Xamarin 開発者ガイド

AWS Mobile SDK

AWS Mobile SDK: Xamarin 開発者ガイド

Table of Contents

AWSの Mobile SDK for Xamarin が、AWS SDK for .NETに含まれるようになりました。このガイドでは、Mobile SDK for Xamarin のアーカイブバージョンについて説明します。

の AWS Mobile SDK for .NET and Xamarin とは

AWSの Mobile SDK for Xamarin がAWS SDK for .NETに含まれるようになりました。詳細については、AWS SDK for .NET デベロッパーガイドを参照してください。

このガイドは更新されなくなりました。このガイドは、MMobile SDK for Xamarin のアーカイブバージョンについて説明しています。

関連ガイドとトピック

- フロントエンドおよびモバイルアプリケーションの開発には、AWS Amplifyの使用をお勧めします。
- AWS SDK for .NETの使用に関する特別な考慮事項については、AWS SDK for .NETデベロッパーガイドの Xamarin の Support に関する特別な考慮事項を参照してください。
- 参考のために、GitHub で、アーカイブバージョンのAWSの Mobile SDK for Xamarin を参照できます。

アーカイブされたリファレンスコンテンツ

アーカイブされた AWS Mobile SDK for .NET and Xamarin は、.NET ライブラリ、コードサンプル、ドキュメントのセットを提供し、以下に対応した接続型モバイルアプリケーションを構築する開発者をサポートします。

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

AWS Mobile SDK for .NET and Xamarin を使用して書き込まれたモバイルアプリケーションでネイティブプラットフォーム API が呼び出されるため、ルックアンドフィールはネイティブアプリケーションそのものです。SDK の .NET ライブラリには、AWS REST API の C# ラッパーが含まれています。

AWS Mobile SDK for .NET and Xamarin に含まれるもの

現在サポートされている AWS サービスには以下のものが含まれますが、これらに限定されません。

- Amazon Cognito
- Amazon S3
- Amazon DynamoDB
- Amazon Mobile Analytics
- Amazon Simple Notification Service

これらのサービスでは、ユーザーの認証、プレーヤーデータおよびゲームデータの保存、クラウド内へのオブジェクトの保存、プッシュ通知の受信、使用状況データの収集および分析を行うことができます。

AWS Mobile SDK for .NET and Xamarin では、AWS SDK for .NET でサポートされるほとんどの AWS サービスを使用することができます。モバイル開発固有の AWS サービスは、この開発者ガイドで説明しています。AWS SDK for .NET の詳細については、以下を参照してください。

- AWS SDK for .NET 入門ガイド
- .NET 用 AWS SDK 開発者ガイド
- AWS SDK for .NET API リファレンス

互換性

AWS Mobile SDK for .NET and Xamarin は、ポータブルクラスライブラリ (PCL) として出荷されています。PCL の Support が、Xamarin.Android 4.10.1 および Xamarin.iOS 7.0.4 に追加されました。ポータブルライブラリプロジェクトは、Visual Studio に組み込まれています。

IDE

Xamarin SDK のアーカイブバージョンでの IDE の使用の詳細については、で AWS Mobile SDK for .NET and Xamarin を設定するを参照してください。

AWS Mobile SDK for .NET and Xamarin を入手する方法

AWS Mobile SDK for .NET and Xamarin の入手方法については、AWS Mobile SDK for .NET and Xamarin を設定するを参照してください。AWS Mobile SDK for .NET and Xamarin は、NuGet パッ

ケージとして配布されています。AWS のサービスパッケージの完全なリストは、NuGet の AWS SDK パッケージまたは AWS SDK for .NET の Github リポジトリで確認できます。

AWS モバイルサービスについて

Amazon Cognito ID

AWS を呼び出すには、必ず AWS 認証情報が必要です。アプリに AWS 認証情報を提供するには、アプリケーションへの認証情報をハードコードするのではなく、Amazon Cognito ID を使用することをお勧めします。「AWS Mobile SDK for .NET and Xamarin をセットアップする」の手順に従って、Amazon Cognito 経由で AWS 認証情報を取得します。

Cognito では、パブリックログインプロバイダー (Amazon、Facebook、Twitter、Google など) や、OpenID Connect をサポートするプロバイダーを使用して、ユーザーを認証することができます。また、Cognito では、未認証ユーザーのアクセスを設定することもできます。Cognito では、Identity and Access Management (IAM) ロールを使用して指定した制限付きアクセス権に一時的な認証情報を提供します。また、Cognito の設定では、IAM ロールに関連付けられた ID プールを作成します。IAM ロールは、アプリケーションでアクセスできるリソースとサービスを指定します。

