Xamarin 개발자 안내서

AWS Mobile SDK

AWS Mobile SDK: Xamarin 개발자 안내서

Table of Contents

AWS Mobile SDK for Xamarin은 이제 AWS SDK for .NET에 포함됩니다. 이 안내서에서는 Xamarin용 모바일 SDK의 아카이브된 버전을 참조합니다.

.NET 및 Xamarin용 AWS Mobile SDK란 무엇인가요?

Xamarin용 AWS Mobile SDK는 이제 AWS SDK for .NET에 포함됩니다. 자세한 내용은 AWS SDK for .NET 개발자 안내서를 참조하세요.

이 안내서는 더 이상 업데이트되지 않으며 Xamarin용 Mobile SDK의 보관된 버전을 참조합니다.

관련 안내서 및 주제

- 프론트엔드 및 모바일 앱 개발의 경우 AWS Amplify를 사용하는 것이 좋습니다.
- Xamarin 앱에 AWS SDK for .NET를 사용할 때 특별히 고려해야 할 사항에 관한 내용은 AWS SDK for .NET 개발자 안내서의 Xamarin 지원에 대한 특별 고려 사항을 참조하세요.
- 참조용으로 GitHub에서 Xamarin용 AWS Mobile SDK의 아카이브된 버전을 찾을 수 있습니다.

아카이브된 참조 콘텐츠

.NET 및 Xamarin용 AWS Mobile SDK는 개발자가 다음 운영 체제를 위한 연결된 모바일 애플리케이션을 구축할 수 있도록 .NET 라이브러리, 코드 샘플 및 설명서 세트를 제공합니다.

- Xamarin iOS
- Xamarin Android
- Windows Phone Silverlight
- Windows RT 8.1
- Windows Phone 8.1

.NET 및 Xamarin용 AWS Mobile SDK로 작성된 모바일 앱은 네이티브 플랫폼 API를 호출하므로 모양과 느낌이 네이티브 애플리케이션과 비슷합니다. SDK의 .NET 라이브러리는 AWS REST API를 둘러싼 C# 래퍼를 제공합니다.

.NET 및 Xamarin용 AWS Mobile SDK에는 무엇이 포함되나요?

지원되는 AWS 서비스는 현재 다음과 같은 서비스를 포함하지만 이들로 국한되지는 않습니다.

- Amazon Cognito
- Amazon S3

- Amazon DynamoDB
- Amazon Mobile Analytics
- Amazon Simple Notification Service

이러한 서비스를 사용하여 사용자를 인증하고, 플레이어 및 게임 데이터를 저장하고, 클라우드에 객체를 저장하고, 푸시 알림을 수신하고, 사용량 데이터를 수집 및 분석할 수 있습니다.

또한 .NET 및 Xamarin용 AWS Mobile SDK를 사용하면 .NET용 AWS SDK에서 지원하는 대부분의 AWS 서비스를 사용할 수 있습니다. 이 개발자 안내서에서는 모바일 개발 전용 AWS 서비스에 대해 설명합니다. .NET용 AWS SDK에 대한 자세한 내용은 다음을 참조하세요.

- .NET용 AWS SDK 시작 안내서
- .NET용 AWS SDK 개발자 안내서
- .NET용 AWS SDK API 참조

호환성

.NET 및 Xamarin용 AWS Mobile SDK은 Portable Class Library(PCL)로 제공됩니다. Xamarin.Android 4.10.1 및 Xamarin.iOS 7.0.4에서 PCL 지원이 추가되었습니다. Portable Library 프로젝트는 Visual Studio에 내장되어 있습니다.

IDE

아카이브된 버전의 Xamarin SDK에서 IDE를 사용하는 방법에 대한 자세한 내용은 .NET and Xamarin용 AWS Mobile SDK 설정 섹션을 참조하세요.

.NET 및 Xamarin용 AWS Mobile SDK는 어떻게 받을 수 있나요?

.NET 및 Xamarin용 AWS Mobile SDK를 받는 방법은 .NET 및 Xamarin용 AWS Mobile SDK 설정을 참조하세요. .NET 및 Xamarin용 AWS Mobile SDK는 NuGet 패키지로 배포됩니다. AWS 서비스 패키지의 전체 목록은 NuGet의 AWS SDK 패키지 또는 .NET용 AWS SDK GitHub리포지토리에서 확인할 수 있습니다.

AWS Mobile 서비스

Amazon Cognito 자격 증명

모든 AWS 호출에는 AWS 자격 증명이 필요합니다. 앱에 자격 증명을 하드코딩하기보다는 Amazon Cognito 자격 증명을 사용하여 애플리케이션에 AWS 자격 증명을 제공할 것을 권장합니다. Amazon Cognito를 통해 AWS 자격 증명을 얻으려면 .NET 및 Xamarin용 AWS Mobile SDK 설정의 지침에 따르세요.

또한 Cognito는 Amazon, Facebook, Twitter, Google 같은 퍼블릭 로그인 공급자 그리고 OpenID Connect를 지원하는 공급자를 사용한 사용자 인증도 지원합니다. 또한 Cognito는 인증되지 않은 사용자도 지원합니다. Cognito는 자격 증명 및 액세스 관리(IAM) 역할을 통해 지정하는 제한적 액세스 권한을 갖는 임시 자격 증명을 제공합니다. Cognito는 IAM 역할과 연결된 자격 증명 풀을 생성하여 구성됩니다. 이 IAM 역할은 앱이 액세스할 수 있는 리소스/서비스를 지정합니다.

Cognito Identity를 시작하려면 .NET 및 Xamarin용 AWS Mobile SDK 설정을 참조하세요.

Cognito Identity에 대한 자세한 내용은 Amazon Cognito Identity를 참조하세요.

Amazon Cognito Sync

Cognito Sync는 애플리케이션 관련 사용자 데이터의 교차 디바이스 동기화를 활성화하는 클라이언트 라이브러리 및 AWS 서비스입니다. Cognito Sync API를 사용하여 여러 디바이스와 여러 로그인 공급자(Amazon, Facebook, Google 및 자체 사용자 지정 자격 증명 공급자) 사이에서 사용자 프로파일 데이터를 동기화할 수 있습니다.

Cognito Sync를 시작하려면 Cognito Sync를 사용하여 사용자 데이터 동기화를 참조하세요.

Cognito Sync에 대한 자세한 내용은 Amazon Cognito Sync를 참조하세요.

Mobile Analytics

Amazon Mobile Analytics를 사용하면 모바일 앱의 사용량을 수집하고, 시각화하고, 이해할 수 있습니다. 활성 사용자, 세션, 유지, 앱 내 수익 및 커스텀 이벤트 지표에 대한 보고서를 사용하고 플랫폼 및 날짜 범위를 기준으로 필터링할 수 있습니다. Amazon Mobile Analytics는 비즈니스 조건에 따라 확장하도록 설계되었으며, 수백만 명의 엔드포인트로부터 수십억 건의 이벤트를 수집하여 처리할 수 있습니다.

Mobile Analytics 사용을 시작하려면 Amazon Mobile Analytics를 사용하여 앱 사용량 데이터 추적을 참조하세요.

Mobile Analytics에 대한 자세한 내용은 Amazon Mobile Analytics를 참조하세요.

Dynamo DB

Amazon DynamoDB는 속도가 빠르고 확장성이 뛰어나며 비용 효과적인 비 관계형 데이터베이스 서비스입니다. DynamoDB는 기존 데이터 스토리지에서의 확장성 제한을 없애면서도 낮은 지연 시간과 예측 가능한 성능을 유지합니다.

Dynamo DB 사용을 시작하려면 DynamoDB를 사용하여 데이터 저장 및 검색을 참조하세요.

Dynamo DB에 대한 자세한 내용은 Amazon DynamoDB를 참조하세요.

Amazon Simple Notification Service

Amazon Simple Notification Service(SNS)는 빠르고 유연한 완전관리형 푸시 알림 서비스로서, 이 서비스를 사용하면 개별 메시지를 전송하거나 대규모의 수신자에게 메시지를 전송할 수 있습니다. Amazon Simple Notification Service를 사용하면 간편하고 비용 효과적으로 모바일 디바이스 사용자와 이메일 수신자에게 푸시 알림을 보내거나 다른 배포된 서비스에도 메시지를 보낼 수 있습니다.

Xamarin iOS용 SNS 사용을 사용하려면 SNS를 사용하여 푸시 알림 수신(Xamarin iOS)을 참조하세요.

Xamarin Android용 SNS 사용을 사용하려면 SNS를 사용하여 푸시 알림 수신(Xamarin Android)을 참조하세요.

SNS에 대한 자세한 내용은 Amazon Simple Notification Service(SNS)를 참조하세요.

.NET and Xamarin용 AWS Mobile SDK 설정

.NET 및 Xamarin용 AWS Mobile SDK를 설정한 후 새 프로젝트를 빌드하거나 SDK를 기존 프로젝트에 통합할 수 있습니다. 또한 샘플을 복제하고 실행하여 SDK가 어떻게 작동하는지 엿볼 수도 있습니다. 다음 단계에 따라 .NET 및 Xamarin용 AWS Mobile SDK를 설정하고 사용을 시작하세요.

필수 조건

.NET and Xamarin용 AWS Mobile SDK and Xamarin을 사용하려면 먼저 다음 작업을 수행해야 합니다.

- AWS 계정을 생성합니다.
- Xamarin을 설치합니다.

사전 조건을 모두 충족한 후:

1. Amazon Cognito를 사용하여 AWS 자격 증명을 받습니다.
2. 애플리케이션에서 사용할 각 AWS 서비스에 필요한 권한을 설정합니다.
3. IDE에서 새 프로젝트를 생성합니다.
4. .NET 및 Xamarin용 AWS Mobile SDK를 설치합니다.
5. .NET 및 Xamarin용 AWS Mobile SDK을 구성합니다.

1단계: AWS 자격 증명 얻기

애플리케이션에서 AWS를 호출하려면 먼저 AWS 자격 증명을 얻어야 합니다. 이렇게 하려면 애플리케이션에 프라이빗 AWS 자격 증명을 포함하지 않고도 애플리케이션이 SDK에서 서비스에 액세스할 수 있도록 하는 AWS 서비스인 Amazon Cognito를 사용합니다.

Amazon Cognito를 시작하려면 자격 증명 풀을 생성해야 합니다. 자격 증명 풀은 다음과 같은 고유한 자격 증명 풀 ID로 식별되는 계정에 고유한 정보의 스토어입니다.

```
"us-east-1:00000000-0000-0000-0000-000000000000"
```

1. Amazon Cognito 콘솔에 로그인하고 연동 자격 증명 관리와 새 자격 증명 풀 생성을 차례대로 선택합니다.

2. 자격 증명 풀의 이름을 입력하고 확인란을 선택하여 인증되지 않은 자격 증명에 대해 액세스를 활성화합니다. 풀 생성을 선택하여 자격 증명 풀을 생성합니다.
3. 허용을 선택하여 자격 증명 풀과 연결된 기본 역할 두 개(인증되지 않은 사용자의 역할 하나, 인증된 사용자의 역할 하나)를 생성합니다. 이 기본 역할은 Amazon Cognito Sync 및 Amazon Mobile Analytics에 대한 자격 증명 풀 액세스를 제공합니다.

일반적으로 애플리케이션당 하나의 자격 증명 풀만 사용합니다.

자격 증명 풀을 생성한 후 다음과 같이 `CognitoAWSCredentials` 객체를 생성(자격 증명 풀 ID를 전달)하고 AWS 클라이언트의 생성자에 전달하여 AWS 자격 증명을 얻습니다.

```
CognitoAWSCredentials credentials = new CognitoAWSCredentials (
    "us-east-1:00000000-0000-0000-0000-000000000000", // Your identity pool ID
    RegionEndpoint.USEast1 // Region
);

// Example for |MA|
analyticsManager = MobileAnalyticsManager.GetOrCreateInstance(
  credentials,
  RegionEndpoint.USEast1, // Region
  APP_ID // app id
);
```

2단계: 권한 설정

애플리케이션에서 사용하려는 모든 AWS 서비스에 필요한 권한을 설정해야 합니다. 먼저 어떻게 AWS가 애플리케이션의 사용자를 보는지 이해해야 합니다.

누군가 애플리케이션을 사용하고 AWS를 호출하면 AWS는 해당 사용자에게 자격 증명을 할당합니다. 1단계에서 생성한 자격 증명 풀이 AWS가 이러한 자격 증명을 저장하는 위치입니다. 자격 증명에는 두 가지 유형, 즉 인증된 자격 증명과 인증되지 않은 자격 증명이 있습니다. 인증된 자격 증명은 퍼블릭 로그인 공급자(예: Facebook, Amazon, Google)에 의해 인증된 사용자를 위한 것이고, 인증되지 않은 자격 증명은 게스트 사용자를 위한 것입니다.

모든 자격 증명은 AWS Identity and Access Management 역할과 연결되어 있습니다. 1단계에서 인증된 사용자와 인증되지 않은 사용자에 대해 각각 IAM 역할을 하나씩 총 두 개 생성했습니다. 모든 IAM 역할에는 해당 역할에 할당된 자격 증명이 액세스할 수 있는 AWS 서비스를 지정하는 정책이 하나 이상 연결되어 있습니다. 예를 들어 다음 샘플 정책은 Amazon S3 버킷에 대한 액세스 권한을 부여합니다.

```
{
  "Statement": [
    {
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:DeleteObject",
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::MYBUCKETNAME/*",
      "Principal": "*"
    }
  ]
}
```

애플리케이션에서 사용하려는 AWS 서비스에 대한 권한을 설정하려면 역할에 연결된 정책을 수정합니다.

1. IAM 콘솔로 이동하여 역할을 선택합니다. 검색 상자에 자격 증명 풀 이름을 입력합니다. 구성할 IAM 역할을 선택합니다. 애플리케이션이 인증된 사용자와 인증되지 않은 사용자를 모두 허용할 경우 두 역할 모두에 권한을 부여해야 합니다.
2. 정책 연결을 클릭하고 원하는 정책을 선택한 다음 정책 연결을 클릭합니다. 앞서 생성한 IAM 역할의 기본 정책은 Amazon Cognito Sync 및 Mobile Analytics에 대한 액세스 권한을 부여합니다.

정책 생성 또는 기존 정책 목록에서 정책 선택에 대한 자세한 내용은 IAM 정책을 참조하세요.

3단계: 새 프로젝트 생성

Windows

Visual Studio를 사용하여 애플리케이션을 개발할 수 있습니다.

OS X

Visual Studio를 사용하여 애플리케이션을 개발할 수 있습니다. Xamarin을 이용한 iOS 개발에서는 앱을 실행하기 위한 Mac 액세스 권한이 있어야 합니다. 자세한 내용은 Windows에 Xamarin.iOS 설치를 참조하세요.

> **Note**
>
> JetBrains의 크로스 플랫폼 상용 IDE Rider에는 Windows 및 Mac 플랫폼 모두에서 Xamarin이 지원됩니다.

4단계: .NET and Xamarin용 AWS Mobile SDK 설치

Windows

옵션 1: Package Manager 콘솔을 사용하여 설치

.NET 및 Xamarin용 AWS Mobile SDK는 일련의 .NET 어셈블리로 구성되어 있습니다. .NET 및 Xamarin용 AWS Mobile SDK를 설치하려면 Package Manager 콘솔에서 각 패키지에 대해 install-package 명령을 실행해야 합니다. 예를 들어 Cognito Identity를 설치하려면 다음을 실행합니다.

```
Install-Package AWSSDK.CognitoIdentity
```

AWS Core Runtime 및 Amazon Cognito Identity 패키지는 모든 프로젝트에 필요합니다. 다음은 각 서비스용 패키지 이름의 전체 목록입니다.

서비스	패키지 이름
AWS Core Runtime	AWSSDK.Core
Amazon Cognito Sync	AWSSDK.CognitoSync
Amazon Cognito 자격 증명	AWSSDK.CognitoIdentity
Amazon DynamoDB	AWSSDK.DynamoDBv2
Amazon Mobile Analytics	AWSSDK.MobileAnalytics
Amazon S3	AWSSDK.S3
Amazon SNS	AWSSDK.SimpleNotificationService

프리뷰 버전 패키지를 포함하려면 패키지를 설치하는 동안 다음과 같이 `-Pre` 명령줄 인수를 삽입합니다.

```
Install-Package AWSSDK.CognitoSync -Pre
```

AWS 서비스 패키지의 전체 목록은 NuGet의 AWS SDK 패키지 또는 .NET용 AWS SDK GitHub 리포지토리에서 확인할 수 있습니다.

옵션 2: IDE를 사용하여 설치

Visual Studio

1. 프로젝트를 마우스 오른쪽 버튼으로 클릭하고 Manage NuGet Packages(NuGet 패키지 관리)를 클릭합니다.
2. 프로젝트에 추가하려는 패키지 이름을 검색합니다. 프리뷰 버전 NuGet 패키지를 포함하려면 Include Prelease(프리뷰 포함)를 선택합니다. AWS 서비스 패키지의 전체 목록은 AWS SDK packages on NuGet에서 확인할 수 있습니다.
3. 패키지를 선택하고 Install(설치)을 선택합니다.

Mac(OS X)

Visual Studio

1. 패키지 폴더를 마우스 오른쪽 버튼으로 클릭하고 패키지 추가를 선택합니다.
2. 프로젝트에 추가하려는 패키지 이름을 검색합니다. 프리뷰 버전 NuGet 패키지를 포함하려면 Show pre-release packages(프리뷰 패키지 표시)를 선택합니다. AWS 서비스 패키지의 전체 목록은 AWS SDK packages on NuGet에서 확인할 수 있습니다.
3. 원하는 패키지 옆의 확인란을 선택하고 패키지 추가를 선택합니다.

> **Important**
>
> Portable Class Library를 사용하여 개발하는 경우 Portable Class Library에서 파생되는 모든 프로젝트에도 AWSSDK.Core NuGet 패키지를 추가해야 합니다.

5단계: .NET and Xamarin용 AWS Mobile SDK 구성

로깅 설정

`Amazon.AWSConfigs` 클래스 및 `Amazon.Util.LoggingConfig` 클래스를 사용하여 로깅 설정을 지정합니다. 이러한 클래스는 Visual Studio에서 Nuget Package Manager를 통해 사용할 수 있는 `AWSSdk.Core` 어셈블리에 포함되어 있습니다. 로깅 설정 코드는 `OnCreate` 파일(Android 앱의 경우) 또는 `MainActivity.cs` 파일(iOS 앱의 경우)의 `AppDelegate.cs` 메서드에 배치할 수 있습니다. `using Amazon` 및 `using Amazon.Util` 문도 .cs 파일에 추가할 수 있습니다.

다음과 같이 로깅 설정을 구성합니다.

```
var loggingConfig = AWSConfigs.LoggingConfig;
loggingConfig.LogMetrics = true;
loggingConfig.LogResponses = ResponseLoggingOption.Always;
loggingConfig.LogMetricsFormat = LogMetricsFormatOption.JSON;
loggingConfig.LogTo = LoggingOptions.SystemDiagnostics;
```

SystemDiagnostics에 로깅할 때 프레임워크가 내부적으로 System.Console에 출력합니다. HTTP 응답을 로깅하려면 `LogResponses` 플래그를 설정합니다. 값은 Always, Never 또는 OnError가 될 수 있습니다.

또한 `LogMetrics` 속성을 사용하여 HTTP 요청에 대한 성능 지표를 로깅할 수도 있습니다. 로그 형식은 `LogMetricsFormat` 속성을 사용하여 지정할 수 있습니다. 유효한 값은 JSON 또는 표준입니다.

리전 엔드포인트 설정

다음과 같이 모든 서비스 클라이언트의 기본 리전을 구성합니다.

```
AWSConfigs.AWSRegion="us-east-1";
```

그러면 SDK 내 모든 서비스 클라이언트에 대해 기본 리전이 설정됩니다. 다음과 같이 서비스 클라이언트의 인스턴스를 생성할 때 명시적으로 리전을 지정하여 이 설정을 재정의할 수 있습니다.

```
IAmazonS3 s3Client = new AmazonS3Client(credentials,RegionEndpoint.USEast1);
```

HTTP 프록시 설정 구성

네트워크가 프록시를 사용하는 경우 다음과 같이 HTTP 요청에 대한 프록시 설정을 구성할 수 있습니다.

```
var proxyConfig = AWSConfigs.ProxyConfig;
proxyConfig.Host = "localhost";
proxyConfig.Port = 80;
proxyConfig.Username = "<username>";
proxyConfig.Password = "<password>";
```

클럭 스큐 보상

이 속성은 SDK가 정확한 서버 시간을 측정하고 올바른 시간으로 요청을 재발행하여 클라이언트 클럭 스큐를 보상해야 하는지 여부를 결정합니다.

```
AWSConfigs.CorrectForClockSkew = true;
```

서비스 호출에서 예외가 발생하고 SDK가 로컬 시간과 서버 시간 간 차이를 확인할 경우 이 필드가 설정됩니다.

```
var offset = AWSConfigs.ClockOffset;
```

클럭 스큐에 대한 자세한 내용은 AWS 블로그에서 Clock-skew Correction을 참조하세요.

