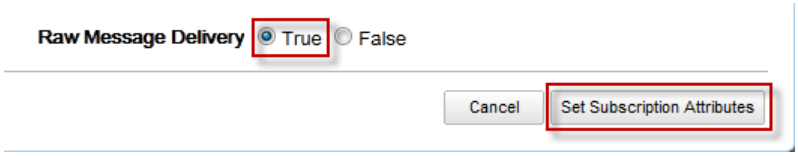
AWS Management Console을 사용하여 Raw Message Delivery 구독 속성을 true 값에 설정함으로써 원시 메시지 전송을 활성화할 수 있습니다.

AWS Management Console을 사용한 원시 메시지 전송 활성화

1. Sign in to the AWS Management Console and open the Amazon SNS console at https://console.aws.amazon.com/sns/.
2. Amazon SQS endpoint 또는 HTTP/S endpoint에 게시된 주제를 클릭합니다.
3. Subscription Attributes를 클릭합니다.



4. True를 선택하고 Set Subscription Attributes를 클릭합니다.



문서 기록

다음 표에서는 최신 *Amazon SNS 개발자 안내서* 릴리스의 중요한 변경 사항에 대해 설명합니다.

API 버전: 2010-03-31

마지막 문서 업데이트: 2013년 8월 13일

변경 사항	설명	날짜 변경됨
모바일 푸시 알림	알림 메시지를 모바일 디바이스의 앱에 직접 전송하도록 지원이 추가되었습니다. 자세한 내용은 Amazon SNS 모바일 푸시 알림 (p. 52)을 참조하십시오.	2013년 8월 13일
최초 릴리스	이 문서는 첫 번째 *Amazon SNS 개발자 안내서* 릴리스입니다.	2013년 5월 1일