Cognito ID の開始方法については、「AWS Mobile SDK for .NET and Xamarin をセットアップする」を参照してください。

Cognito ID の詳細については、「Amazon Cognito ID」を参照してください。

Amazon Cognito Sync

Cognito Sync は、アプリケーション関連のユーザーデータのデバイス間の同期を有効にする、AWS サービスとクライアントライブラリです。Cognito Sync API を使用すると、デバイス間やログインプロバイダー (Amazon、Facebook、Google、独自のカスタム ID プロバイダー) 間でユーザーのプロファイルデータを同期することができます。

Cognito Sync の利用を開始するには、「ユーザーデータと Cognito Sync を同期する」を参照してください。

Cognito Sync の詳細については、「Amazon Cognito Sync」を参照してください。

Mobile Analytics

Amazon Mobile Analytics では、モバイルアプリの使用状況を収集および可視化することで、把握することができます。レポートは、アクティブユーザー、セッション、保持期間、アプリ内収益、カス

タムイベントのメトリクスで使用することができるだけでなく、プラットフォームや日付範囲でフィルタリングすることができます。Amazon Mobile Analytics は、ビジネスに合わせてスケーリングされるように設計されており、数百万のエンドポイントから数十億のイベントを収集して処理できます。

Mobile Analytics の使用を開始するには、「Amazon Mobile Analytics を使用してアプリの使用データを追跡する」を参照します。

Mobile Analytics の詳細については、「Amazon Mobile Analytics」を参照してください。

DynamoDB

Amazon DynamoDB は、拡張性と可用性に優れた、費用効果の高い、高速な非リレーショナルデータベースサービスです。DynamoDB により、データストレージに対して低いレイテンシーと予測可能なパフォーマンスを維持しながら、従来の拡張性の限界を排除できます。

DynamoDB の使用を開始するには、「DynamoDB を使用したデータの保存および取得」を参照してください。

DynamoDB に関する詳細については、「Amazon DynamoDB」を参照してください。

Amazon Simple Notification Service

Amazon Simple Notification Service (SNS) は、高速かつ柔軟な完全マネージド型のプッシュ通知サービスです。このサービスを使用すると、個々のメッセージを送信したり、多数の受信者にメッセージをファンアウトしたりできます。Amazon Simple Notification Service により、簡単かつコスト効率の高い方法で、モバイルデバイスユーザーおよびメール受信者にプッシュ通知を送信したり、他の分散サービスにメッセージを送信したりできます。

SNS for Xamarin iOS の使用を開始するには、「SNS を使用してプッシュ通知を受信する (Xamarin iOS)」を参照してください。

Xamarin Android の使用を開始するには、「SNS を使用してプッシュ通知を受信する (Xamarin Android)」を参照してください。

SNS の詳細については、「Amazon Simple Notification Service (SNS)」を参照してください。

で AWS Mobile SDK for .NET and Xamarin を設定する

AWS Mobile SDK for .NET and Xamarin を設定して新しいプロジェクトの構築を開始するか、SDK を既存のプロジェクトに統合することができます。また、SDK の仕組みを確認するために、クローンを作成して、サンプルを実行することもできます。AWS Mobile SDK for .NET and Xamarin を設定して使用を開始するには、次のステップに従います。

前提条件

AWS Mobile SDK for .NET and Xamarin を使用するには、以下を行います。

- AWS アカウントを作成します。
- Xamarin をインストールします。

前提条件を満たしたら、次の手順を行います。

1. Amazon Cognito を使用して AWS 認証情報を取得します。
2. アプリケーションで使用する各 AWS サービスに対して、必要なアクセス許可を設定します。
3. IDE で新しいプロジェクトを作成します。
4. AWS Mobile SDK for .NET and Xamarin をインストールします。
5. AWS Mobile SDK for .NET and Xamarin を設定します。

ステップ 1: AWS 認証情報を取得する

アプリケーションで AWS を呼び出すには、まず AWS 認証情報を取得する必要があります。このためには、アプリケーションにプライベートの AWS 認証情報を埋め込まずにアプリケーションから SDK のサービスにアクセスできる、Amazon Cognito、AWS のサービスを使用します。