다음 단계

.NET and Xamarin용 AWS Mobile SDK 설정 완료 후 다음을 수행할 수 있습니다.

- 시작하기. .NET용 AWS Mobile SDK 및 Xamarin에서 서비스를 사용하고 구성하는 방법에 대한 빠른 시작 지침은 .NET용 AWS Mobile SDK 및 Xamarin .NET용 AWS Mobile SDK 시작하기를 참조하세요.
- 서비스 주제 살펴보기. 각 서비스에 대해 알아보고 .NET and Xamarin용 AWS Mobile SDK and Xamarin에서 어떻게 작동하는지 알아보세요.
- 데모 실행. 일반 사용 사례를 보여 주는 샘플 Xamarin 애플리케이션을 봅니다. 샘플 앱을 실행하려면 앞서 설명한 대로 .NET 및 Xamarin용 AWS Mobile SDK를 설정하고 개별 샘플의 README 파일에 수록된 지침을 따릅니다.
- API 알아보기. |sdk-xamarin-ref|_를 봅니다.
- 질문하기: AWS Mobile SDK Forums에 질문을 올리거나 GitHub에 문제를 개설합니다.

.NET 및 Xamarin용 AWS Mobile SDK 시작하기

.NET 및 Xamarin용 AWS Mobile SDK는 Xamarin 애플리케이션에서 AWS 서비스를 호출하는 데 필요한 라이브러리, 샘플 및 설명서를 제공합니다.

아래 서비스를 사용하려면 .NET 및 Xamarin용 AWS Mobile SDK 설정의 지침을 모두 완료해야 합니다.

시작하기 주제에서는 다음 항목을 살펴봅니다.

주제

- Amazon S3를 사용한 파일 저장 및 검색
- Cognito 동기화로 사용자 데이터 동기화
- DynamoDB를 사용한 데이터 저장 및 검색
- Amazon Mobile Analytics를 통한 앱 사용 데이터 추적
- SNS를 사용하여 푸시 알림 수신(Xamarin iOS)
- SNS를 사용하여 푸시 알림 수신(Xamarin Android)

다른 AWS Mobile SDK에 대한 자세한 내용은 AWS Mobile SDK를 참조하세요.

Amazon S3를 사용한 파일 저장 및 검색

Amazon Simple Storage Service(Amazon S3)는 모바일 개발자에 안전하고 내구성과 확장성이 뛰어난 객체 스토리지를 제공합니다. Amazon S3는 간단한 웹 서비스 인터페이스를 통해 웹 어디서나 원하는 양의 데이터를 저장 및 검색할 수 있으므로 사용하기가 쉽습니다.

다음 자습서에서는 S3를 사용하기 위한 상위 수준 유틸리티인 S3 TransferUtility를 앱에 통합하는 방법을 설명합니다. Xamarin 애플리케이션에서 S3을 사용하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service(S3)를 참조하세요.

프로젝트 설정

필수 조건

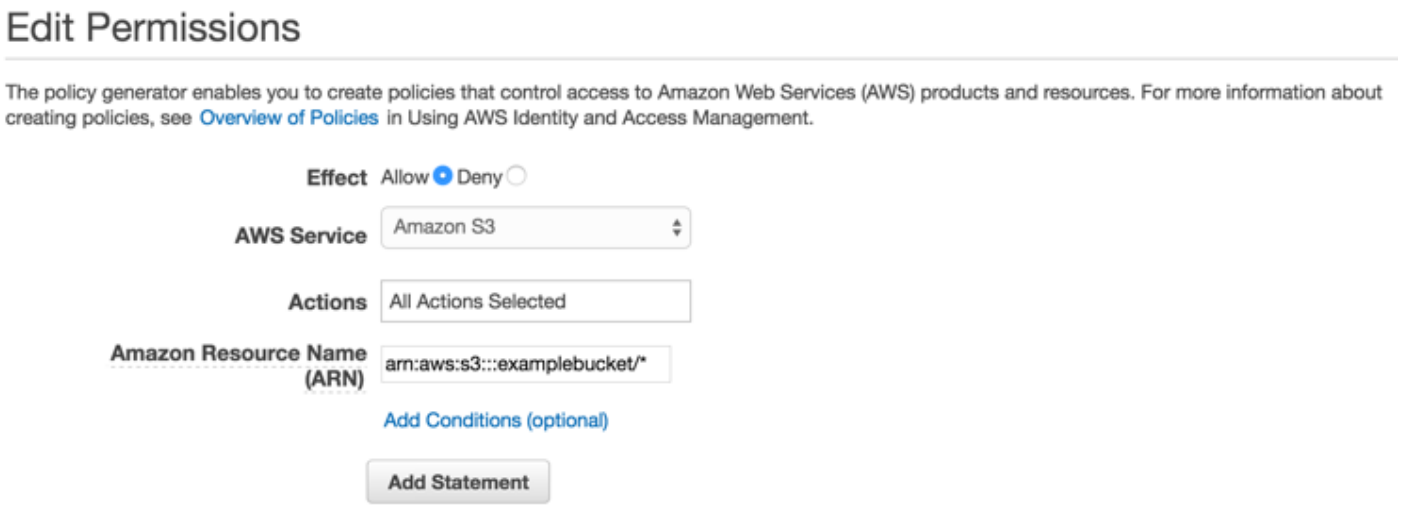
이 자습서를 시작하기 전에 .NET 및 Xamarin용 AWS Mobile SDK 설정의 지침을 모두 완료해야 합니다.

또한 이 자습서에서는 이미 S3 버킷을 생성한 것으로 가정합니다. S3 버킷을 생성하려면 S3 AWS 콘솔로 이동하세요.

S3에 대한 권한 설정

기본 IAM 역할 정책은 애플리케이션에 Amazon Mobile Analytics 및 Amazon Cognito Sync에 대한 액세스 권한을 부여합니다. Cognito 자격 증명 풀이 Amazon S3에 액세스할 수 있으려면 자격 증명 풀의 역할을 수정해야 합니다.

1. Identity and Access Management 콘솔로 이동하여 왼쪽 창에서 역할을 클릭합니다.
2. 검색 상자에 자격 증명 풀 이름을 입력합니다. 인증된 사용자와 인증되지 않은 사용자에 대해 하나씩 2개의 역할이 나열됩니다.
3. 인증되지 않은 사용자의 역할을 클릭합니다(자격 증명 풀 이름에 unauth가 추가됨).
4. Create Role Policy(역할 정책 생성)를 클릭하고 정책 생성기를 선택한 다음 선택을 클릭합니다.
5. Edit Permissions(권한 편집) 페이지에서 다음 이미지에 표시된 설정을 입력합니다. Amazon 리소스 이름(ARN)은 사용자의 것으로 바꿔야 합니다. S3 버킷의 ARN은 `arn:aws:s3:::examplebucket/*`과 비슷하며, 버킷이 위치하는 리전과 버킷의 이름으로 구성됩니다. 아래 표시된 설정은 자격 증명 풀에 지정된 버킷의 모든 작업에 대한 전체 액세스를 부여합니다.



1. 설명문 추가 버튼을 클릭한 다음 다음 단계를 클릭합니다.
2. 마법사가 앞서 생성한 구성을 표시합니다. 정책 적용을 클릭합니다.

S3 액세스 권한 부여에 대한 자세한 내용은 Amazon S3 버킷에 대한 액세스 권한 부여를 참조하세요.

프로젝트에 S3용 NuGet 패키지 추가

.NET 및 Xamarin용 AWS Mobile SDK 설정 내 지침의 4단계를 따라 S3 NuGet 패키지를 프로젝트에 추가합니다.

(선택 사항) S3 요청의 서명 버전을 구성합니다.

Amazon S3와의 모든 상호 작용은 인증을 거치거나 익명으로 할 수 있습니다. AWS는 서명 버전 4 또는 서명 버전 2 알고리즘을 사용해 서비스 호출을 인증합니다.

2014년 1월 이후 생성된 모든 새 AWS 리전은 서명 버전 4만 지원합니다. 하지만 그 이전의 리전은 계속해서 서명 버전 4 및 서명 버전 2 요청을 지원합니다.

버킷이 이 페이지에 나열된 서명 버전 2 요청을 지원하지 않는 리전에 위치하는 경우 다음과 같이 AWSConfigsS3.UseSignatureVersion4 속성을 "true"로 설정해야 합니다.

```
AWSConfigsS3.UseSignatureVersion4 = true;
```

AWS 서명 버전에 대한 자세한 내용은 요청 인증(AWS 서명 버전 4)을 참조하세요.

S3 TransferUtility 클라이언트 초기화

다음과 같이 S3 클라이언트를 생성하여 AWS 자격 증명 객체를 전달한 다음 S3 클라이언트를 TransferUtility로 전달합니다.

```
var s3Client = new AmazonS3Client(credentials,region);
var transferUtility = new TransferUtility(s3Client);
```

Amazon S3로 파일 업로드

S3로 파일을 업로드하려면 Transfer Utility 객체에서 `Upload`를 호출하여 다음 파라미터를 전달합니다.

- `file` - 업로드할 파일의 문자열 이름
- `bucketName` - 파일을 저장할 S3 버킷의 문자열 이름

```
transferUtility.Upload(
  Path.Combine(Environment.SpecialFolder.ApplicationData,"file"),
  "bucketName"
```

```
);
```

위 코드는 Environment.SpecialFolder.ApplicationData 디렉터리에 파일이 있다고 가정합니다. 업로드는 처리량을 높이기 위해 대용량 파일에서 자동으로 S3의 멀티파트 업로드 기능을 사용합니다.

Amazon S3에서 파일 다운로드

S3에서 파일을 다운로드하려면 Transfer Utility 객체에서 `Download`를 호출하여 다음 파라미터를 전달합니다.

- `file` - 다운로드할 파일의 문자열 이름
- `bucketName` - 다운로드할 파일이 저장된 S3 버킷의 문자열 이름
- `key` - 다운로드할 S3 객체(이 경우에는 파일)의 이름을 표시하는 문자열

```
transferUtility.Download(
  Path.Combine(Environment.SpecialFolder.ApplicationData,"file"),
  "bucketName",
  "key"
);
```

Xamarin 애플리케이션에서 Amazon S3에 액세스하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service(S3)를 참조하세요.

Cognito 동기화로 사용자 데이터 동기화

Amazon Cognito Sync를 사용하면 백엔드 코드를 작성하거나 인프라를 관리하지 않고도 앱 기본 설정 또는 게임 상태 같은 모바일 사용자 데이터를 손쉽게 AWS 클라우드에 저장할 수 있습니다. 사용자의 디바이스에 로컬로 데이터를 저장하여 디바이스가 오프라인 상태에서도 애플리케이션이 작동하도록 할 수 있습니다. 이러한 데이터를 사용자 디바이스 간 동기화할 수도 있으므로 사용하는 디바이스에 관계없이 일관된 앱 환경을 제공합니다.

자습에서는 앱에 Sync를 통합하는 방법을 설명합니다.

프로젝트 설정

필수 조건

이 자습서를 시작하기 전에 .NET 및 Xamarin용 AWS Mobile SDK 설정의 지침을 모두 완료해야 합니다.

Cognito Sync 리소스에 대한 액세스 권한 부여

설정 시 생성한 인증되지 않은 역할 및 인증된 역할에 연결된 기본 정책은 애플리케이션에 Cognito Sync에 대한 액세스 권한을 부여합니다. 추가 구성이 필요하지 않습니다.

프로젝트에 Cognito Sync용 NuGet 패키지 추가

.NET 및 Xamarin용 AWS Mobile SDK 설정 내 지침의 4단계를 따라 Cognito SyncManager NuGet 패키지를 프로젝트에 추가합니다.

CognitoSyncManager를 초기화합니다.

초기화된 Amazon Cognito 자격 증명 공급자를 `CognitoSyncManager` 생성자로 전달합니다.

```
CognitoSyncManager syncManager = new CognitoSyncManager (
    credentials,
    new AmazonCognitoSyncConfig {
        RegionEndpoint = RegionEndpoint.USEast1 // Region
    }
);
```

사용자 데이터 동기화

인증되지 않은 사용자 데이터를 동기화하는 방법:

1. 데이터 세트를 생성합니다.
2. 데이터 세트에 사용자 데이터를 추가합니다.
3. 데이터 세트를 클라우드와 동기화합니다.

데이터 세트 생성

`Dataset`의 인스턴스를 만듭니다. 새 데이터 세트를 생성하거나 디바이스에서 로컬로 저장된 데이터 세트의 기존 인스턴스를 여는 데 openOrCreateDataset 메서드가 사용됩니다.

```
Dataset dataset = syncManager.OpenOrCreateDataset("myDataset");
```

데이터 세트에 사용자 데이터 추가

사용자 데이터는 키/값 페어 형식으로 추가됩니다.

```
dataset.OnSyncSuccess += SyncSuccessCallback;
dataset.Put("myKey", "myValue");
```

Cognito 데이터 세트는 키를 통해 값에 액세스할 수 있는 사전으로 작동합니다.

```
string myValue = dataset.Get("myKey");
```

데이터 세트 동기화

데이터 세트를 동기화하려면 해당 synchronize 메서드를 호출합니다.

```
dataset.SynchronizeAsync();

void SyncSuccessCallback(object sender, SyncSuccessEventArgs e) {
  // Your handler code here
}
```

데이터 세트가 동기화될 때까지 데이터 세트에 기록된 모든 데이터가 로컬에 저장됩니다. 이 섹션의 코드는 인증되지 않은 Cognito 자격 증명을 사용하는 것으로 가정합니다. 따라서 사용자 데이터는 클라우드와 동기화되면 디바이스별로 저장됩니다. 디바이스에는 디바이스 ID가 연결되어 있습니다. 사용자 데이터가 클라우드와 동기화되면 해당 디바이스 ID와 연결됩니다.

Cognito Sync에 대한 자세한 내용은 Amazon Cognito Sync를 참조하세요.

DynamoDB를 사용한 데이터 저장 및 검색

Amazon DynamoDB는 속도가 빠르고 확장성이 뛰어나며 비용 효과적인 비 관계형 데이터베이스 서비스입니다. DynamoDB는 기존 데이터 스토리지에서의 확장성 제한을 없애면서도 낮은 지연 시간과 예측 가능한 성능을 유지합니다.

다음 자습서에서는 DynamoDB에 객체를 저장하는 앱에 DynamoDB 객체 지속성 모델을 통합하는 방법을 설명합니다.

프로젝트 설정

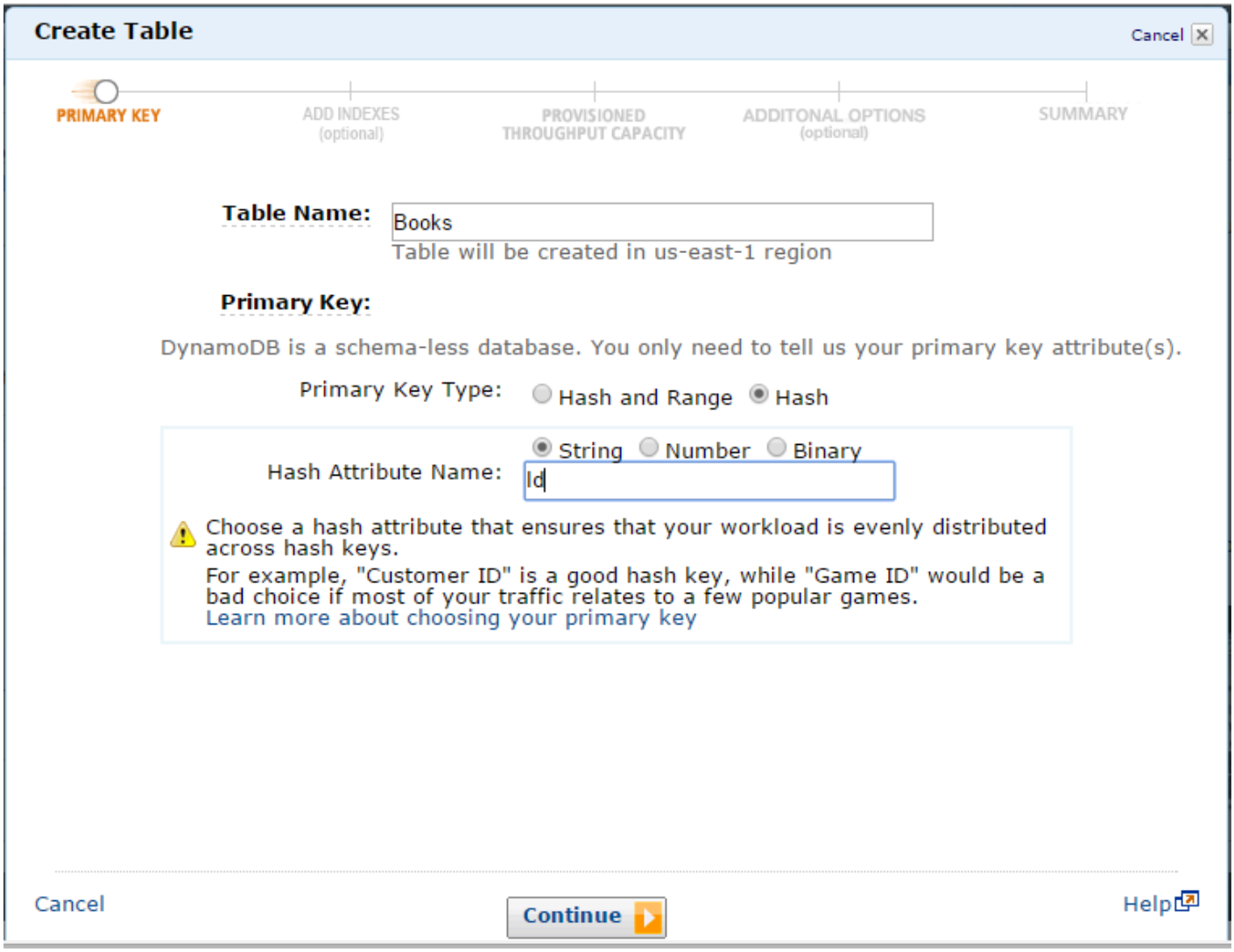
필수 조건

이 자습서를 시작하기 전에 .NET 및 Xamarin용 AWS Mobile SDK 설정의 지침을 모두 완료해야 합니다.

DynamoDB 테이블 생성

DynamoDB 데이터베이스에서 데이터를 읽고 쓰려면 먼저 테이블을 생성해야 합니다. 테이블을 생성할 때 기본 키를 지정해야 합니다. 기본 키는 해시 속성과 범위 속성(선택 사항) 두 가지로 구성됩니다. 기본 및 범위 속성의 사용 방식에 대한 자세한 내용은 테이블 작업을 참조하세요.

1. DynamoDB 콘솔로 이동하여 테이블 생성을 클릭합니다. [Create Table] 마법사가 표시됩니다.
2. 아래와 같이 테이블 이름, 기본 키 유형(해시) 및 해시 속성 이름("ID")을 지정하고 계속을 클릭합니다.



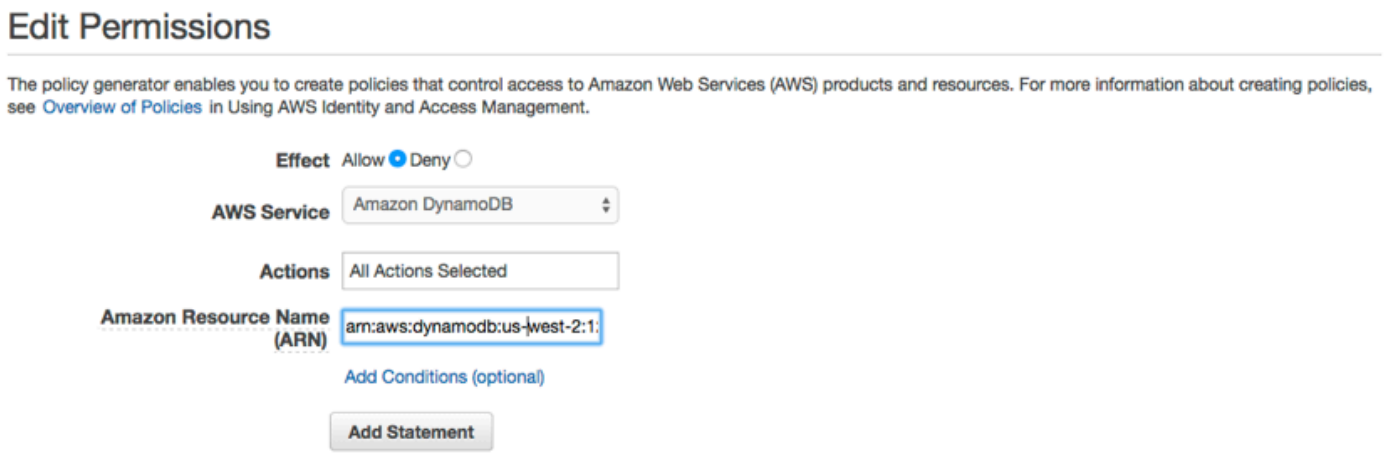
3. 다음 화면의 편집 필드는 비워 둔 상태에서 계속을 클릭합니다.
4. 읽기 용량 단위 및 쓰기 용량 단위의 기본값을 수락하고 계속을 클릭합니다.
5. 다음 화면의 알림 보내기: 텍스트 상자에 이메일 주소를 입력하고 계속을 클릭합니다. 검토 화면이 표시됩니다.

6. 생성을 클릭합니다. 테이블이 생성되는 데 몇 분 정도 걸릴 수 있습니다.

DynamoDB에 대한 권한 설정

자격 증명 풀이 Amazon DynamoDB에 액세스할 수 있으려면 자격 증명 풀의 역할을 수정해야 합니다.

1. Identity and Access Management 콘솔로 이동하여 왼쪽 창에서 역할을 클릭합니다. 자격 증명 풀 이름을 검색합니다(자격 증명당 인증된 사용자 및 인증되지 않은 사용자에 대해 하나씩 2개의 역할이 생성됨).
2. 인증되지 않은 사용자의 역할을 클릭하고(자격 증명 풀 이름에 "unauth"가 추가됨) 역할 정책 생성을 클릭합니다.
3. 정책 생성기를 클릭하고 선택을 클릭합니다.
4. 권한 편집 페이지에서 다음 그림과 같이 설정을 입력합니다. DynamoDB 테이블의 Amazon 리소스 이름(ARN)은 `arn:aws:dynamodb:us-west-2:123456789012:table/Books`와 비슷하며 테이블이 위치한 리전, 소유자의 AWS 계정 번호 및 `table/Books` 형식의 테이블 이름으로 구성됩니다. ARN 지정에 대한 자세한 내용은 DynamoDB의 Amazon 리소스 이름을 참조하세요.



5. 설명문 추가를 클릭하고 다음 단계를 클릭합니다. 마법사가 앞서 생성한 구성을 표시합니다.
6. Apply Policy(정책 적용)를 클릭합니다.

프로젝트에 DynamoDB용 NuGet 패키지 추가

.NET 및 Xamarin용 AWS Mobile SDK 설정 내 지침의 4단계를 따라 DynamoDB NuGet 패키지를 프로젝트에 추가합니다.

AmazonDynamoDBClient 초기화

초기화된 Amazon Cognito 자격 증명 공급자 및 리전을 AmazonDynamoDB 생성자로 전달한 다음 클라이언트를 DynamoDBContext로 전달합니다.

```
var client = new AmazonDynamoDBClient(credentials,region);
DynamoDBContext context = new DynamoDBContext(client);
```

클래스 생성

테이블에 행을 기록하려면 행 데이터를 보관할 클래스를 정의합니다. 클래스는 행의 속성 데이터를 보관하는 속성도 포함해야 하며, 콘솔에서 생성된 DynamoDB 테이블에 매핑됩니다. 다음 클래스 선언은 그러한 클래스를 예시합니다.

```
[DynamoDBTable("Books")]
public class Book
{
    [DynamoDBHashKey]    // Hash key.
    public int Id { get; set; }
    public string Title { get; set; }
    public string ISBN { get; set; }
    public int Price { get; set; }
    public string PageCount { get; set; }
    public string Author{ get; set; }
}
```

항목 저장

항목을 저장하려면 먼저 객체를 생성합니다.

```
Book songOfIceAndFire = new Book()
{
    Id=1,
    Title="Game Of Thrones",
    ISBN="978-0553593716",
    Price=4,
    PageCount="819",
    Author="GRRM"
};
```

그런 다음 저장합니다.

```
context.Save(songOfIceAndFire);
```

행을 업데이트하려면 위와 같이 DDTableRow 클래스의 인스턴스를 수정하고 AWSDynamoObjectMapper.save()를 호출합니다.

항목 검색

기본 키를 사용하여 항목을 검색합니다.

```
Book retrievedBook = context.Load<Book>(1);
```

항목 업데이트

항목 업데이트 방법:

```
Book retrievedBook = context.Load<Book>(1);
retrievedBook.ISBN = "978-0553593716";
context.Save(retrievedBook);
```

항목 삭제

항목 삭제 방법:

```
Book retrievedBook = context.Load<Book>(1);
context.Delete(retrievedBook);
```

Xamarin 애플리케이션에서 DynamoDB에 액세스하는 방법에 대한 자세한 내용은 Amazon DynamoDB를 참조하세요.

Amazon Mobile Analytics를 통한 앱 사용 데이터 추적

Amazon Mobile Analytics를 통해 앱 사용량 및 앱 수익을 측정할 수 있습니다. 신규 사용자 대 기존 사용자, 앱 수익, 사용자 유지, 커스텀 인앱 동작 이벤트와 같은 핵심 트렌드를 추적하여 데이터에 기반한 결정을 내려 앱에 대한 사용자 참여와 수익 창출을 높입니다.

다음 자습에서는 앱에 Mobile Analytics를 통합하는 방법을 설명합니다.

프로젝트 설정

필수 조건

이 자습서를 시작하기 전에 .NET 및 Xamarin용 AWS Mobile SDK 설정의 지침을 모두 완료해야 합니다.

Mobile Analytics 콘솔에서 앱 생성

Amazon Mobile Analytics 콘솔로 이동하여 앱을 생성합니다. 나중에 필요하므로 `appId` 값을 기록합니다. Mobile Analytics 콘솔에서 앱을 생성할 때 자격 증명 풀 ID를 지정해야 합니다. 자격 증명 풀을 생성하기 위한 지침은 .NET 및 Xamarin용 AWS Mobile SDK 설정을 참조하세요.

콘솔 사용에 대한 자세한 내용은 Amazon Mobile Analytics 사용 설명서를 참조하세요.

Mobile Analytics에 대한 권한 설정

설정 시 생성한 역할에 연결된 기본 정책은 애플리케이션에 Mobile Analytics에 대한 액세스 권한을 부여합니다. 추가 구성이 필요하지 않습니다.

프로젝트에 Mobile Analytics용 NuGet 패키지 추가

.NET 및 Xamarin용 AWS Mobile SDK 설정 내 지침의 4단계를 따라 Mobile Analytics NuGet 패키지를 프로젝트에 추가합니다.

Mobile Analytics 설정 구성

Mobile Analytics는 몇 가지 설정을 정의하며, 각 설정은 awsconfig.xml 파일에서 구성할 수 있습니다.

```
var config = new MobileAnalyticsManagerConfig();
config.AllowUseDataNetwork = true;
config.DBWarningThreshold = 0.9f;
config.MaxDBSize = 5242880;
config.MaxRequestSize = 102400;
config.SessionTimeout = 5;
```

- AllowUseDataNetwork - 세션 이벤트가 데이터 네트워크에서 전송되는지 여부를 지정하는 부울입니다.

- DBWarningThreshold - 데이터베이스 크기 제한입니다. 이 제한에 도달할 경우 경고 로그가 생성됩니다.
- MaxDBSize - SQLIte 데이터베이스의 크기입니다. 데이터베이스가 최대 크기에 도달하면 추가 이벤트가 삭제됩니다.
- MaxRequestSize - HTTP 요청에서 Mobile Analytics 서비스로 전송되어야 할 요청의 최대 크기(바이트)입니다.
- SessionTimeout - 애플리케이션이 백그라운드로 전환된 후 세션이 종료될 때까지의 시간 간격입니다.

위에 표시된 설정은 각 구성 항목의 기본값입니다.

MobileAnalyticsManager 초기화

MobileAnalyticsManager를 초기화하려면 `MobileAnalyticsManager`에서 GetOrCreateInstance를 호출하여 AWS 자격 증명, 리전, Mobile Analytics 애플리케이션 ID 및 config 객체(선택 사항)를 가져옵니다.

```
var manager = MobileAnalyticsManager.GetOrCreateInstance(
  "APP_ID",
  "Credentials",
  "RegionEndPoint",
  config
);
```

세션 이벤트 추적

Xamarin Android

세션 이벤트를 기록하려면 작업의 `OnPause()` 및 `OnResume()` 메서드를 재정의합니다.

```
protected override void OnResume()
{
    manager.ResumeSession();
    base.OnResume();
}

protected override void OnPause()
{
    manager.PauseSession();
```

```
    base.OnPause();
}
```

이는 애플리케이션 내 각 작업에 대해 구현되어야 합니다.

Xamarin iOS

AppDelegate.cs에서

```
public override void DidEnterBackground(UIApplication application)
{
    manager.PauseSession();
}

public override void WillEnterForeground(UIApplication application)
{
    manager.ResumeSession();
}
```

Mobile Analytics에 대한 자세한 내용은 Amazon Mobile Analytics를 참조하세요.

SNS를 사용하여 푸시 알림 수신(Xamarin iOS)

이 문서에서는 Amazon Simple Notification Service(SNS) 및 .NET 및 Xamarin용 AWS Mobile SDK를 사용하여 Xamarin iOS 애플리케이션으로 푸시 알림을 전송하는 방법을 설명합니다.

프로젝트 설정

필수 조건

이 자습서를 시작하기 전에 .NET 및 Xamarin용 AWS Mobile SDK 설정의 지침을 모두 완료해야 합니다.

SNS에 대한 권한 설정

.NET 및 Xamarin용 AWS Mobile SDK 설정의 2단계를 따라 아래에서 언급하는 정책을 애플리케이션의 역할에 연결합니다. 그러면 애플리케이션이 SNS에 액세스할 수 있는 적절한 권한을 부여 받습니다.

1. IAM 콘솔로 이동하여 구성할 IAM 역할을 선택합니다.
2. 정책 연결을 클릭하고 AmazonSNSFullAccess 정책을 선택한 다음 정책 연결을 클릭합니다.

> **Warning**
>
> AmazonSNSFullAccess를 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 여기서는 빠르게 실행할 수 있도록 사용하는 것입니다. IAM 역할 권한 지정에 대한 자세한 내용은 IAM 역할 권한 개요를 참조하세요.

Apple iOS 개발자 프로그램 멤버십 받기

푸시 알림을 수신하려면 물리적 디바이스에서 앱을 실행해야 합니다. 디바이스에서 앱을 실행하려면 Apple iOS 개발자 프로그램 멤버십이 있어야 합니다. 멤버십이 있으면 Xcode를 사용하여 서명 자격 증명을 생성할 수 있습니다. 자세한 내용은 Apple의 App Distribution Quick Start 설명서를 참조하세요.

iOS 인증서 생성

먼저 iOS 인증서를 생성해야 합니다. 그런 다음 푸시 알림용으로 구성된 프로비저닝 프로파일을 생성해야 합니다. 그렇게 하려면 다음을 수행하세요.

1. Apple Developer Member Center로 이동하여 인증서, ID, 프로필을 클릭합니다.
2. iOS 앱에서 식별자를 클릭하고 웹 페이지 오른쪽 상단의 더하기 버튼을 클릭하여 새 iOS 앱 ID를 추가한 다음 앱 ID 설명을 입력합니다.
3. 아래로 스크롤하여 Add ID Suffix(ID 접미사 추가) 섹션에서 Explicit App ID(명시적 앱 ID)를 선택하고 번들 식별자를 입력합니다.
4. 아래로 스크롤하여 App Services(앱 서비스) 섹션에서 푸시 알림을 선택합니다.
5. 계속을 클릭합니다.
6. 제출을 클릭합니다.
7. 완료를 클릭합니다.
8. 방금 생성한 앱 ID를 선택하고 편집을 클릭합니다.
9. 아래로 스크롤하여 푸시 알림 섹션을 찾습니다. 개발 SSL 인증서 아래에서 인증서 생성을 클릭합니다.
10. 지침을 따라 인증서 서명 요청(CSR)을 생성하고 요청을 업로드한 다음 Apple 알림 서비스(APNS)와 통신하는 데 사용될 SSL 인증서를 다운로드합니다.
11. Certificates, Identifiers & Profiles(인증서, ID, 프로필) 페이지로 돌아갑니다. 프로비저닝 프로파일 아래의 모두를 클릭합니다.
12. 오른쪽 위 모서리에 있는 더하기 버튼을 클릭하여 새 프로비저닝 프로파일을 추가합니다.

13iOS 앱 개발을 선택하고 계속을 클릭합니다.

14.앱 ID를 선택하고 계속을 클릭합니다.

15.개발자 인증서를 선택하고 계속을 클릭합니다.

16.디바이스를 선택하고 계속을 클릭합니다.

17.프로필 이름을 입력하고 생성을 클릭합니다.

18.프로비전 파일을 다운로드하고 두 번 클릭하여 프로비저닝 프로파일을 설치합니다.

푸시 알림용으로 구성된 프로파일을 프로비저닝하는 데 대한 자세한 내용은 Apple의 Configuring Push Notifications 설명서를 참조하세요.

SNS 콘솔에서 인증서를 사용해 플랫폼 ARN 생성

1. KeyChain 액세스 앱을 실행하고 화면의 왼쪽 아래에서 내 인증서를 선택합니다. 그런 다음 앞서 생성한 SSL 인증서를 마우스 오른쪽 버튼으로 클릭하여 APNS에 연결하고 내보내기를 선택합니다. 파일의 이름과 인증서를 보호할 암호를 지정하라는 메시지가 표시됩니다. 인증서는 P12 파일로 저장됩니다.
2. SNS 콘솔로 이동하여 화면 왼쪽에서 애플리케이션을 클릭합니다.
3. 플랫폼 애플리케이션 생성을 클릭하여 새 SNS 플랫폼 애플리케이션을 생성합니다.
4. 애플리케이션 이름을 입력합니다.
5. 푸시 알림 플랫폼으로 Apple Development를 선택합니다.
6. 파일 선택을 선택하고 SSL 인증서를 내보낼 때 생성한 P12 파일을 선택합니다.
7. SSL 인증서를 내보낼 때 지정한 암호를 입력하고 파일에서 자격 증명 로드를 클릭합니다.
8. 플랫폼 애플리케이션 생성을 클릭합니다.
9. 방금 생성한 플랫폼 애플리케이션을 선택하고 애플리케이션 ARN을 복사합니다. 나중 단계에서 이 ARN이 필요합니다.

프로젝트에 SNS용 NuGet 패키지 추가

.NET 및 Xamarin용 AWS Mobile SDK 설정 내 지침의 4단계를 따라 Amazon Simple Notification Service NuGet 패키지를 프로젝트에 추가합니다.

SNS 클라이언트 생성

```
var snsClient = new AmazonSimpleNotificationServiceClient(credentials, region);
```

원격 알림에 애플리케이션 등록

애플리케이션을 등록하려면 아래와 같이 UIApplication 객체에서 RegisterForRemoteNotifications를 호출합니다. AppDelegate.cs에 다음 코드를 배치합니다(아래에서 메시지가 표시된 위치에 플랫폼 애플리케이션 ARN을 삽입).

```
public override bool FinishedLaunching(UIApplication app, NSDictionary options) {
// do something
var pushSettings = UIUserNotificationSettings.GetSettingsForTypes (
  UIUserNotificationType.Alert |
  UIUserNotificationType.Badge |
  UIUserNotificationType.Sound,
  null
);
app.RegisterUserNotifications(pushSettings);
app.RegisterForRemoteNotifications();
// do something
  return true;
}

public override void RegisteredForRemoteNotifications(UIApplication application, NSData
 token) {
  var deviceToken = token.Description.Replace("<", "").Replace(">", "").Replace(" ",
 "");
  if (!string.IsNullOrEmpty(deviceToken)) {
    //register with SNS to create an endpoint ARN
    var response = await SnsClient.CreatePlatformEndpointAsync(
    new CreatePlatformEndpointRequest {
      Token = deviceToken,
      PlatformApplicationArn = "YourPlatformArn" /* insert your platform application
 ARN here */
    });
  }
}
```

SNS 콘솔에서 엔드포인트로 메시지 전송

1. SNS 콘솔 > 애플리케이션으로 이동합니다.
2. 플랫폼 애플리케이션을 선택하고 엔드포인트를 선택한 다음 엔드포인트에 게시를 클릭합니다.
3. 텍스트 상자에 텍스트 메시지를 입력하고 메시지 게시를 클릭하여 메시지를 게시합니다.

SNS를 사용하여 푸시 알림 수신(Xamarin Android)

이 자습서에서는 Amazon Simple Notification Service(SNS)와 .NET 및 Xamarin용 AWS Mobile SDK를 사용하여 Xamarin Android 애플리케이션으로 푸시 알림을 전송하는 방법을 설명합니다.

프로젝트 설정

필수 조건

이 자습서를 시작하기 전에 .NET 및 Xamarin용 AWS Mobile SDK 설정의 지침을 모두 완료해야 합니다.

SNS에 대한 권한 설정

.NET 및 Xamarin용 AWS Mobile SDK 설정의 2단계를 따라 아래에서 언급하는 정책을 애플리케이션의 역할에 연결합니다. 그러면 애플리케이션이 SNS에 액세스할 수 있는 적절한 권한을 부여 받습니다.

1. IAM 콘솔로 이동하여 구성할 IAM 역할을 선택합니다.
2. 정책 연결을 클릭하고 AmazonSNSFullAccess 정책을 선택한 다음 정책 연결을 클릭합니다.

> **Warning**
>
> AmazonSNSFullAccess를 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 여기서는 빠르게 실행할 수 있도록 사용하는 것입니다. IAM 역할 권한 지정에 대한 자세한 내용은 IAM 역할 권한 개요를 참조하세요.

Google Cloud에서 푸시 알림 활성화

먼저 새 Google API 프로젝트를 추가합니다.

1. Google 개발자 콘솔로 이동합니다.
2. 프로젝트 생성을 클릭합니다.
3. 새 프로젝트 상자에 프로젝트 이름을 입력하고 프로젝트 ID(나중에 필요)를 적어 둔 다음 생성을 클릭합니다.

그런 다음 프로젝트에서 Google Cloud Messaging(GCM) 서비스를 활성화합니다.

1. Google 개발자 콘솔에는 새 프로젝트가 이미 선택되어 있을 것입니다. 그렇지 않으면 페이지 상단에 있는 드롭다운 목록에서 선택합니다.
2. 페이지 왼쪽의 사이드바에서 APIs & auth(API 및 인증)를 선택합니다.
3. 검색 상자에서 "Google Cloud Messaging for Android"를 입력하고 Google Cloud Messaging for Android 링크를 클릭합니다.
4. Enable API(API 활성화)를 클릭합니다.

마지막으로 API 키를 받습니다.

1. Google 개발자 콘솔에서 APIs & auth(API 및 인증) > 자격 증명을 선택합니다.
2. Public API access(퍼블릭 API 액세스) 아래에서 Create new key(새 키 생성)를 클릭합니다.
3. Create a new key(새 키 생성) 대화 상자에서 Server key(서버 키)를 클릭합니다.
4. 표시되는 대화 상자에서 생성을 클릭하고 표시된 API 키를 복사합니다. 나중에 이 API 키를 사용해 인증을 수행합니다.