Amazon Cognito の使用を開始するには、アイデンティティプールを作成する必要があります。ID プールは、アカウント固有の情報のストアであり、次のような一意のプール ID で識別されます。

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

1. Amazon Cognito コンソールにサインインし、[フェデレーテッドアイデンティティの管理] を選択し、続いて [新しい ID プールの作成] を選択します。

2. ID プールの名前を入力し、認証されていない ID へのアクセスを有効にするチェックボックスをオンにします。[プールの作成] を選択して、ID プールを作成します。
3. [許可] を選択して、ID プールに関連付けられた 2 つのデフォルトロール (未認証ユーザー用と認証済みユーザー用) を作成します。これらのデフォルトのロールは、Amazon Cognito Sync および Amazon Mobile Analytics へのアイデンティティプールアクセスを提供します。

通常、アプリケーションごとに使用する ID プールは 1 つのみです。

ID プールを作成したら、AWS 認証情報を取得するために、次の手順で `CognitoAWSCredentials` オブジェクト (ID プール ID に渡す) を作成し、AWS クライアントのコンストラクタに渡します。

```
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
);
```

ステップ 2: アクセス許可の設定

アプリケーションで使用する AWS サービスごとにアクセス許可を設定する必要があります。まず、アプリケーションのユーザーに対して、AWS がどのように表示されるかを理解する必要があります。

ユーザーがアプリケーションを使用して AWS を呼び出すと、AWS より ユーザーに ID が割り当てられます。ステップ 1 で作成した ID プールは、AWS によって割り当てられた ID が保存される場所です。ID には、認証済みと未認証の 2 種類あります。認証済み ID は、パブリックログインプロバイダー (例: Facebook、Amazon、Google) によって認証されているユーザーに属します。未認証 ID はゲストユーザーに属します。

各 ID は、AWS Identity and Access Management ロールに関連付けられています。ステップ 1 では、2 種類の IAM ロール (認証済みユーザーと未認証ユーザー) を作成しました。IAM ロールには、1 つ以上のポリシーがアタッチされており、そのロールに割り当てられたアイデンティティでアクセス

できる AWS のサービスを指定します。たとえば、次のサンプルポリシーは、Amazon S3 バケットへのアクセスを許可します。

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

アプリケーションで使用する AWS のサービスのアクセス許可を設定するには、ロールにアタッチされているポリシーを変更します。

1. IAM コンソールに移動し、[ロール] を選択します。検索ボックスに ID プールの名前を入力します。設定する IAM ロールを選択します。認証済みユーザーと未認証ユーザーがいずれもアプリケーションにアクセスできるようにするには、両方のロールのアクセス許可を付与する必要があります。
2. [Attach Policy (ポリシーのアタッチ)] をクリック後、アタッチするポリシーを選択し、[Attach Policy (ポリシーのアタッチ)] をクリックします。作成した IAM ロールのデフォルトポリシーで、Amazon Cognito Sync および Mobile Analytics にアクセスできます。

ポリシーの作成方法や既存ポリシーのリストから選択する方法については、「IAM ポリシー」を参照してください。

ステップ 3: 新しい プロジェクトを作成する

Windows

Visual Studio を使用してアプリケーションを開発できます。

OS X

Visual Studio を使用して、アプリケーションを開発できます。Xamarin を使用して iOS 開発を行う場合は、アプリケーションを実行するために Mac にアクセスする必要があります。詳細については、「Windows への Xamarin.iOS のインストール」を参照してください。

> **Note**
>
> JetBrains の クロスプラットフォーム商用 IDE Rider には、Windows および Mac プラットフォームの両方で、Xamarin Support が含まれています。

ステップ 4。 AWS Mobile SDK for .NET and Xamarin をインストールする

Windows

オプション 1: Package Manager コンソールを使用してインストールする

The AWS Mobile SDK for .NET and Xamarin は、.NET アセンブリのセットで構成されています。AWS Mobile SDK for .NET and Xamarin をインストールするには、Package Manager コンソールで、install-package コマンドを各パッケージに対して実行します。たとえば、Cognito Identity をインストールするには、次のコマンドを実行します。

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

AWS Core Runtime および Amazon Cognito アイデンティティパッケージは、すべてのプロジェクトにおいて必要です。各サービスのパッケージ名の完全なリストは、次のとおりです。

サービス	パッケージ名
AWS Core Runtime	AWSSDK.Core
Amazon Cognito Sync	AWSSDK.CognitoSync
Amazon Cognito ID	AWSSDK.CognitoIdentity