SNS 콘솔에서 프로젝트 ID를 사용해 플랫폼 ARN 생성

1. SNS 콘솔로 이동합니다.
2. 화면 왼쪽에 있는 애플리케이션을 클릭합니다.
3. 플랫폼 애플리케이션 생성을 클릭하여 새 SNS 플랫폼 애플리케이션을 생성합니다.
4. 애플리케이션 이름을 입력합니다.
5. 푸시 알림 플랫폼으로 Google 클라우드 메시징(GCM)을 선택합니다.
6. API 키를 API 키로 표시된 텍스트 상자에 붙여 넣습니다.
7. 플랫폼 애플리케이션 생성을 클릭합니다.
8. 방금 생성한 플랫폼 애플리케이션을 선택하고 애플리케이션 ARN을 복사합니다.

프로젝트에 SNS용 NuGet 패키지 추가

.NET 및 Xamarin용 AWS Mobile SDK 설정 내 지침의 4단계를 따라 Amazon Simple Notification Service NuGet 패키지를 프로젝트에 추가합니다.

SNS 클라이언트 생성

```
var snsClient = new AmazonSimpleNotificationServiceClient(credentials, region);
```

원격 알림에 애플리케이션 등록

Android에서 원격 알림에 등록하려면 Google Cloud 메시지를 수신할 수 있는 BroadcastReceiver를 생성해야 합니다. 아래에서 해당 메시지가 표시된 위치의 패키지 이름을 변경합니다.

```
[BroadcastReceiver(Permission = "com.google.android.c2dm.permission.SEND")]
[IntentFilter(new string[] {
      "com.google.android.c2dm.intent.RECEIVE"
}, Categories = new string[] {
      "com.amazonaws.sns" /* change to match your package */
})]
[IntentFilter(new string[] {
      "com.google.android.c2dm.intent.REGISTRATION"
}, Categories = new string[] {
      "com.amazonaws.sns" /* change to match your package */
})]
[IntentFilter(new string[] {
      "com.google.android.gcm.intent.RETRY"
}, Categories = new string[] {
      "com.amazonaws.sns" /* change to match your package */
})]
public class GCMBroadcastReceiver: BroadcastReceiver {
      const string TAG = "PushHandlerBroadcastReceiver";
      public override void OnReceive(Context context, Intent intent) {
              GCMIntentService.RunIntentInService(context, intent);
              SetResult(Result.Ok, null, null);
      }
}

[BroadcastReceiver]
[IntentFilter(new[] {
      Android.Content.Intent.ActionBootCompleted
})]
public class GCMBootReceiver: BroadcastReceiver {
      public override void OnReceive(Context context, Intent intent) {
              GCMIntentService.RunIntentInService(context, intent);
              SetResult(Result.Ok, null, null);
      }
}
```

다음은 BroadcastReceiver로부터 푸시 알림을 수신하여 디바이스의 알림 표시줄에 표시하는 서비스입니다.

```
[Service]
 public class GCMIntentService: IntentService {
  static PowerManager.WakeLock sWakeLock;
  static object LOCK = new object();

  public static void RunIntentInService(Context context, Intent intent) {
    lock(LOCK) {
      if (sWakeLock == null) {
        // This is called from BroadcastReceiver, there is no init.
        var pm = PowerManager.FromContext(context);
        sWakeLock = pm.NewWakeLock(
        WakeLockFlags.Partial, "My WakeLock Tag");
      }
    }

    sWakeLock.Acquire();
    intent.SetClass(context, typeof(GCMIntentService));
    context.StartService(intent);
  }

  protected override void OnHandleIntent(Intent intent) {
    try {
      Context context = this.ApplicationContext;
      string action = intent.Action;

      if (action.Equals("com.google.android.c2dm.intent.REGISTRATION")) {
        HandleRegistration(intent);
      } else if (action.Equals("com.google.android.c2dm.intent.RECEIVE")) {
        HandleMessage(intent);
      }
    } finally {
      lock(LOCK) {
        //Sanity check for null as this is a public method
        if (sWakeLock != null) sWakeLock.Release();
      }
    }
  }

  private void HandleRegistration(Intent intent) {
    string registrationId = intent.GetStringExtra("registration_id");
    string error = intent.GetStringExtra("error");
    string unregistration = intent.GetStringExtra("unregistered");
```

```
    if (string.IsNullOrEmpty(error)) {
      var response = await SnsClient.CreatePlatformEndpointAsync(new
 CreatePlatformEndpointRequest {
        Token = registrationId,
        PlatformApplicationArn = "YourPlatformArn" /* insert your platform application
 ARN here */
      });
    }
  }

  private void HandleMessage(Intent intent) {
    string message = string.Empty;
    Bundle extras = intent.Extras;
    if (!string.IsNullOrEmpty(extras.GetString("message"))) {
      message = extras.GetString("message");
    } else {
      message = extras.GetString("default");
    }

    Log.Info("Messages", "message received = " + message);
    ShowNotification(this, "SNS Push", message);
    //show the message

  }

  public void ShowNotification(string contentTitle,
  string contentText) {
    // Intent
    Notification.Builder builder = new Notification.Builder(this)
      .SetContentTitle(contentTitle)
      .SetContentText(contentText)
      .SetDefaults(NotificationDefaults.Sound | NotificationDefaults.Vibrate)
      .SetSmallIcon(Resource.Drawable.Icon)
      .SetSound(RingtoneManager.GetDefaultUri(RingtoneType.Notification));

    // Get the notification manager:
    NotificationManager notificationManager =
 this.GetSystemService(Context.NotificationService) as NotificationManager;

    notificationManager.Notify(1001, builder.Build());
  }
}
```

SNS 콘솔에서 엔드포인트로 메시지 전송

1. SNS 콘솔 > 애플리케이션으로 이동합니다.
2. 플랫폼 애플리케이션을 선택하고 엔드포인트를 선택한 다음 엔드포인트에 게시를 클릭합니다.
3. 텍스트 상자에 텍스트 메시지를 입력하고 메시지 게시를 클릭하여 메시지를 게시합니다.

Amazon Cognito 자격 증명

Amazon Cognito Identity는 무엇입니까?

Amazon Cognito Identity를 사용하여 사용자 고유의 자격 증명을 생성하고 자격 증명 공급자를 통해 인증할 수 있습니다. 자격 증명으로 권한이 제한된 임시 AWS 자격 증명을 얻어 Amazon Cognito Sync를 통해 데이터를 동기화하거나 다른 AWS 서비스에 직접 액세스할 수 있습니다. Amazon Cognito Identity는 퍼블릭 자격 증명 공급자(Amazon, Facebook 및 Google)와 인증되지 않은 자격 증명을 지원합니다. 또한 자체의 백엔드 인증 프로세스를 통해 사용자를 등록하고 인증할 수 있는 개발자 인증 자격 증명도 지원합니다.

Cognito 자격 증명에 대한 자세한 내용은 Amazon Cognito 개발자 안내서를 참조하세요.

Cognito 인증 리전 가용성에 대한 자세한 내용은 AWS 서비스 리전 가용성을 참조하세요.

퍼블릭 공급자를 사용하여 사용자 인증

Amazon Cognito Identity를 사용하여 사용자 고유의 자격 증명을 생성하고 Amazon S3 또는 Amazon DynamoDB 같은 AWS 리소스에 안전하게 액세스하도록 사용자를 인증할 수 있습니다. Amazon Cognito Identity는 Amazon, Facebook, Twitter/Digits, Google 또는 OpenID Connect 호환 공급자와 같은 퍼블릭 ID 공급자와 인증되지 않은 자격 증명을 지원합니다.

Amazon, Facebook, Twitter/Digits, Google 같은 퍼블릭 자격 증명 공급자를 사용한 사용자 인증에 대한 자세한 내용은 Amazon Cognito 개발자 안내서의 외부 공급자를 참조하세요.

개발자 인증 자격 증명 사용

Amazon Cognito는 Facebook, Google 및 Amazon을 통한 웹 자격 증명 연동 이외에 개발자 인증 자격 증명을 지원합니다. 개발자 인증 자격 증명을 사용하면 Amazon Cognito Sync를 사용하여 사용자 데이터를 동기화하고 AWS 리소스에 액세스하면서도 기존의 자체 인증 프로세스를 통해 사용자를 등록 및 인증할 수 있습니다. 개발자 인증 자격 증명의 사용에는 최종 사용자 장치, 인증을 위한 백엔드 및 Amazon Cognito 간의 상호 작용이 포함됩니다.

개발자 인증 자격 증명에 대한 자세한 내용은 Amazon Cognito 개발자 안내서의 개발자 인증 자격 증명을 참조하세요.

Amazon Cognito Sync

Amazon Cognito Sync는 무엇입니까?

Cognito Sync는 사용자 데이터(예: 게임 점수, 사용자 기본 설정, 게임 상태)의 교차 디바이스 동기화를 활성화하는 AWS 서비스 및 클라이언트 라이브러리입니다. Cognito Sync API를 사용하여 디바이스 간에 사용자 데이터를 동기화할 수 있습니다. 앱에서 Cognito Sync를 사용하려면 프로젝트에 |를 포함해야 합니다.

Amazon Cognito Sync를 애플리케이션에 통합하는 방법에 대한 자세한 내용은 Amazon Cognito Sync 개발자 안내서를 참조하세요.

Amazon Mobile Analytics

Amazon Mobile Analytics는 대규모로 앱 사용량 데이터를 수집, 시각화, 분석 및 추출하는 서비스입니다. Mobile Analytics는 표준 디바이스 데이터와 커스텀 이벤트를 모두 손쉽게 캡처하고 사용자 대신 자동으로 보고서를 계산합니다. 아래 나열된 집계 보고서 이외에 추가 분석으로 위해 자동으로 데이터를 Redshift 및 S3으로 내보내도록 설정할 수도 있습니다.

Amazon Mobile Analytics를 사용하면 고객 행동을 추적하고, 지표를 집계하고, 데이터 시각화를 생성하고, 유의미한 패턴을 식별할 수 있습니다.

핵심 개념

보고서 유형

Mobile Analytics에서는 Mobile Analytics 콘솔을 통해 다음과 같은 보고서를 즉시 사용할 수 있습니다.

- Daily Active Users(DAU), Monthly Active Users(MAU) 및 New Users
- Sticky Factor(DAU / MAU)
- 세션 수 및 Daily Active User당 평균 세션 수
- DAU당 평균 수익(ARPDAU) 및 결제 DAU당 평균 수익(ARPPDAU)
- 1일, 3일, 7일 유지 및 1주, 2주, 3주 유지
- 커스텀 이벤트

다음 보고서가 콘솔에서 6개 보고 탭을 통해 제공됩니다.

- 개요 – 참여를 빠르게 파악할 수 있도록 검토하기 쉬운 대시보드에서 9개의 미리 선택된 보고서를 추적합니다. MAU, DAU, 신규 사용자, 일별 세션, 고정 계수, 1일 유지, ARPDAU, 일별 결제 사용자, ARPPDAU.
- 활성 사용자 – 일별 및 월별로 몇 명의 사용자가 게임에 참여하는지 추적하고 참여, 어필 및 수익화를 측정하기 위해 고정 계수를 모니터링합니다.
- 세션 – 특정일에 앱이 사용되는 횟수와 특정일에 각 사용자가 앱을 여는 횟수를 추적합니다.
- 유지 – 일별 및 주별로 고객이 앱을 다시 사용하는 비율을 추적합니다.
- 수익 – 수익화 개선 영역을 알아보기 위해 앱 내 수익 추세를 추적합니다.
- 커스텀 이벤트 – 앱에 고유한 사용자 지정 사용자 작업을 추적합니다.

Mobile Analytics 보고서 및 Mobile Analytics 콘솔 사용에 대한 자세한 내용은 Mobile Analytics 개발자 안내서의 Mobile Analytics 콘솔 보고서 개요를 참조하세요.

프로젝트 설정

필수 조건

애플리케이션에서 Mobile Analytics를 사용하려면 프로젝트에 SDK를 추가해야 합니다. 이렇게 하려면 .NET 및 Xamarin용 AWS Mobile SDK 설정의 지침을 따르세요.

Mobile Analytics 설정 구성

Mobile Analytics는 몇 가지 설정을 정의하며, 각 설정은 awsconfig.xml 파일에서 구성할 수 있습니다.

```
var config = new MobileAnalyticsManagerConfig();
config.AllowUseDataNetwork = true;
config.DBWarningThreshold = 0.9f;
config.MaxDBSize = 5242880;
config.MaxRequestSize = 102400;
config.SessionTimeout = 5;
```

- SessionTimeout - 앱이 SessionTimeout보다 오래 백그라운드로 유지될 경우 Mobile Analytics가 현재 세션을 종료하고 앱이 포그라운드로 다시 전환되면 새로운 세션이 생성됩니다. 5~10 사이의 값을 사용하는 것이 좋습니다. 기본값은 5입니다.
- MaxDBSize - 이벤트의 로컬 스토리지에 사용되는 데이터베이스의 최대 크기(바이트)입니다. 데이터베이스 크기가 이 값을 초과할 경우 추가 이벤트가 무시됩니다. 1~10MB 사이의 값을 사용하는 것이 좋습니다. 기본값은 5242880(5MB)입니다.
- DBWarningThreashold - 경고 임계값입니다. 사용할 수 있는 값은 0~1입니다. 값이 임계값을 초과할 경우 경고 로그가 생성됩니다. 기본값은 0.9입니다.
- MaxRequestSize - Mobile Analytics 서비스에 대한 HTTP 요청의 최대 크기입니다. 이 값은 바이트 단위로 지정되며 사용 가능한 범위는 1~512KB입니다. 기본값은 102400(100KB)입니다. 512KB보다 큰 값을 사용하지 마세요. 서비스가 HTTP 요청을 거부할 수 있습니다.
- AllowUseDataNetwork - 셀룰러 데이터 네트워크에서 서비스 호출이 허용되는지 여부를 나타내는 값입니다. 고객의 데이터 사용량이 증가할 수 있으므로 이 옵션은 주의해서 사용해야 합니다.

위에 표시된 설정은 각 구성 항목의 기본값입니다.

애플리케이션에 Mobile Analytics 통합

다음 섹션에서는 앱에 Mobile Analytics를 통합하는 방법을 설명합니다.

Mobile Analytics 콘솔에서 앱 생성

Amazon Mobile Analytics 콘솔로 이동하여 앱을 생성합니다. 나중에 필요하므로 `appId` 값을 기록합니다. Mobile Analytics 콘솔에서 앱을 생성할 때 자격 증명 풀 ID를 지정해야 합니다. 자격 증명 풀을 생성하기 위한 지침은 .NET 및 Xamarin용 AWS Mobile SDK 설정을 참조하세요.

Mobile Analytics 콘솔 사용에 대한 자세한 내용은 Mobile Analytics 개발자 안내서의 Mobile Analytics 콘솔 보고서 개요를 참조하세요.

MobileAnalyticsManager 클라이언트 생성

MobileAnalyticsManager를 초기화하려면 `MobileAnalyticsManager`에서 GetOrCreateInstance를 호출하여 AWS 자격 증명, 리전, Mobile Analytics 애플리케이션 ID 및 config 객체(선택 사항)를 가져옵니다.

```
// Initialize the MobileAnalyticsManager
analyticsManager = MobileAnalyticsManager.GetOrCreateInstance(
  cognitoCredentials,
  RegionEndpoint.USEast1,
  APP_ID,
  config
);
```

`APP_ID`는 앱 생성 마법사가 자동으로 생성합니다. 이들 두 값이 Mobile Analytics 콘솔 내 값과 일치해야 합니다. `APP_ID`는 Mobile Analytics 콘솔에서 데이터를 그룹화하는 데 사용됩니다. Mobile Analytics 콘솔에서 앱을 생성한 후 앱 ID를 찾으려면 Mobile Analytics 콘솔로 이동하여 화면의 오른쪽 위에 있는 기어 모양 아이콘을 클릭합니다. 그러면 [App Management] 페이지가 열려 모든 등록된 앱 및 해당 앱 ID가 나열됩니다.

수익화 이벤트 기록

.NET 및 Xamarin용 AWS Mobile SDK는 `MonetizationEvent` 클래스를 제공합니다. 이 클래스를 사용하면 수익화 이벤트를 생성하여 모바일 애플리케이션 내에서 발생한 구매를 추적할 수 있습니다. 다음 코드 조각은 수익화 이벤트를 생성하는 방법을 보여줍니다.

```
// Create the monetization event object
MonetizationEvent monetizationEvent = new MonetizationEvent();

// Set the details of the monetization event
monetizationEvent.Quantity = 3.0;
monetizationEvent.ItemPrice = 1.99;
monetizationEvent.ProductId = "ProductId123";
monetizationEvent.ItemPriceFormatted = "$1.99";
monetizationEvent.Store = "Your-App-Store";
monetizationEvent.TransactionId = "TransactionId123";
monetizationEvent.Currency = "USD";

// Record the monetiziation event
analyticsManager.RecordEvent(monetizationEvent);
```

커스텀 이벤트 기록

Mobile Analytics를 사용하면 커스텀 이벤트를 정의할 수 있습니다. 커스텀 이벤트는 전적으로 개발자가 정의합니다. 이러한 이벤트는 앱 또는 게임에 고유한 사용자 작업을 추적하는 데 도움이 됩니다. 커스텀 이벤트에 대한 자세한 내용은 커스텀 이벤트를 참조하세요.

이 예제에서는 앱이 게임이고 사용자가 레벨을 완료할 때마다 이벤트를 기록하는 것으로 가정합니다. 새 `AmazonMobileAnalyticsEvent` 인스턴스를 생성하여 "LevelComplete" 이벤트를 생성합니다.

```
CustomEvent customEvent = new CustomEvent("LevelComplete");

// Add attributes
customEvent.AddAttribute("LevelName","Level1");
customEvent.AddAttribute("CharacterClass","Warrior");
customEvent.AddAttribute("Successful","True");

// Add metrics
customEvent.AddMetric("Score",12345);
customEvent.AddMetric("TimeInLevel",64);

// Record the event
analyticsManager.RecordEvent(customEvent);
```

세션 기록

Xamarin iOS

애플리케이션이 포커스를 잃을 경우 세션을 일시 중지할 수 있습니다. iOS 앱인 경우 다음 코드 조각에 표시된 대로 AppDelegate.cs 파일에서 `DidEnterBackground` 및 `WillEnterForeground`를 재정의하여 `MobileAnalyticsManager.PauseSesion` 및 `MobileAnalyticsManager.ResumeSession`을 호출합니다.

```
public override void DidEnterBackground(UIApplication application)
{
  // ...
  _manager.PauseSesson();
  // ...
}

public override void WillEnterForeground(UIApplication application)
{
  // ...
  _manager.ResumeSession();
  // ...
}
```

Xamarin Android

Android 앱인 경우 다음 코드 조각에 표시된 대로 OnPause() 메서드에서 `MobileAnalyticsManager.PauseSesion`을 호출하고 OnResume() 메서드에서 `MobileAnalyticsManager.ResumeSession`을 호출합니다.

```
protected override void OnResume()
{
  _manager.ResumeSession();
  base.OnResume();
}

protected override void OnPause()
{
  _manager.PauseSession();
  base.OnPause();
}
```

기본적으로 사용자가 5초 미만으로 앱에서 포커스를 돌렸다가 다시 앱에 포커스를 맞출 경우 세션이 재개됩니다. 사용자가 5초 이상 앱에서 포커스를 돌릴 경우 새로운 세션이 생성됩니다. 이 설정은 aws_mobile_analytics.json 구성 파일에서 "SESSION_DELTA" 속성을 신규 세션을 생성하기 전에 대기할 시간(초)으로 설정하여 구성할 수 있습니다.

Amazon Simple Storage Service(S3)

S3는 무엇입니까?

Amazon Simple Storage Service(Amazon S3)는 개발자에 안전하고 내구성과 확장성이 뛰어난 객체 스토리지를 제공합니다. Amazon S3는 간단한 웹 서비스 인터페이스를 통해 웹 어디서나 원하는 양의 데이터를 저장 및 검색할 수 있으므로 사용하기가 쉽습니다. Amazon S3에서는 사용한 스토리지에 대해서만 비용을 지불합니다. 최소 요금이나 설치 비용이 없습니다.

Amazon S3는 클라우드 애플리케이션, 콘텐츠 배포, 백업 및 아카이빙, 재해 복구, 빅 데이터 분석을 비롯한 다양한 사용 사례에 적합한 비용 효과적인 객체 스토리지를 제공합니다.

AWS S3 리전 가용성에 대한 자세한 내용은 AWS 서비스 리전 가용성을 참조하세요.

핵심 개념

버킷

Amazon S3에 저장한 모든 객체는 버킷에 존재합니다. 디렉토리로 파일 시스템 내 파일을 그룹화하듯 버킷으로 관련 객체를 그룹화할 수 있습니다. 버킷은 액세스 권한 및 버전 관리 상태 등의 속성을 지니며, 사용자는 버킷이 속할 리전을 원하는 대로 설정할 수 있습니다.

S3 버킷에 대한 자세한 내용은 S3 개발자 안내서의 버킷 작업을 참조하세요.

객체

객체는 Amazon S3에 저장되는 데이터입니다. 모든 객체는 특정 AWS 리전에서 생성되는 버킷 내에 상주합니다.

특정 리전에 저장된 객체는 사용자가 명시적으로 객체를 다른 리전으로 전송하지 않는 한 해당 리전을 벗어나지 않습니다. 예를 들어, EU(아일랜드) 리전에 저장된 객체는 EU 밖으로 이동하지 않습니다. Amazon S3 리전에 저장된 객체는 물리적으로 해당 리전에 유지됩니다. Amazon S3에서는 복사본을 유지하거나 객체를 다른 지역으로 이동하지 않습니다. 그러나 필요한 권한이 있는 한 어디서든 객체에 액세스할 수 있습니다.

객체는 이미지, 백업 데이터, 동영상 등 임의의 파일 형식일 수 있습니다. 객체는 최대 5TB가 될 수 있습니다. 또한 버킷에 저장할 수 있는 객체 수에는 제한이 없습니다.

객체를 Amazon S3로 업로드하려면 먼저 버킷에 대한 쓰기 권한이 있어야 합니다. 버킷 권한을 설정하는 자세한 방법은 S3 개발자 안내서의 버킷 권한 편집을 참조하세요.

S3 객체에 대한 자세한 내용은 S3 개발자 안내서의 객체 작업을 참조하세요.

객체 메타데이터

Amazon S3의 각 객체에는 해당 메타데이터를 나타내는 키-값 페어 세트가 있습니다. 다음과 같은 두 가지 유형의 메타데이터가 있습니다.

- 시스템 메타데이터 – 때때로 Amazon S3에 의해 처리됩니다. 예: 콘텐츠 유형 및 콘텐츠 길이.
- 사용자 메타데이터 - 절대로 Amazon S3에 의해 처리되지 않습니다. 사용자 메타데이터는 객체와 함께 저장되고 반환됩니다. 사용자 메타데이터의 최대 크기는 2KB이며, 키와 해당 값 모두 US-ASCII 표준에 부합해야 합니다.

S3 객체 메타데이터에 대한 자세한 내용은 객체 메타데이터 편집을 참조하세요.

프로젝트 설정

필수 조건

애플리케이션에서 Amazon S3를 사용하려면 프로젝트에 SDK를 추가해야 합니다. 이렇게 하려면 .NET 및 Xamarin용 AWS Mobile SDK 설정의 지침을 따르세요.

S3 버킷 생성

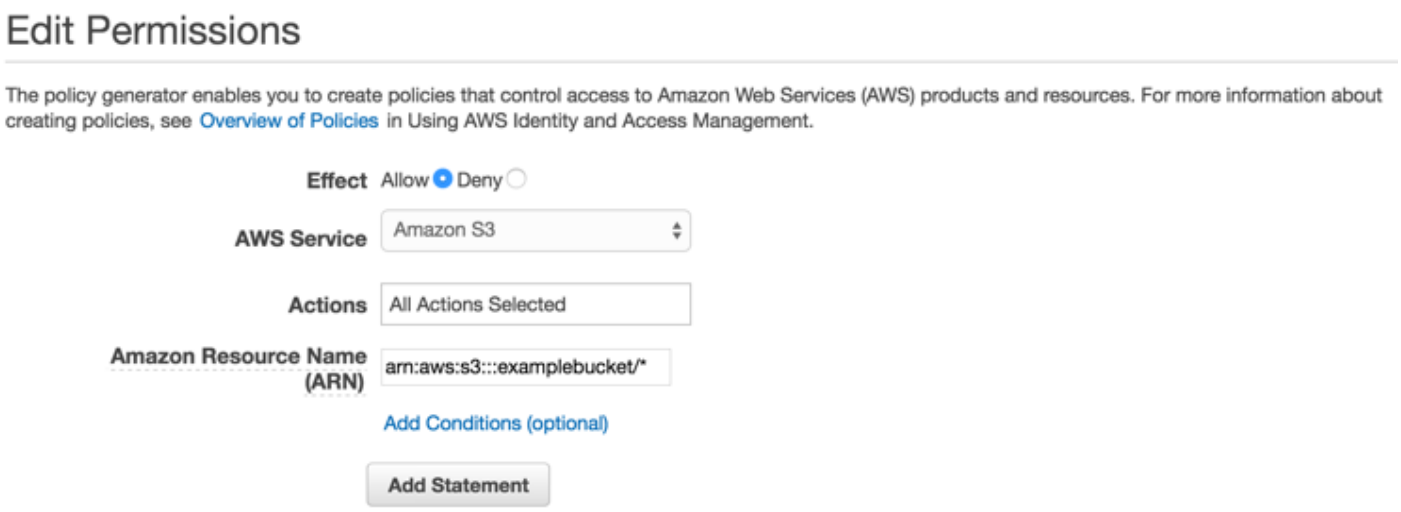
Amazon S3는 애플리케이션의 리소스를 Amazon S3 버킷(특정 리전에 위치하는 클라우드 스토리지 컨테이너)에 저장합니다. 각 Amazon S3 버킷은 전역적으로 고유한 이름을 가져야 합니다. Amazon S3 콘솔을 사용하여 버킷을 생성할 수 있습니다.

1. Amazon S3 콘솔에 로그인하고 버킷 만들기를 클릭합니다.
2. 버킷 이름을 입력하고 리전을 선택한 다음 생성을 클릭합니다.

S3에 대한 권한 설정

기본 IAM 역할 정책은 애플리케이션에 Amazon Mobile Analytics 및 Amazon Cognito Sync에 대한 액세스 권한을 부여합니다. Cognito 자격 증명 풀이 Amazon S3에 액세스할 수 있으려면 자격 증명 풀의 역할을 수정해야 합니다.

1. Identity and Access Management 콘솔로 이동하여 왼쪽 창에서 역할을 클릭합니다.
2. 검색 상자에 자격 증명 풀 이름을 입력합니다. 인증된 사용자와 인증되지 않은 사용자에 대해 하나씩 2개의 역할이 나열됩니다.
3. 인증되지 않은 사용자의 역할을 클릭합니다(자격 증명 풀 이름에 unauth가 추가됨).
4. 역할 정책 생성을 클릭하고 정책 생성기를 선택한 다음 선택을 클릭합니다.
5. Edit Permissions(권한 편집) 페이지에서 다음 이미지에 표시된 설정을 입력합니다. Amazon 리소스 이름(ARN)은 사용자의 것으로 바꿔야 합니다. S3 버킷의 ARN은 `arn:aws:s3:::examplebucket/*`과 비슷하며, 버킷이 위치하는 리전과 버킷의 이름으로 구성됩니다. 아래 표시된 설정은 자격 증명 풀에 지정된 버킷의 모든 작업에 대한 전체 액세스를 부여합니다.



1. 설명문 추가 버튼을 클릭한 다음 다음 단계를 클릭합니다.
2. 마법사가 앞서 생성한 구성을 표시합니다. 정책 적용을 클릭합니다.

S3 액세스 권한 부여에 대한 자세한 내용은 Amazon S3 버킷에 대한 액세스 권한 부여를 참조하세요.

(선택 사항) S3 요청의 서명 버전을 구성합니다.

Amazon S3와의 모든 상호 작용은 인증을 거치거나 익명으로 할 수 있습니다. AWS는 서명 버전 4 또는 서명 버전 2 알고리즘을 사용해 서비스 호출을 인증합니다.

2014년 1월 이후 생성된 모든 새 AWS 리전은 서명 버전 4만 지원합니다. 하지만 그 이전의 리전은 계속해서 서명 버전 4 및 서명 버전 2 요청을 지원합니다.

버킷이 이 페이지에 나열된 서명 버전 2 요청을 지원하지 않는 리전에 위치하는 경우 다음과 같이 AWSConfigsS3.UseSignatureVersion4 속성을 "true"로 설정해야 합니다.

```
AWSConfigsS3.UseSignatureVersion4 = true;
```

AWS 서명 버전에 대한 자세한 내용은 요청 인증(AWS 서명 버전 4)을 참조하세요.

애플리케이션에 S3 통합

Xamarin 애플리케이션에서 S3와 상호 작용하는 방법은 두 가지가 있습니다. 다음 주제에서 두 방법을 자세히 살펴봅니다.

S3 전송 유틸리티 사용

S3 Transfer Utility를 사용하면 보다 간편하게 S3와 Xamarin 애플리케이션 사이에서 파일을 업로드 및 다운로드할 수 있습니다.

TransferUtility 초기화

다음과 같이 S3 클라이언트를 생성하여 AWS 자격 증명 객체를 전달한 다음 S3 클라이언트를 TransferUtility로 전달합니다.

```
var s3Client = new AmazonS3Client(credentials,region);
var transferUtility = new TransferUtility(s3Client);
```

(선택 사항) TransferUtility 구성

선택적 속성 3개를 구성할 수 있습니다.

- ConcurrentServiceRequests - 파일 업로드/다운로드에 사용될 활성 스레드 또는 동시 비동기 웹 요청 수를 지정합니다. 기본값은 10입니다.
- MinSizeBeforePartUpload - 업로드 파트의 최소 파트 크기(바이트)를 가져오거나 설정합니다. 기본값은 16MB입니다. 최소 파트 크기를 낮출 경우 멀티파트 업로드가 더 많은 수의 더 작은 파트로 분할됩니다. 이 값을 너무 낮게 설정하면 전송 속도에 악영향을 미쳐 지연 시간과 각 파트의 네트워크 통신이 증가합니다.
- NumberOfUploadThreads - 실행 스레드 수를 가져오거나 설정합니다. 이 속성은 파일 업로드 시 사용될 활성 스레드 수를 지정합니다. 기본값은 10개 스레드입니다.

S3 TransferUtility 클라이언트를 구성하려면 다음과 같이 config 객체를 생성하고, 속성을 설정한 다음 객체를 TransferUtility 생성자로 전달합니다.

```
var config = new TransferUtilityConfig();

config.ConcurrentServiceRequests = 10;
config.MinSizeBeforePartUpload=16*1024*1024;
config.NumberOfUploadThreads=10;

var s3Client = new AmazonS3Client(credentials);
var utility = new TransferUtility(s3Client,config);
```

파일 다운로드

S3에서 파일을 다운로드하려면 Transfer Utility 객체에서 `Download`를 호출하여 다음 파라미터를 전달합니다.

- `file` - 다운로드할 파일의 문자열 이름
- `bucketName` - 다운로드할 파일이 저장된 S3 버킷의 문자열 이름
- `key` - 다운로드할 S3 객체(이 경우에는 파일)의 이름을 표시하는 문자열

```
transferUtility.Download(
  Path.Combine(Environment.SpecialFolder.ApplicationData,"file"),
  "bucketName",
  "key"
);
```

파일 업로드

S3로 파일을 업로드하려면 Transfer Utility 객체에서 `Upload`를 호출하여 다음 파라미터를 전달합니다.

- `file` - 업로드할 파일의 문자열 이름
- `bucketName` - 파일을 저장할 S3 버킷의 문자열 이름

```
transferUtility.Upload(
  Path.Combine(Environment.SpecialFolder.ApplicationData,"file"),
```

```
  "bucketName"
);
```

위 코드는 Environment.SpecialFolder.ApplicationData 디렉터리에 파일이 있다고 가정합니다. 업로드는 처리량을 높이기 위해 대용량 파일에서 자동으로 S3의 멀티파트 업로드 기능을 사용합니다.

서비스 수준 S3 API 사용

S3 TransferUtility를 사용하는 이외에 하위 수준 S3 API를 사용하여 S3와 상호 작용할 수도 있습니다.

Amazon S3 클라이언트 초기화

Amazon S3를 사용하려면 먼저 앞서 생성한 CognitoAWSCredentials 인스턴스 및 리전을 참조하는 AmazonS3Client 인스턴스를 생성해야 합니다.

```
AmazonS3Client S3Client = new AmazonS3Client (credentials,region);
```

파일 다운로드

S3에서 파일을 다운로드하는 방법:

```
// Create a GetObject request
GetObjectRequest request = new GetObjectRequest
{
    BucketName = "SampleBucket",
    Key = "Item1"
};

// Issue request and remember to dispose of the response
using (GetObjectResponse response = client.GetObject(request))
{
    using (StreamReader reader = new StreamReader(response.ResponseStream))
    {
        string contents = reader.ReadToEnd();
        Console.WriteLine("Object - " + response.Key);
        Console.WriteLine(" Version Id - " + response.VersionId);
        Console.WriteLine(" Contents - " + contents);
    }
}
```

파일 업로드

S3로 파일을 업로드하는 방법:

```
// Create a client
AmazonS3Client client = new AmazonS3Client();

// Create a PutObject request
PutObjectRequest request = new PutObjectRequest
{
    BucketName = "SampleBucket",
    Key = "Item1",
    FilePath = "contents.txt"
};

// Put object
PutObjectResponse response = client.PutObject(request);
```

항목 삭제

S3에서 항목을 삭제하는 방법:

```
// Create a client
AmazonS3Client client = new AmazonS3Client();

// Create a DeleteObject request
DeleteObjectRequest request = new DeleteObjectRequest
{
    BucketName = "SampleBucket",
    Key = "Item1"
};

// Issue request
client.DeleteObject(request);
```

다중 항목 삭제

단일 HTTP 요청을 사용하여 한 버킷에서 여러 객체를 삭제하는 방법:

```
// Create a client
AmazonS3Client client = new AmazonS3Client();
```

```
// Create a DeleteObject request
DeleteObjectsRequest request = new DeleteObjectsRequest
{
    BucketName = "SampleBucket",
    Objects = new List<KeyVersion>
    {
        new KeyVersion() {Key = "Item1"},
        // Versioned item
        new KeyVersion() { Key = "Item2", VersionId =
 "Rej8CiBxcZKVK81cLr39j27Y5FVXghDK", },
        // Item in subdirectory
        new KeyVersion() { Key = "Logs/error.txt"}
    }
};

try
{
    // Issue request
    DeleteObjectsResponse response = client.DeleteObjects(request);
}
catch (DeleteObjectsException doe)
{
    // Catch error and list error details
    DeleteObjectsResponse errorResponse = doe.Response;

    foreach (DeletedObject deletedObject in errorResponse.DeletedObjects)
    {
        Console.WriteLine("Deleted item " + deletedObject.Key);
    }
    foreach (DeleteError deleteError in errorResponse.DeleteErrors)
    {
        Console.WriteLine("Error deleting item " + deleteError.Key);
        Console.WriteLine(" Code - " + deleteError.Code);
        Console.WriteLine(" Message - " + deleteError.Message);
    }
}
```

최대 1,000개의 키를 지정할 수 있습니다.

버킷 목록 생성

요청의 인증된 발신자가 소유한 모든 버킷의 목록을 반환하는 방법:

```
// Create a client
AmazonS3Client client = new AmazonS3Client();

// Issue call
ListBucketsResponse response = client.ListBuckets();

// View response data
Console.WriteLine("Buckets owner - {0}", response.Owner.DisplayName);
foreach (S3Bucket bucket in response.Buckets)
{
    Console.WriteLine("Bucket {0}, Created on {1}", bucket.BucketName,
 bucket.CreationDate);
}
```

객체 목록 생성

S3 버킷에 저장된 객체를 일부 또는 전부(최대 1,000개) 반환할 수 있습니다. 이렇게 하려면 버킷에 대한 읽기 액세스 권한이 있어야 합니다.

```
// Create a GetObject request
GetObjectRequest request = new GetObjectRequest
{
  BucketName = "SampleBucket",
  Key = "Item1"
};

// Issue request and remember to dispose of the response
using (GetObjectResponse response = client.GetObject(request))
{
  using (StreamReader reader = new StreamReader(response.ResponseStream))
  {
      string contents = reader.ReadToEnd();
      Console.WriteLine("Object - " + response.Key);
      Console.WriteLine(" Version Id - " + response.VersionId);
      Console.WriteLine(" Contents - " + contents);
  }
}
```

버킷 리전 가져오기

버킷이 위치하는 리전을 가져오는 방법:

```
// Create a client
AmazonS3Client client = new AmazonS3Client();

// Construct request
GetBucketLocationRequest request = new GetBucketLocationRequest
{
    BucketName = "SampleBucket"
};

// Issue call
GetBucketLocationResponse response = client.GetBucketLocation(request);

// View response data
Console.WriteLine("Bucket location - {0}", response.Location);
```

버킷 정책 가져오기

버킷 정책을 가져오는 방법:

```
// Create a client
AmazonS3Client client = new AmazonS3Client();

// Construct request
GetBucketPolicyRequest getRequest = new GetBucketPolicyRequest
{
   BucketName = "SampleBucket"
};
string policy = client.GetBucketPolicy(getRequest).Policy;

Console.WriteLine(policy);
Debug.Assert(policy.Contains("BasicPerms"));
```

Amazon DynamoDB

What is Amazon DynamoDB?

Amazon DynamoDB은 속도가 빠르고 확장성이 뛰어난 비 관계형 데이터베이스 서비스입니다. DynamoDB는 기존 데이터 스토리지에서의 확장성 제한을 없애면서도 낮은 지연 시간과 예측 가능한 성능을 유지합니다.

핵심 개념

DynamoDB 데이터 모델 개념에는 테이블, 항목 및 속성이 포함됩니다.

테이블

Amazon DynamoDB에서 데이터베이스는 테이블의 모음입니다. 테이블은 항목 집합이고, 그리고 각 항목은 속성 집합입니다.

관계형 데이터베이스에서는 테이블 이름, 기본 키, 열 이름 목록 및 데이터 형식 등의 테이블 스키마가 사전에 정의되어 있습니다. 테이블에 저장되는 레코드 역시 모두 동일한 열 집합을 가져야 합니다. 대조적으로 DynamoDB에서는 테이블에 기본 키만 있으면 되며, 모든 속성 이름과 데이터 형식을 사전에 정의할 필요도 없습니다.

테이블 작업에 대한 자세한 내용은 DynamoDB의 테이블 작업을 참조하세요.

항목 및 속성

DynamoDB 테이블의 각 항목은 크기가 400KB로 제한되기는 하지만 속성은 몇 개든지 가질 수 있습니다. 항목 크기는 속성 이름 길이와 값의 길이(이진수와 UTF-8 길이)를 더하여 결정됩니다.

각 항목 속성은 이름-값 페어입니다. 또한 단일 값이나 다중 값의 집합이 되기도 합니다. 예들 들어 서적 항목은 제목과 저자 속성, 두 가지를 가질 수 있습니다. 서적마다 제목은 하나지만 저자가 다수일 수도 있습니다. 이러한 다중 값 속성은 집합을 이루지만 중복 값은 허용되지 않습니다.

예를 들어 DynamoDB에 제품 카탈로그를 저장한다고 가정하겠습니다. Id 속성을 기본 키로 하여 ProductCatalog 테이블을 생성할 수 있습니다. 기본 키는 각 항목의 고유 식별자입니다. 따라서 테이블의 두 제품이 동일한 기본 ID를 가질 수는 없습니다.

항목 작업에 대한 자세한 내용은 DynamoDB의 항목 작업을 참조하세요.

데이터 유형

Amazon DynamoDB는 다음 데이터 형식을 지원합니다.

- 스칼라 형식 – 숫자, 문자열, 이진수, 부울 및 null
- 다중 값 형식 – 문자열 집합, 숫자 집합 및 이진수 집합
- 문서 형식 – 목록 및 맵

스칼라 데이터 형식, 다중 값 데이터 형식 및 문서 데이터 형식에 대한 자세한 내용은 DynamoDB 데이터 형식을 참조하세요.

기본 키

테이블을 생성할 때는 테이블 이름 외에도 테이블의 기본 키를 지정해야 합니다. 기본 키는 테이블의 각 항목을 나타내는 고유 식별자입니다. 따라서 두 항목이 동일한 키를 가질 수는 없습니다. DynamoDB는 다음과 같이 두 가지 형식의 기본 키를 지원합니다.

- 해시 기본 키: 기본 키가 한 가지 속성인 해시 속성으로 구성됩니다. DynamoDB는 이 기본 키 속성을 기반으로 정렬되지 않은(unordered) 해시 인덱스를 빌드합니다. 테이블의 각 항목은 해시 키 값을 기준으로 고유 식별됩니다.
- 해시 및 범위 키: 기본 키가 두 가지 속성으로 구성됩니다. 첫 번째 속성은 해시 속성이고, 두 번째는 범위 속성입니다. DynamoDB는 해시 기본 키 속성을 기반으로 정렬되지 않은(unordered) 해시 인덱스를 빌드하고, 범위 기본 키 속성을 기반으로 정렬된 범위 인덱스를 빌드합니다. 테이블의 각 항목은 해시 키 값을 기준으로 고유 식별됩니다. 두 항목이 동일한 해시 키 값을 가질 수는 있지만 범위 키 값은 서로 달라야 합니다.

보조 인덱스

해시 및 범위 키와 함께 테이블을 생성하면 옵션으로 해당 테이블에 보조 인덱스를 하나 이상 정의할 수 있습니다. 보조 인덱스를 사용하면 기본 키에 대한 쿼리는 물론이고 대체 키를 사용하여 테이블의 데이터도 쿼리할 수 있습니다.

DynamoDB는 로컬 보조 인덱스와 글로벌 보조 인덱스 두 유형의 보조 인덱스를 지원합니다.

- 로컬 보조 인덱스: 해시 키는 테이블과 같지만 범위 키는 다른 인덱스입니다.

- 글로벌 보조 인덱스: 해시 및 범위 키가 테이블과 다를 수 있는 인덱스입니다.

테이블당 최대 5개의 글로벌 보조 인덱스 및 5개의 로컬 보조 인덱스를 정의할 수 있습니다. 자세한 내용은 DynamoDB 개발자 안내서의 DynamoDB에서 보조 인덱스를 사용하여 데이터 액세스 향상을 참조하세요.

Query and Scan

항목에 액세스하기 위해 기본 키를 사용하는 방법 외에도 Amazon DynamoDB는 데이터를 검색할 수 있는 두 가지 API인 쿼리와 스캔을 제공합니다. DynamoDB 개발자 안내서에서 쿼리 및 스캔 지침을 읽고 몇 가지 모범 사례를 숙지하는 것이 좋습니다.

쿼리

쿼리 작업은 기본 키 속성 값만 사용하여 테이블 또는 보조 인덱스의 항목을 찾습니다. 따라서 해시 키 속성 이름과 검색할 개별 값은 직접 입력해야 합니다. 범위 키 속성 이름과 값은 옵션으로 입력할 수 있으며, 비교 연산자를 사용하여 검색 결과의 범위를 좁힐 수도 있습니다.

샘플 쿼리는 다음을 참조하세요.

- 문서 모델 사용
- 객체 지속성 모델 사용
- DynamoDB 서비스 수준 API 사용

쿼리에 대한 자세한 내용은 DynamoDB 개발자 안내서의 쿼리를 참조하세요.

스캔

스캔 작업은 테이블 또는 보조 인덱스의 모든 항목을 읽어옵니다. 기본적으로 스캔 작업은 테이블이나 인덱스에 속한 항목의 데이터 속성을 모두 반환합니다. 하지만 스캔 작업에서 ProjectionExpression 파라미터를 사용하면 모두가 아닌 일부 속성만 가져올 수 있습니다.

샘플 스캔은 다음을 참조하세요.

- 문서 모델 사용
- 객체 지속성 모델 사용
- DynamoDB 서비스 수준 API 사용

스캔에 대한 자세한 내용은 DynamoDB 개발자 안내서의 스캔을 참조하세요.

프로젝트 설정

필수 조건

애플리케이션에서 DynamoDB를 사용하려면 프로젝트에 SDK를 추가해야 합니다. 이렇게 하려면 .NET 및 Xamarin용 AWS Mobile SDK 설정의 지침을 따르세요.

DynamoDB 테이블 생성

테이블을 생성하려면 DynamoDB 콘솔로 이동하여 다음 단계를 수행합니다.

1. 테이블 만들기를 클릭합니다.
2. 테이블의 이름을 입력합니다.
3. 기본 키 유형으로 해시를 선택합니다.
4. 형식을 선택하고 해시 속성 이름에 값을 입력합니다. 계속을 클릭합니다.
5. 글로벌 보조 인덱스를 사용하려면 인덱스 추가 페이지에서 인덱스 유형을 "글로벌 보조 인덱스"로 설정하고 인덱스 해시 키 아래에서 보조 인덱스의 값을 입력합니다. 그러면 기본 인덱스와 보조 인덱스를 모두 사용하여 쿼리 및 스캔할 수 있습니다. 테이블에 인덱스 추가를 클릭하고 계속을 클릭합니다. 글로벌 보조 인덱스 사용을 건너뛰려면 계속을 클릭합니다.
6. 읽기 및 쓰기 용량을 원하는 수준으로 설정합니다. 용량 구성에 대한 자세한 내용은 Amazon DynamoDB의 프로비저닝된 처리량을 참조하세요. 계속을 클릭합니다.
7. 다음 화면에서 필요한 경우 처리량 경보를 생성할 알림 이메일을 입력합니다. 계속을 클릭합니다.
8. 요약 페이지에서 생성을 클릭합니다. DynamoDB가 데이터베이스를 생성합니다.

DynamoDB에 대한 권한 설정

애플리케이션에서 DynamoDB를 사용하려면 올바른 권한을 설정해야 합니다. 다음 IAM 정책은 사용자가 ARN으로 정의되는 특정 DynamoDB 테이블 내 항목에 대해 삭제, 가져오기, 내보내기, 쿼리, 스캔 및 업데이트 작업을 수행하도록 허용합니다.

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

```
        "dynamodb:DeleteItem",
        "dynamodb:GetItem",
        "dynamodb:PutItem",
        "dynamodb:Query",
        "dynamodb:Scan",
        "dynamodb:UpdateItem"
      ],
      "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/MyTable"
    }
  ]
}
```

IAM 콘솔에서 정책을 수정할 수 있습니다. 앱의 필요에 따라 허용되는 작업을 추가 또는 제거해야 합니다.

IAM 정책에 대한 자세한 내용은 IAM 사용을 참조하세요.

DynamoDB 고유 정책에 대한 자세한 내용은 DynamoDB 개발자 안내서의 IAM을 사용하여 DynamoDB 리소스에 대한 액세스 제어를 참조하세요.

애플리케이션에 DynamoDB 통합

.NET 및 Xamarin용 AWS Mobile SDK는 DynamoDB에서의 작업을 위한 상위 수준 라이브러리를 제공합니다. 또한 하위 수준 DynamoDB API에 대해 직접 요청을 생성할 수 있지만, 대부분의 사용 사례에서는 상위 수준 라이브러리가 권장됩니다. AmazonDynamoDBClient는 상위 수준 라이브러리에서 특히 유용한 부분입니다. 이 클래스를 사용하여 다양한 생성, 읽기, 업데이트 및 삭제(CRUD) 작업을 수행하고 쿼리를 실행할 수 있습니다.

.NET 및 Xamarin용 AWS Mobile SDK에서는 DynamoDB 작업을 위해 .NET용 AWS SDK에서 API를 사용하여 호출할 수 있습니다. 모든 API는 AWSSDK.dll로 사용할 수 있습니다. .NET용 AWS SDK 다운로드에 대한 자세한 내용은 .NET용 AWS SDK를 참조하세요.

Xamarin 애플리케이션에서 DynamoDB와 상호 작용할 수 있는 방법은 세 가지가 있습니다.

- 문서 모델: 이 API는 하위 수준 DyanmoDB API를 중심으로 래퍼 클래스를 제공하여 프로그래밍 작업을 더욱 간소화합니다. 테이블과 문서가 주요 래퍼 클래스입니다. 문서 모델은 항목 생성, 검색, 업데이트 및 삭제 등의 데이터 작업에 사용됩니다. 이 API는 Amazon.DynamoDB.DocumentModel 네임스페이스에서 사용할 수 있습니다.
- 객체 지속성 모델: 객체 지속성 API를 사용하면 클라이언트 쪽 클래스를 DynamoDB 테이블로 매핑할 수 있습니다. 그러면 각 객체 인스턴스도 해당 테이블 항목으로 매핑됩니다. 이 API의

DynamoDBContext 클래스는 클라이언트 측 객체를 테이블에 저장하거나, 항목을 객체로 가져오거나, 쿼리 및 스캔을 실행할 수 있는 메서드를 제공합니다. 객체 지속성 모델은 항목 생성, 검색, 업데이트 및 삭제 등의 데이터 작업에 사용됩니다. 먼저 서비스 클라이언트 API를 사용하여 테이블을 생성한 다음 객체 지속성 모델을 사용하여 클래스를 테이블로 매핑해야 합니다. 이 API는 Amazon.DynamoDB.DataModel 네임스페이스에서 사용할 수 있습니다.

- 서비스 클라이언트 API: 이 프로토콜 수준 API는 DynamoDB API에 밀접하게 매핑됩니다. 테이블 및 항목 생성부터 업데이트, 삭제에 이르기까지 모든 테이블 및 항목 작업에는 이 하위 수준 API를 사용할 수 있습니다. 또한 테이블에 대한 쿼리나 스캔도 가능합니다. 이 API는 Amazon.DynamoDB 네임스페이스에서 사용할 수 있습니다.

다음 주제에서 세 방법을 자세히 살펴봅니다.

문서 모델 사용

문서 모델은 하위 수준 .NET API를 둘러싼 래퍼 클래스를 제공합니다. 테이블과 문서가 주요 래퍼 클래스입니다. 문서 모델을 사용하여 항목을 생성, 검색, 업데이트 및 삭제할 수 있습니다. 테이블을 생성, 업데이트 및 삭제하려면 하위 수준 API를 사용해야 합니다. 하위 수준 API를 사용하는 방법에 대한 자세한 내용은 DynamoDB 서비스 수준 API 사용을 참조하세요. 이 하위 수준 API는 Amazon.DynamoDB.DocumentModel 네임스페이스에서 사용할 수 있습니다.

문서 모델에 대한 자세한 내용은 .NET 문서 모델을 참조하세요.

DynamoDB 클라이언트 생성

DynamoDB 클라이언트를 생성하는 방법:

```
var client = new AmazonDynamoDBClient(credentials,region);
DynamoDBContext context = new DynamoDBContext(client);
```

CRUD 연산

항목 저장

항목을 생성합니다.

```
Table table = Table.LoadTable(client, "Books");
id = Guid.NewGuid().ToString();
```

```
var books = new Document();
books["Id"] = id;
books["Author"] = "Mark Twain";
books["Title"] = "Adventures of Huckleberry Finn";
books["ISBN"] = "112-111111";
books["Price"] = "10";
```

DynamoDB 테이블에 항목을 저장합니다.

```
var book = await table.PutItemAsync(books);
```

항목 검색

항목을 검색하는 방법:

```
public async Task GetItemAsync(AWSCredentials credentials, RegionEndpoint region)
{
      var client = new AmazonDynamoDBClient(credentials, region);
      Table books = Table.LoadTable(client, "Books");
      var book = await books.GetItemAsync(id);
}
```

항목 업데이트

항목 업데이트 방법:

```
public async Task UpdateItemAttributesAsync(AWSCredentials credentials, RegionEndpoint
 region)
{
      var book = new Document();
      book["Id"] = id;
      book["PageCount"] = "200";
      var client = new AmazonDynamoDBClient(credentials, region);
      Table books = Table.LoadTable(client, "Books");
      Document updatedBook = await books.UpdateItemAsync(book);
}
```

항목을 조건부로 업데이트하는 방법:

```
public async Task UpdateItemConditionallyAsync(AWSCredentials credentials,
 RegionEndpoint region) {
```

```
      var book = new Document();
      book["Id"] = id;
      book["Price"] = "30";

      // For conditional price update, creating a condition expression.
      Expression expr = new Expression();
      expr.ExpressionStatement = "Price = :val";
      expr.ExpressionAttributeValues[":val"] = 10.00;

      var client = new AmazonDynamoDBClient(credentials, region);
      Table books = Table.LoadTable(client, "Books");

      Document updatedBook = await books.UpdateItemAsync(book);
}
```

항목 삭제

항목 삭제 방법:

```
public async Task DeleteItemAsync(AWSCredentials credentials, RegionEndpoint region)
{
  var client = new AmazonDynamoDBClient(credentials, region);
  Table books = Table.LoadTable(client, "Books");
  await books.DeleteItemAsync(id);
}
```

Query and Scan

저자가 "Mark Twain"인 모든 도서를 쿼리하고 가져오는 방법은 다음과 같습니다.

```
public async Task QueryAsync(AWSCredentials credentials, RegionEndpoint region) {
  var client = new AmazonDynamoDBClient(credentials, region);
  Table books = Table.LoadTable(client, "Books");
  var search = books.Query(new QueryOperationConfig() {
   IndexName = "Author-Title-index",
   Filter = new QueryFilter("Author", QueryOperator.Equal, "Mark Twain")
  });
  Console.WriteLine("ScanAsync: printing query response");
  var documents = await search.GetRemainingAsync();
  documents.ForEach((d) = > {
   PrintDocument(d);
  });
```

```
}
```

아래의 스캔 코드 예제는 테이블에서 모든 도서를 반환합니다.

```
public async Task ScanAsync(AWSCredentials credentials, RegionEndpoint region) {
      var client = new AmazonDynamoDBClient(credentials, region);
      Table books = Table.LoadTable(client, "Books");
      var search = books.Scan(new ScanOperationConfig() {
              ConsistentRead = true
      });
      Console.WriteLine("ScanAsync: printing scan response");
      var documents = await search.GetRemainingAsync();
      documents.ForEach((d) = > {
              PrintDocument(d);
      });
}
```

객체 지속성 모델 사용

.NET 및 Xamarin용 AWS Mobile SDK는 객체 지속성 모델을 제공하므로 이것으로 클라이언트 측 클래스를 DynamoDB 테이블에 매핑할 수 있습니다. 그러면 각 객체 인스턴스도 해당 테이블의 항목으로 매핑됩니다. 클라이언트 쪽 객체를 테이블에 저장하기 위해 객체 지속성 모델에서는 DynamoDB에 대한 진입점인 DynamoDBContext 클래스를 제공합니다. 이 클래스는 DynamoDB로 연결하는 역할을 하기 때문에 테이블에 액세스하여 다양한 CRUD 작업이 가능할 뿐만 아니라 쿼리를 실행할 수 있습니다.

객체 지속성 모델은 테이블을 생성, 업데이트 또는 삭제할 수 있는 API를 제공하지 않으며 데이터 작업만 제공합니다. 테이블을 생성, 업데이트 및 삭제하려면 하위 수준 API를 사용해야 합니다. 하위 수준 API를 사용하는 방법에 대한 자세한 내용은 DynamoDB 서비스 수준 API 사용을 참조하세요.

개요

객체 지속성 모델은 속성 세트를 제공하여 클라이언트 측 클래스를 테이블로 매핑할 수 있으며, 속성/필드를 테이블 속성으로 매핑할 수 있습니다. 객체 지속성 모델은 클래스 속성 및 테이블 속성 간 명시적 매핑과 기본 매핑 모두를 지원합니다.

- 명시적 매핑: 속성을 기본 키에 매핑하기 위해서는 DynamoDBHashKey 및 DynamoDBRangeKey 객체 지속성 모델 속성을 사용해야 합니다. 또한, 기본이 아닌 키 속성의 경우, 클래스의 속성 이름과 이를 매핑하려는 해당 테이블 속성이 같지 않다면 DynamoDBProperty 속성을 명시적으로 추가하여 매핑을 정의해야 합니다.

- 기본 매핑 - 기본적으로 객체 지속성 모델은 클래스 속성을 같은 이름의 테이블 속성으로 매핑합니다.

각 클래스 속성을 전부 하나씩 매핑할 필요는 없습니다. DynamoDBIgnore 속성을 추가하여 이러한 속성을 확인할 수 있습니다. 객체의 인스턴스를 저장 및 검색하면 이 속성으로 표시된 속성이 모두 생략됩니다.

지원되는 데이터 형식

객체 지속성 모델은 여러 개의 기본 .NET 데이터 유형, 컬렉션 및 임의 데이터 유형을 지원합니다. 모델이 지원하는 기본 데이터 유형은 다음과 같습니다.

- bool
- 바이트
- char
- DateTime
- decimal, double, float
- Int16, Int32, Int64
- SByte
- 문자열
- UInt16, UInt32, UInt64

객체 지속성 모델은 .NET 컬렉션 형식도 지원하지만 다음과 같은 제한 사항이 있습니다.

- 컬렉션 형식은 ICollection 인터페이스를 구현해야 합니다.
- 컬렉션 형식은 지원되는 기본 유형으로 구성되어야 합니다. 예를 들면 ICollection<string>, ICollection<bool>과 같습니다.
- 컬렉션 형식은 파라미터가 없는 생성자를 사용할 수 있어야 합니다.

객체 지속성 모델에 대한 자세한 내용은 .NET 객체 지속성 모델을 참조하세요.

DynamoDB 클라이언트 생성

DynamoDB 클라이언트를 생성하는 방법:

```
var client = new AmazonDynamoDBClient(credentials,region);
DynamoDBContext context = new DynamoDBContext(client);
```

CRUD 연산

객체 저장

객체를 생성합니다.

```
[DynamoDBTable("Books")]
public class Book {
  [DynamoDBHashKey] // Hash key.
  public string Id {
    get;
    set;
  }

  [DynamoDBGlobalSecondaryIndexHashKey]
  public string Author {
    get;
    set;
  }

  [DynamoDBGlobalSecondaryIndexRangeKey]
  public string Title {
    get;
    set;
  }
  public string ISBN {
    get;
    set;
  }
  public int Price {
    get;
    set;
  }
  public string PageCount {
    get;
    set;
  }
}
```

```
Book myBook = new Book
{
    Id = id,
    Author = "Charles Dickens",
    Title = "Oliver Twist",
    ISBN = "111-1111111001",
    Price = 10,
    PageCount = 300
};
```

DynamoDB 테이블에 객체를 저장합니다.

```
context.Save(myBook);
```

객체 검색

객체를 검색하는 방법:

```
Book retrievedBook = context.Load<Book>(1);
```

객체 업데이트

객체를 업데이트하는 방법:

```
Book retrievedBook = context.Load<Book>(1);
retrievedBook.ISBN = "111-1111111001";
context.Save(retrievedBook);
```

객체 삭제

객체를 삭제하려면:

```
Book retrievedBook = context.Load<Book>(1);
context.Delete(retrievedBook);
```

Query and Scan

저자가 "Charles Dickens"인 모든 도서를 쿼리하고 가져오는 방법은 다음과 같습니다.

```
public async Task QueryAsync(AWSCredentials credentials, RegionEndpoint region) {
```

```
  var client = new AmazonDynamoDBClient(credentials, region);
  DynamoDBContext context = new DynamoDBContext(client);

  var search = context.FromQueryAsync < Book > (new
 Amazon.DynamoDBv2.DocumentModel.QueryOperationConfig() {
    IndexName = "Author-Title-index",
    Filter = new Amazon.DynamoDBv2.DocumentModel.QueryFilter("Author",
 Amazon.DynamoDBv2.DocumentModel.QueryOperator.Equal, "Charles Dickens")
  });

  Console.WriteLine("items retrieved");

  var searchResponse = await search.GetRemainingAsync();
  searchResponse.ForEach((s) = > {
    Console.WriteLine(s.ToString());
  });
}
```

아래의 스캔 코드 예제는 테이블에서 모든 도서를 반환합니다.

```
public async Task ScanAsync(AWSCredentials credentials, RegionEndpoint region) {
  var client = new AmazonDynamoDBClient(credentials, region);
  DynamoDBContext context = new DynamoDBContext(client);

  var search = context.FromScanAsync < Book > (new
 Amazon.DynamoDBv2.DocumentModel.ScanOperationConfig() {
   ConsistentRead = true
  });

  Console.WriteLine("items retrieved");

  var searchResponse = await search.GetRemainingAsync();
  searchResponse.ForEach((s) = > {
   Console.WriteLine(s.ToString());
  });
}
```

DynamoDB 서비스 수준 API 사용

Dynamo 서비스 수준 API를 사용하면 테이블을 생성하고, 업데이트하고, 삭제할 수 있습니다. 또한 이 API를 사용하여 테이블의 항목에 대한 일반적인 생성, 읽기, 업데이트 및 삭제(CRUD) 작업도 수행할 수 있습니다.

DynamoDB 클라이언트 생성

DynamoDB 클라이언트를 생성하는 방법:

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient(credentials,region);
```

CRUD 연산

항목 저장

DynamoDB 테이블에 항목을 저장하는 방법:

```
// Create a client
AmazonDynamoDBClient client = new AmazonDynamoDBClient(credentials,region);

// Define item attributes
Dictionary<string, AttributeValue> attributes = new Dictionary<string,
 AttributeValue>();

// Author is hash-key
attributes["Author"] = new AttributeValue { S = "Mark Twain" };
attributes["Title"] = new AttributeValue { S = "The Adventures of Tom Sawyer" };
attributes["PageCount"] = new AttributeValue { N = "275" };
attributes["Price"] = new AttributeValue{N = "10.00"};
attributes["Id"] = new AttributeValue{N="10"};
attributes["ISBN"] = new AttributeValue{S="111-1111111"};

// Create PutItem request
PutItemRequest request = new PutItemRequest
{
    TableName = "Books",
    Item = attributes
};

// Issue PutItem request
var response = await client.PutItemAsync(request);
```

항목 검색

항목을 검색하는 방법:

```
// Create a client
```

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient(credentials,region);

Dictionary<string, AttributeValue> key = new Dictionary<string, AttributeValue>
{
    { "Id", new AttributeValue { N = "10" } }
};

// Create GetItem request
GetItemRequest request = new GetItemRequest
{
    TableName = "Books",
    Key = key,
};

// Issue request
var result = await client.GetItemAsync(request);

// View response
Console.WriteLine("Item:");
Dictionary<string, AttributeValue> item = result.Item;
foreach (var keyValuePair in item)
{
    Console.WriteLine("Author := {0}", item["Author"]);
    Console.WriteLine("Title := {0}", item["Title"]);
    Console.WriteLine("Price:= {0}", item["Price"]);
    Console.WriteLine("PageCount := {0}", item["PageCount"]);
}
```

항목 업데이트

항목 업데이트 방법:

```
// Create a client
AmazonDynamoDBClient client = new AmazonDynamoDBClient(credentials,region);

Dictionary<string, AttributeValue> key = new Dictionary<string, AttributeValue>
{
    { "Id", new AttributeValue { N = "10" } }
};

// Define attribute updates
Dictionary<string, AttributeValueUpdate> updates = new Dictionary<string,
 AttributeValueUpdate>();
```

```
// Add a new string to the item's Genres SS attribute
updates["Genres"] = new AttributeValueUpdate()
{
    Action = AttributeAction.ADD,
    Value = new AttributeValue { SS = new List<string> { "Bildungsroman" } }
};

// Create UpdateItem request
UpdateItemRequest request = new UpdateItemRequest
{
    TableName = "Books",
    Key = key,
    AttributeUpdates = updates
};

// Issue request
var response = await client.UpdateItemAsync(request);
```

항목 삭제

항목 삭제 방법:

```
// Create a client
AmazonDynamoDBClient client = new AmazonDynamoDBClient(credentials,region);

Dictionary<string, AttributeValue> key = new Dictionary<string, AttributeValue>
{
  { "Id", new AttributeValue { N = "10" } }
};

// Create DeleteItem request
DeleteItemRequest request = new DeleteItemRequest
{
  TableName = "Books",
  Key = key
};

// Issue request
var response = await client.DeleteItemAsync(request);
```

Query and Scan

저자가 "Mark Twain"인 모든 도서를 쿼리하고 가져오는 방법은 다음과 같습니다.

```
public void Query(AWSCredentials credentials, RegionEndpoint region) {
  using(var client = new AmazonDynamoDBClient(credentials, region)) {
    var queryResponse = await client.QueryAsync(new QueryRequest() {
      TableName = "Books",
      IndexName = "Author-Title-index",
      KeyConditionExpression = "Author = :v_Id",
      ExpressionAttributeValues = new Dictionary < string, AttributeValue > {
        {
          ":v_Id", new AttributeValue {
            S = "Mark Twain"
          }
        }
      }
    });
    queryResponse.Items.ForEach((i) = > {
      Console.WriteLine(i["Title"].S);
    });

  }
}
```

아래의 스캔 코드 예제는 테이블에서 모든 도서를 반환합니다.

```
public void Scan(AWSCredentials credentials, RegionEndpoint region) {
      using(var client = new AmazonDynamoDBClient(credentials, region)) {
              var queryResponse = client.Scan(new ScanRequest() {
                      TableName = "Books"
              });
              queryResponse.Items.ForEach((i) = > {
                      Console.WriteLine(i["Title"].S);
              });
      }
}
```

Amazon Simple Notification Service(SNS)

SNS와 AWS Mobile SDK를 사용하여 모바일 푸시 알림을 수신할 수 있는 애플리케이션을 작성할 수 있습니다. SNS에 대한 자세한 내용은 Amazon Simple Notification Service를 참조하세요.

핵심 개념

Amazon SNS를 이용하면 서로 다른 디바이스의 애플리케이션과 최종 사용자가 모바일 푸시 알림(Apple, Google 및 Kindle Fire 디바이스), HTTP/HTTPS, Email/Email-JSON, SMS 또는 Amazon Simple Queue Service(SQS) 대기열, AWS Lambda 함수 등을 통해 알림을 수신할 수 있습니다. SNS를 사용하여 개별 메시지를 전송하거나 단일 주제를 구독하는 대규모의 수신자에게 메시지를 전송할 수 있습니다.

주제

주제는 수신자가 동일한 알림의 동일한 사본을 동적으로 구독할 수 있는 "액세스 지점"입니다. 하나의 주제가 여러 엔드포인트 유형으로의 전송을 지원합니다. 예를 들어, iOS, Android 및 SMS 수신자를 그룹화할 수 있습니다.

구독

주제에 게시된 메시지를 받으려면 해당 주제를 구독하도록 엔드포인트를 등록해야 합니다. 엔드포인트는 Amazon SNS로부터 알림 메시지를 수신할 수 있는 모바일 앱, 웹 서버, 이메일 주소 또는 Amazon SQS 대기열입니다. 주제를 구독하도록 엔드포인트를 등록하고 구독이 확인되면, 엔드포인트는 주제에 게시된 모든 메시지를 받게 됩니다.

게시

주제에 게시하면 SNS가 해당 주제의 각 구독자에게 적절히 서식이 지정된 메시지의 사본을 전달합니다. 모바일 푸시 알림의 경우 엔드포인트에 직접 게시하거나 엔드포인트가 주제를 구독하게 등록할 수 있습니다.

프로젝트 설정

필수 조건

애플리케이션에서 SNS를 사용하려면 프로젝트에 SDK를 추가해야 합니다. 이렇게 하려면 AWS Mobile SDK for .NET and Xamarin 설정의 지침을 따르세요.

SNS에 대한 권한 설정

SNS에 대한 권한을 설정하는 방법에 대한 자세한 내용은 Amazon SNS 주제에 대한 액세스 관리를 참조하세요.

프로젝트에 SNS용 NuGet 패키지 추가

AWS Mobile SDK for .NET and Xamarin 설정 내 지침의 4단계를 따라 Amazon Simple Notification Service NuGet 패키지를 프로젝트에 추가합니다.

애플리케이션에 SNS 통합

Xamarin 애플리케이션에서 SNS와 상호 작용하는 방법은 여러 가지가 있습니다.

푸시 알림 전송(Xamarin Android)

이 문서에서는 Amazon Simple Notification Service(SNS)와 .NET 및 Xamarin용 AWS Mobile SDK를 사용하여 Xamarin Android 애플리케이션으로 푸시 알림을 전송하는 방법을 설명합니다.

프로젝트 설정

필수 조건

이 자습서를 시작하기 전에 .NET 및 Xamarin용 AWS Mobile SDK 설정의 지침을 모두 완료해야 합니다.

SNS에 대한 권한 설정

.NET 및 Xamarin용 AWS Mobile SDK 설정의 2단계를 따라 아래에서 언급하는 정책을 애플리케이션의 역할에 연결합니다. 그러면 애플리케이션이 SNS애 액세스할 수 있는 적절한 권한을 부여 받습니다.

1. IAM 콘솔로 이동하여 구성할 IAM 역할을 선택합니다.
2. 정책 연결을 클릭하고 AmazonSNSFullAccess 정책을 선택한 다음 정책 연결을 클릭합니다.

> **Warning**
>
> AmazonSNSFullAccess를 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 여기서는 빠르게 실행할 수 있도록 사용하는 것입니다. IAM 역할 권한 지정에 대한 자세한 내용은 IAM 역할 권한 개요를 참조하세요.

Google Cloud에서 푸시 알림 활성화

먼저 새 Google API 프로젝트를 추가합니다.

1. Google 개발자 콘솔로 이동합니다.
2. 프로젝트 생성을 클릭합니다.
3. 새 프로젝트 상자에 프로젝트 이름을 입력하고 프로젝트 ID(나중에 필요)를 적어 둔 다음 생성을 클릭합니다.

그런 다음 프로젝트에서 Google Cloud Messaging(GCM) 서비스를 활성화합니다.

1. Google 개발자 콘솔에는 새 프로젝트가 이미 선택되어 있을 것입니다. 그렇지 않으면 페이지 상단에 있는 드롭다운 목록에서 선택합니다.
2. 페이지 왼쪽의 사이드바에서 APIs & auth(API 및 인증)를 선택합니다.
3. 검색 상자에서 "Google Cloud Messaging for Android"를 입력하고 Google Cloud Messaging for Android 링크를 클릭합니다.
4. API 활성화를 클릭합니다.

마지막으로 API 키를 받습니다.

1. Google 개발자 콘솔에서 APIs & auth(API 및 인증) > Credentials(자격 증명)을 선택합니다.
2. Public API access(퍼블릭 API 액세스) 아래에서 Create new key(새 키 생성)를 클릭합니다.
3. Create a new key(새 키 생성) 대화 상자에서 Server key(서버 키)를 클릭합니다.
4. 표시되는 대화 상자에서 Create(생성)을 클릭하고 표시된 API 키를 복사합니다. 나중에 이 API 키를 사용해 인증을 수행합니다.

SNS 콘솔에서 프로젝트 ID를 사용해 플랫폼 ARN 생성

1. SNS 콘솔로 이동합니다.
2. 화면 왼쪽에 있는 애플리케이션을 클릭합니다.
3. 플랫폼 애플리케이션 생성을 클릭하여 새 SNS 플랫폼 애플리케이션을 생성합니다.
4. 애플리케이션 이름을 입력합니다.
5. 푸시 알림 플랫폼으로 Google 클라우드 메시징(GCM)을 선택합니다.
6. API 키를 API 키로 표시된 텍스트 상자에 붙여 넣습니다.

7. 플랫폼 애플리케이션 생성을 클릭합니다.

8. 방금 생성한 플랫폼 애플리케이션을 선택하고 애플리케이션 ARN을 복사합니다.

프로젝트에 SNS용 NuGet 패키지 추가

.NET 및 Xamarin용 AWS Mobile SDK 설정 내 지침의 4단계를 따라 Amazon Simple Notification Service NuGet 패키지를 프로젝트에 추가합니다.

SNS 클라이언트 생성

```
var snsClient = new AmazonSimpleNotificationServiceClient(credentials, region);
```

원격 알림에 애플리케이션 등록

Android에서 원격 알림에 등록하려면 Google Cloud 메시지를 수신할 수 있는 BroadcastReceiver를 생성해야 합니다. 아래에서 해당 메시지가 표시된 위치의 패키지 이름을 변경합니다.

```
[BroadcastReceiver(Permission = "com.google.android.c2dm.permission.SEND")]
[IntentFilter(new string[] {
      "com.google.android.c2dm.intent.RECEIVE"
}, Categories = new string[] {
      "com.amazonaws.sns" /* change to match your package */
})]
[IntentFilter(new string[] {
      "com.google.android.c2dm.intent.REGISTRATION"
}, Categories = new string[] {
      "com.amazonaws.sns" /* change to match your package */
})]
[IntentFilter(new string[] {
      "com.google.android.gcm.intent.RETRY"
}, Categories = new string[] {
      "com.amazonaws.sns" /* change to match your package */
})]
public class GCMBroadcastReceiver: BroadcastReceiver {
      const string TAG = "PushHandlerBroadcastReceiver";
      public override void OnReceive(Context context, Intent intent) {
              GCMIntentService.RunIntentInService(context, intent);
              SetResult(Result.Ok, null, null);
      }
}
```

```
[BroadcastReceiver]
[IntentFilter(new[] {
      Android.Content.Intent.ActionBootCompleted
})]
public class GCMBootReceiver: BroadcastReceiver {
      public override void OnReceive(Context context, Intent intent) {
              GCMIntentService.RunIntentInService(context, intent);
              SetResult(Result.Ok, null, null);
      }
}
```

다음은 BroadcastReceiver로부터 푸시 알림을 수신하여 디바이스의 알림 표시줄에 표시하는 서비스입니다.

```
[Service]
 public class GCMIntentService: IntentService {
  static PowerManager.WakeLock sWakeLock;
  static object LOCK = new object();

  public static void RunIntentInService(Context context, Intent intent) {
    lock(LOCK) {
      if (sWakeLock == null) {
        // This is called from BroadcastReceiver, there is no init.
        var pm = PowerManager.FromContext(context);
        sWakeLock = pm.NewWakeLock(
        WakeLockFlags.Partial, "My WakeLock Tag");
      }
    }

    sWakeLock.Acquire();
    intent.SetClass(context, typeof(GCMIntentService));
    context.StartService(intent);
  }

  protected override void OnHandleIntent(Intent intent) {
    try {
      Context context = this.ApplicationContext;
      string action = intent.Action;

      if (action.Equals("com.google.android.c2dm.intent.REGISTRATION")) {
        HandleRegistration(intent);
      } else if (action.Equals("com.google.android.c2dm.intent.RECEIVE")) {
```

```
        HandleMessage(intent);
      }
    } finally {
      lock(LOCK) {
        //Sanity check for null as this is a public method
        if (sWakeLock != null) sWakeLock.Release();
      }
    }
  }

  private void HandleRegistration(Intent intent) {
    string registrationId = intent.GetStringExtra("registration_id");
    string error = intent.GetStringExtra("error");
    string unregistration = intent.GetStringExtra("unregistered");

    if (string.IsNullOrEmpty(error)) {
      var response = await SnsClient.CreatePlatformEndpointAsync(new
CreatePlatformEndpointRequest {
        Token = registrationId,
        PlatformApplicationArn = "YourPlatformArn" /* insert your platform application
ARN here */
      });
    }
  }

  private void HandleMessage(Intent intent) {
    string message = string.Empty;
    Bundle extras = intent.Extras;
    if (!string.IsNullOrEmpty(extras.GetString("message"))) {
      message = extras.GetString("message");
    } else {
      message = extras.GetString("default");
    }

    Log.Info("Messages", "message received = " + message);
    ShowNotification(this, "SNS Push", message);
    //show the message

  }

  public void ShowNotification(string contentTitle,
  string contentText) {
    // Intent
    Notification.Builder builder = new Notification.Builder(this)
```

```
        .SetContentTitle(contentTitle)
        .SetContentText(contentText)
        .SetDefaults(NotificationDefaults.Sound | NotificationDefaults.Vibrate)
        .SetSmallIcon(Resource.Drawable.Icon)
        .SetSound(RingtoneManager.GetDefaultUri(RingtoneType.Notification));

    // Get the notification manager:
    NotificationManager notificationManager =
 this.GetSystemService(Context.NotificationService) as NotificationManager;

    notificationManager.Notify(1001, builder.Build());
  }
}
```

SNS 콘솔에서 엔드포인트로 메시지 전송

1. SNS 콘솔 > 애플리케이션으로 이동합니다.
2. 플랫폼 애플리케이션을 선택하고 엔드포인트를 선택한 다음 엔드포인트에 게시를 클릭합니다.
3. 텍스트 상자에 텍스트 메시지를 입력하고 메시지 게시를 클릭하여 메시지를 게시합니다.

푸시 알림 전송(Xamarin iOS)

이 문서에서는 Amazon Simple Notification Service(SNS)와 .NET 및 Xamarin용 AWS Mobile SDK를 사용하여 Xamarin iOS 애플리케이션으로 푸시 알림을 전송하는 방법을 설명합니다.

프로젝트 설정

필수 조건

이 자습서를 시작하기 전에 .NET 및 Xamarin용 AWS Mobile SDK 설정의 지침을 모두 완료해야 합니다.

SNS에 대한 권한 설정

.NET 및 Xamarin용 AWS Mobile SDK 설정의 2단계를 따라 아래에서 언급하는 정책을 애플리케이션의 역할에 연결합니다. 그러면 애플리케이션이 SNS에 액세스할 수 있는 적절한 권한을 부여 받습니다.

1. IAM 콘솔로 이동하여 구성할 IAM 역할을 선택합니다.
2. 정책 연결을 클릭하고 AmazonSNSFullAccess 정책을 선택한 다음 정책 연결을 클릭합니다.

> **Warning**
>
> AmazonSNSFullAccess를 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 여기서는 빠르게 실행할 수 있도록 사용하는 것입니다. IAM 역할 권한 지정에 대한 자세한 내용은 IAM 역할 권한 개요를 참조하세요.

Apple iOS 개발자 프로그램 멤버십 받기

푸시 알림을 수신하려면 물리적 디바이스에서 앱을 실행해야 합니다. 디바이스에서 앱을 실행하려면 Apple iOS 개발자 프로그램 멤버십이 있어야 합니다. 멤버십이 있으면 Xcode를 사용하여 서명 자격 증명을 생성할 수 있습니다. 자세한 내용은 Apple의 App Distribution Quick Start 설명서를 참조하세요.

iOS 인증서 생성

먼저 iOS 인증서를 생성해야 합니다. 그런 다음 푸시 알림용으로 구성된 프로비저닝 프로파일을 생성해야 합니다. 그렇게 하려면 다음을 수행하세요.

1. Apple Developer Member Center로 이동하여 인증서, ID, 프로필을 클릭합니다.
2. iOS 앱에서 식별자를 클릭하고 웹 페이지 오른쪽 상단의 더하기 버튼을 클릭하여 새 iOS 앱 ID를 추가한 다음 앱 ID 설명을 입력합니다.
3. 아래로 스크롤하여 Add ID Suffix(ID 접미사 추가) 섹션에서 Explicit App ID(명시적 앱 ID)를 선택하고 번들 식별자를 입력합니다.
4. 아래로 스크롤하여 App Services(앱 서비스) 섹션에서 푸시 알림을 선택합니다.
5. 계속을 클릭합니다.
6. Submit을 클릭합니다.
7. 완료를 클릭합니다.
8. 방금 생성한 앱 ID를 선택하고 편집을 클릭합니다.
9. 아래로 스크롤하여 푸시 알림 섹션을 찾습니다. 개발 SSL 인증서 아래에서 인증서 생성을 클릭합니다.
10. 지침을 따라 인증서 서명 요청(CSR)을 생성하고 요청을 업로드한 다음 Apple 알림 서비스(APNS)와 통신하는 데 사용될 SSL 인증서를 다운로드합니다.
11. Certificates, Identifiers & Profiles(인증서, ID, 프로필) 페이지로 돌아갑니다. 프로비저닝 프로파일 아래의 모두를 클릭합니다.
12. 오른쪽 위 모서리에 있는 더하기 버튼을 클릭하여 새 프로비저닝 프로파일을 추가합니다.

13iOS 앱 개발을 선택하고 계속을 클릭합니다.

14.앱 ID를 선택하고 계속을 클릭합니다.

15.개발자 인증서를 선택하고 계속을 클릭합니다.

16.디바이스를 선택하고 계속을 클릭합니다.

17.프로필 이름을 입력하고 생성을 클릭합니다.

18.프로비전 파일을 다운로드하고 두 번 클릭하여 프로비저닝 프로파일을 설치합니다.

푸시 알림용으로 구성된 프로파일을 프로비저닝하는 데 대한 자세한 내용은 Apple의 Configuring Push Notifications 설명서를 참조하세요.

SNS 콘솔에서 인증서를 사용해 플랫폼 ARN 생성

1. KeyChain 액세스 앱을 실행하고 화면의 왼쪽 아래에서 내 인증서를 선택합니다. 그런 다음 앞서 생성한 SSL 인증서를 마우스 오른쪽 버튼으로 클릭하여 APNS에 연결하고 내보내기를 선택합니다. 파일의 이름과 인증서를 보호할 암호를 지정하라는 메시지가 표시됩니다. 인증서는 P12 파일로 저장됩니다.
2. SNS 콘솔로 이동하여 화면 왼쪽에서 애플리케이션을 클릭합니다.
3. 플랫폼 애플리케이션 생성을 클릭하여 새 SNS 플랫폼 애플리케이션을 생성합니다.
4. 애플리케이션 이름을 입력합니다.
5. 푸시 알림 플랫폼으로 Apple Development를 선택합니다.
6. 파일 선택을 선택하고 SSL 인증서를 내보낼 때 생성한 P12 파일을 선택합니다.
7. SSL 인증서를 내보낼 때 지정한 암호를 입력하고 파일에서 자격 증명 로드를 클릭합니다.
8. 플랫폼 애플리케이션 생성을 클릭합니다.
9. 방금 생성한 플랫폼 애플리케이션을 선택하고 애플리케이션 ARN을 복사합니다. 나중 단계에서 이 ARN이 필요합니다.

프로젝트에 SNS용 NuGet 패키지 추가

.NET 및 Xamarin용 AWS Mobile SDK 설정 내 지침의 4단계를 따라 Amazon Simple Notification Service NuGet 패키지를 프로젝트에 추가합니다.

SNS 클라이언트 생성

```
var snsClient = new AmazonSimpleNotificationServiceClient(credentials, region);
```

원격 알림에 애플리케이션 등록

애플리케이션을 등록하려면 아래와 같이 UIApplication 객체에서 RegisterForRemoteNotifications를 호출합니다. AppDelegate.cs에 다음 코드를 배치합니다(아래에서 메시지가 표시된 위치에 플랫폼 애플리케이션 ARN을 삽입).

```
public override bool FinishedLaunching(UIApplication app, NSDictionary options) {
// do something
var pushSettings = UIUserNotificationSettings.GetSettingsForTypes (
  UIUserNotificationType.Alert |
  UIUserNotificationType.Badge |
  UIUserNotificationType.Sound,
  null
);
app.RegisterUserNotifications(pushSettings);
app.RegisterForRemoteNotifications();
// do something
  return true;
}

public override void RegisteredForRemoteNotifications(UIApplication application, NSData
 token) {
  var deviceToken = token.Description.Replace("<", "").Replace(">", "").Replace(" ",
 "");
  if (!string.IsNullOrEmpty(deviceToken)) {
    //register with SNS to create an endpoint ARN
    var response = await SnsClient.CreatePlatformEndpointAsync(
    new CreatePlatformEndpointRequest {
      Token = deviceToken,
      PlatformApplicationArn = "YourPlatformArn" /* insert your platform application
 ARN here */
    });
  }
}
```

SNS 콘솔에서 엔드포인트로 메시지 전송

1. SNS 콘솔 > 애플리케이션으로 이동합니다.
2. 플랫폼 애플리케이션을 선택하고 엔드포인트를 선택한 다음 엔드포인트에 게시를 클릭합니다.
3. 텍스트 상자에 텍스트 메시지를 입력하고 메시지 게시를 클릭하여 메시지를 게시합니다.

SMS 알림 송수신

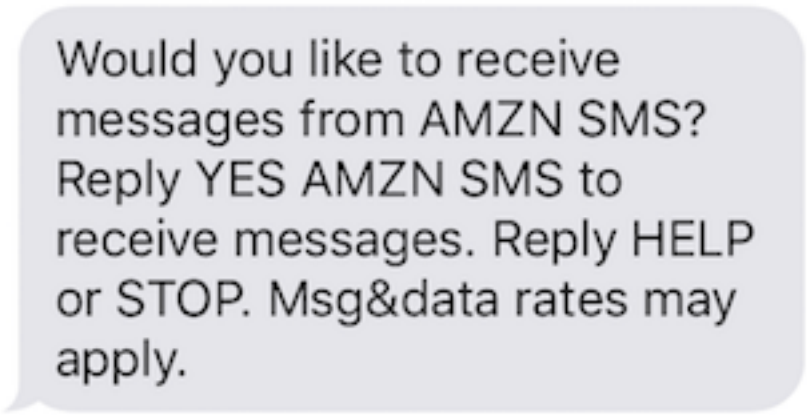
Amazon Simple Notification Service (Amazon SNS)를 사용하여 SMS 사용 가능한 휴대전화 및 스마트폰에 SMS(문자 서비스) 알림을 송수신할 수 있습니다.

> **Note**
>
> SMS 알림은 현재 미국 내 전화번호를 지원합니다. SMS 메시지는 미국 동부(버지니아 북부) 리전에서 생성된 주제에서만 전송할 수 있습니다. 하지만 미국 동부(버지니아 북부) 리전에서 생성한 주제에 대한 메시지는 어느 리전에서든 게시할 수 있습니다.

주제 생성

주제를 생성하려면 다음과 같이 합니다.

1. Amazon SNS 콘솔에서 새 주제 생성을 클릭합니다. [Create new topic] 대화 상자가 나타납니다.
2. [Topic name] 상자에 주제 이름을 입력합니다.
3. [Display name] 상자에 디스플레이 이름을 입력합니다. 표시 이름의 첫 10 문자가 문자 메시지 접두사의 앞부분으로 사용되므로 주제는 지정된 표시 이름을 보유해야 합니다. 여기에 입력하는 디스플레이 이름은 SNS가 사용자에게 전송하는 확인 메시지에 표시됩니다(아래의 디스플레이 이름은 "AMZN SMS"임).



1. 주제 생성을 클릭합니다. [Topics] 페이지에 새 주제가 나타납니다.
2. 새 주제를 선택한 다음 주제 ARN을 클릭합니다. [Topic Details] 페이지가 나타납니다.

3. 다음 단계에서 주제를 구독할 때 필요하므로 주제 ARN을 복사합니다.

```
arn:aws:sns:us-west-2:111122223333:MyTopic
```

SMS 프로토콜을 이용한 주제 구독

SNS 클라이언트를 생성하여 자격 증명 풀의 자격 증명 객체 및 리전을 전달합니다.

```
var snsClient = new AmazonSimpleNotificationServiceClient(credentials, region);
```

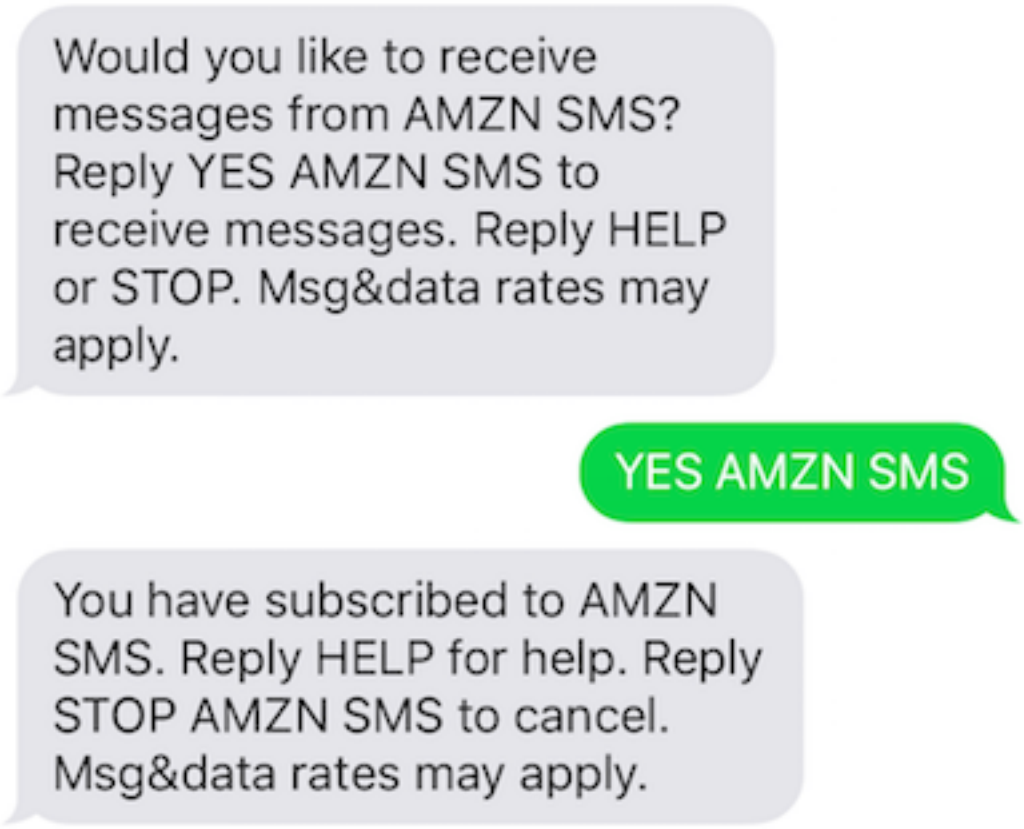
주제를 구독하려면 SubscribeAsync를 호출하여 구독하려는 주제의 ARN, 프로토콜("sms") 및 전화 번호를 전달합니다.

```
var response = await snsClient.SubscribeAsync(topicArn, "sms", "1234567890");
```

구독 응답 객체에서 구독 ARN을 수신합니다. 구독 ARN은 다음과 비슷합니다.

```
arn:aws:sns:us-west-2:123456789012:MyTopic:6b0e71bd-7e97-4d97-80ce-4a0994e55286
```

디바이스가 주제를 구독하면 SNS가 해당 디바이스로 확인 메시지를 전송하며 사용자는 아래와 같이 알림을 수신하기 원함을 확인해야 합니다.

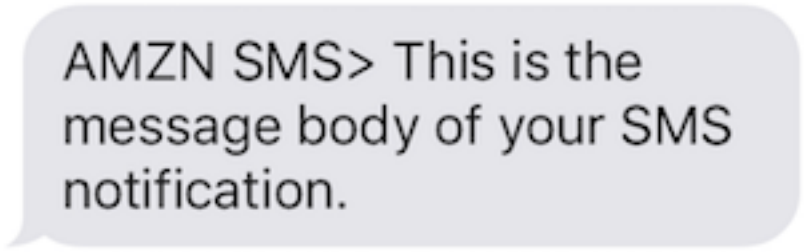


사용자가 주제를 구독한 후 해당 주제에 게시할 경우 사용자가 SMS 메시지를 수신합니다.

메시지 게시

주제에 대한 메시지 게시:

1. AWS Management Console에 로그인한 후 Amazon SNS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 주제를 클릭한 다음 게시할 주제를 선택합니다.
3. 주제 게시를 클릭합니다.
4. [Subject] 상자에 제목을 입력합니다.
5. [Message] 상자에 메시지를 입력합니다. Amazon SNS는 [Message] 상자에 입력한 문자를 SMS 구독자들에게 전송합니다([Subject] 상자에 문자를 입력하지 않을 경우). 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.
6. 메시지 게시를 클릭합니다. Amazon SNS는 확인 대화 상자를 표시합니다. 아래 그림과 같이 SMS 수신 가능한 디바이스에 SMS 메시지가 표시됩니다.



HTTP/HTTPS 엔드포인트로 메시지 전송

Amazon SNS를 사용하여 하나 이상의 HTTP 또는 HTTPS 엔드포인트에 알림 메시지를 전송할 수 있습니다. 프로세스는 다음과 같습니다.

1. 엔드포인트가 Amazon SNS 메시지를 수신하도록 구성합니다.
2. HTTP/HTTPS 엔드포인트가 주제를 구독하게 등록합니다.
3. 구독을 확인합니다.
4. 주제에 알림을 게시합니다. 그러면 Amazon SNS가 HTTP POST 요청을 전송하여 알림의 내용을 구독 엔드포인트로 전달합니다.

HTTP/HTTPS 엔드포인트가 Amazon SNS 메시지를 수신하도록 구성

HTTP/HTTPS 엔드포인트로 SNS 메시지 전송의 1단계 지침을 따라 엔드포인트를 구성합니다.

HTTP/HTTPS 엔드포인트가 Amazon SNS 주제를 구독하게 등록

SNS 클라이언트를 생성하여 자격 증명 풀의 자격 증명 객체 및 리전을 전달합니다.

```
var snsClient = new AmazonSimpleNotificationServiceClient(credentials, region);
```

주제를 통해 메시지를 HTTP 엔드포인트 또는 HTTPS 엔드포인트에 전송하려면 엔드포인트가 Amazon SNS 주제를 구독하게 등록해야 합니다. 사용자는 해당 URL을 사용하여 엔드포인트를 지정합니다:

```
var response = await snsClient.SubscribeAsync(
  "topicArn",
  "http",  /* "http" or "https" */
  "endpointUrl" /* endpoint url beginning with http or https */
```

```
);
```

구독 확인

엔드포인트가 구독을 등록한 후 Amazon SNS가 엔드포인트에 구독 확인 메시지를 전송합니다. 엔드포인트의 코드는 구독 확인 메시지로부터 SubscribeURL 값을 검색하여 SubscribeURL 자체에 의해 지정된 위치를 방문하거나 사용자에게 제공하여 사용자가 수동으로 SubscribeURL을 방문할 수 있도록 해야 합니다(예: 웹 브라우저를 사용하는 경우).

Amazon SNS는 구독이 확인되기 전에는 엔드포인트에 메시지를 보내지 않습니다. SubscribeURL 방문 시, 응답은 구독에 대한 ARN을 지정하는 요소 SubscriptionArn를 담고 있는 XML 문서를 포함합니다.

HTTP/HTTPS 엔드포인트로 메시지 전송

주제를 게시하면 주제의 구독에 메시지를 전송할 수 있습니다. PublishAsync를 호출하여 여기에 주제 ARN 및 메시지를 전달합니다.

```
var response = await snsClient.PublishAsync(topicArn, "This is your message");
```

SNS 문제 해결

Amazon SNS 콘솔에서 전달 상태 사용

Amazon SNS 콘솔에는 모바일 푸시 알림 플랫폼(Apple(APNS), Google(GCM), Amazon(ADM), Windows(WNS 및 MPNS) 및 Baidu)에 대한 메시지 전달 시도가 성공 또는 실패했는지에 대한 피드백을 수집할 수 있는 전달 상태 기능이 포함되어 있습니다.

이 기능은 Amazon SNS 내 유지 시간과 같은 다른 중요한 정보도 제공합니다. 이 정보는 Amazon SNS 콘솔에서 또는 Amazon SNS API를 통해 이 기능이 활성화될 때 Amazon SNS에 의해 자동으로 생성되는 Amazon CloudWatch Logs 그룹에 캡처됩니다.

전달 상태 기능 사용에 대한 지침은 AWS 모바일 블로그에서 Using the Delivery Status feature of Amazon SNS를 참조하세요.

.NET 및 Xamarin용 AWS Mobile SDK 사용 모범 사례

.NET 및 Xamarin용 AWS 모바일 SDK를 사용할 때 알아두면 도움이 되는 몇 가지 기본 사항과 모범 사례가 있습니다.

- 애플리케이션에서 자격 증명을 하드코딩하는 대신 Amazon Cognito를 사용하여 AWS 자격 증명을 얻습니다. 애플리케이션에서 자격 증명을 하드코딩할 경우 자격 증명이 외부로 노출되어 AWS를 호출하는 데 도용될 수 있습니다. Amazon Cognito를 사용하여 AWS 자격 증명을 얻는 방법에 대한 자세한 내용은 .NET 및 Xamarin용 AWS Mobile SDK 설정을 참조하세요.
- S3 사용 모범 사례는 AWS 블로그의 이 기사를 참조하세요.
- DynamoDB 사용 모범 사례는 DynamoDB 개발자 안내서의 DynamoDB 모범 사례를 참조하세요.

AWS는 늘 고객의 성공을 지원하기 위해 노력하고 있으며 언제든지 피드백을 환영합니다. 자유롭게 AWS 포럼에 게시하거나 Github에서 문제를 개설해 주세요.

AWS 서비스 설명서 목록

.NET 및 Xamarin용 AWS Mobile SDK의 모든 서비스에는 유용한 추가 정보가 담긴 별도의 개발자 안내서 및 서비스 API 참조가 있습니다.

Amazon Cognito 자격 증명

- Cognito 개발자 안내서
- Cognito Identity 서비스 API 참조

Amazon Cognito Sync

- Cognito 개발자 안내서
- Cognito Sync 서비스 API 참조

Amazon Mobile Analytics

- Mobile Analytics 개발자 안내서
- Mobile Analytics 서비스 API 참조

Amazon S3

- S3 개발자 안내서
- S3 시작 안내서
- S3 서비스 API 참조

Amazon DynamoDB

- DynamoDB 개발자 안내서
- DynamoDB 시작 안내서
- DynamoDB 서비스 API 참조

Amazon Simply Notification Service(SNS)

- SNS 개발자 안내서
- SNS 서비스 API 참조

기타 유용한 링크

- AWS 용어집
- AWS 자격 증명

문제 해결

이 주제에서는 .NET 및 Xamarin용 AWS Mobile SDK를 사용할 때 발생할 수 있는 문제를 해결하기 위한 몇 가지 아이디어를 설명합니다.

IAM 역할에 필요한 권한이 있는지 확인

AWS 서비스를 호출할 때 앱은 Cognito 자격 증명 풀의 자격 증명을 사용해야 합니다. 풀의 각 자격 증명은 IAM(자격 증명 및 액세스 관리) 역할과 연결되어 있습니다.

역할은 해당 역할에 할당된 사용자가 액세스할 수 있는 AWS 리소스를 지정하는 하나 이상의 정책 파일이 연결되어 있습니다. 기본적으로 자격 증명 풀당 인증된 사용자 및 인증되지 않은 사용자에 대해 하나씩 2개의 역할이 생성됩니다.

기존 정책 파일을 수정하거나 새 정책 파일을 앱이 요구하는 권한과 연결해야 합니다. 앱이 인증된 사용자와 인증되지 않은 사용자를 모두 허용할 경우 두 역할 모두 앱에서 필요한 AWS 리소스에 액세스할 수 있는 권한이 부여되어야 합니다.

다음 정책 파일은 S3 버킷에 대한 액세스 권한을 부여하는 방법을 보여줍니다.

```
{
   "Statement":[
      {
         "Action":[
            "s3:AbortMultipartUpload",
            "s3:DeleteObject",
            "s3:GetObject",
            "s3:PutObject"
         ],
         "Effect":"Allow",
         "Resource":"arn:aws:s3:::MYBUCKETNAME/*",
         "Principal":"*"
      }
   ]
}
```

다음 정책 파일은 DynamoDB 데이터베이스에 대한 액세스 권한을 부여하는 방법을 보여줍니다.

```
{
   "Statement":[
```

```
      {
         "Effect":"Allow",
         "Action":[
            "dynamodb:DeleteItem",
            "dynamodb:GetItem",
            "dynamodb:PutItem",
            "dynamodb:Scan",
            "dynamodb:UpdateItem"
         ],
         "Resource":"arn:aws:dynamodb:us-west-2:123456789012:table/MyTable"
      }
   ]
}
```

정책 지정에 대한 자세한 내용은 IAM 정책을 참조하세요.

HTTP 프록시 디버거 사용

앱이 호출하는 AWS 서비스에 HTTP 또는 HTTPS 엔드포인트가 있는 경우 HTTP/HTTPS 프록시 디버거를 사용하여 요청 및 응답을 보며 상황을 파악할 수 있습니다. 다음과 같이 다수의 HTTP 프록시 디버거를 사용할 수 있습니다.

- Charles - Windows 및 OSX용 웹 디버깅 프록시
- Fiddler - Windows용 웹 디버깅 프록시

Charles와 Fiddler 모두 SSL 암호화된 트래픽을 볼 수 있도록 약간의 구성이 필요합니다. 자세한 내용은 해당 도구의 설명서를 참조하세요. 암호화된 트래픽을 표시하도록 구성할 수 없는 웹 디버깅 프록시를 사용하는 경우 aws_endpoints_json 파일을 열고 디버깅해야 하는 AWS 서비스의 HTTP 태그를 true로 설정합니다.

문서 기록

다음 표에서는 .NET 및 Xamarin용 AWS Mobile SDK의 최신 릴리스가 발표된 이후 이 설명서에서 변경된 중요 사항에 대해 설명합니다.

- API 버전:: 2015년 8월 27일
- 마지막 설명서 업데이트: 2021년 2월 23일

변경 사항	API 버전	설명	릴리스 날짜
보관됨	2015-08-27	Xamarin용 AWS Mobile SDK는 이제 AWS SDK for .NET에 포함됩니다. 이 안내서에서는 Xamarin용 모바일 SDK의 아카이브된 버전을 참조합니다.	2021-02-23
GA 릴리스	2015-08-27	GA 릴리스	2015-08-27
베타 릴리스	2015-07-28	베타 릴리스	rn2015-07-28