サービス	パッケージ名
Amazon DynamoDB	AWSSDK.DynamoDBv2
Amazon Mobile Analytics	AWSSDK.MobileAnalytics
Amazon S3	AWSSDK.S3
Amazon SNS	AWSSDK.SimpleNotificationService

未公開パッケージを含めるには、次のように、パッケージのインストール中に `-Pre` コマンドライン引数を含めます。

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

AWS のサービスパッケージの完全なリストは、NuGet の AWS SDK パッケージまたは AWS SDK for .NET Github リポジトリで確認できます。

オプション 2: IDE を使用してインストールする

Visual Studio で、

1. プロジェクトを右クリックし、[Manage NuGet Packages] をクリックします。
2. プロジェクトに追加するパッケージ名を検索します。未公開の NuGet パッケージを含めるには、[Include Prelease] を選択します。AWS サービスパッケージの完全なリストは、「NuGet の AWS SDK パッケージ」から確認できます。
3. パッケージを選択し、[Install] を選択します。

Mac (OS X)

Visual Studio で、

1. パッケージフォルダを右クリックし、[Add Packages] を選択します。
2. プロジェクトに追加するパッケージ名を検索します。未公開の NuGet パッケージを含めるには、[Show pre-release packages] を選択します。AWS サービスパッケージの完全なリストは、「NuGet の AWS SDK パッケージ」から確認できます。
3. 必要なパッケージの横にあるチェックボックスを選択し、[Add Package] を選択します。

> **Important**
>
> ポータブルクラスライブラリを使用して開発している場合は、ポータブルクラスライブラリから継承するすべてのプロジェクトにも AWSSDK.Core NuGet パッケージを追加する必要があります。

ステップ 5。 AWS Mobile SDK for .NET and Xamarin を設定する

ログ記録の設定

ログ記録を設定するには、`Amazon.AWSConfigs` クラスと `Amazon.Util.LoggingConfig` クラスを使用します。これらのクラスは、`AWSSdk.Core` アセンブリで確認できます。Visual Studio の Nuget Package Manager から利用できます。`OnCreate` メソッドのログ設定コードは、`MainActivity.cs` ファイル (Android アプリ) または `AppDelegate.cs` ファイル (iOS アプリ) に配置できます。また、ステートメント (`using Amazon` および `using Amazon.Util`) を .cs ファイルに追加します。

ログ記録設定は、次のように行います。

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

SystemDiagnostics にログを記録すると、フレームワーク内部で System.Console に出力されます。HTTP レスポンスのログを記録するには、`LogResponses` フラグを設定します。値は、Always、Never、OnError のいずれかです。

また、`LogMetrics` プロパティを使用して、HTTP リクエストのパフォーマンスメトリクスを記録します。ログ形式は、`LogMetricsFormat` プロパティを使用して指定できます。有効な値は、JSON またはスタンダードです。

リージョンのエンドポイントを設定する

次のように、すべてのサービスクライアントのデフォルトのリージョンを設定します。

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

これにより、SDK のすべてのサービスクライアントのデフォルトのリージョンが設定されます。この設定は、次の通り、サービスクライアントのインスタンス作成時に、リージョンを明示的に指定して上書きできます。

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

HTTP プロキシ設定を設定する

ネットワークがプロキシの背後にある場合は、次のように HTTP リクエストのプロキシ設定を構成できます。

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

クロックスキューの修正

このプロパティでは、適切なサーバー時間を特定して、正しい時間でリクエストを再び発行することで、SDK でクライアントクロックを修正する必要があるかどうかを判断します。

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

このフィールドは、サービスの呼び出しが例外になり、ローカル時間とサーバー時間の間にずれがあると SDK で判断された場合に設定されます。

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

クロックスキューの詳細については、AWS ブログの「クロックスキューの修正」を参照してください。

次のステップ

これで、AWS Mobile SDK for .NET and Xamarin が設定され、以下を行えるようになります。

- 使用を開始します。AWS Mobile SDK for .NET and Xamarin のサービスの使用方法および設定方法のクイックスタート手順については、AWS Mobile SDK for .NET and Xamarin の開始方法をお読み下さい。

- サービストピックを表示します。AWS Mobile SDK for .NET and Xamarin の各サービスの詳細と機能について説明します。
- デモを実行します。サンプルの Xamarin アプリケーションを使用して、一般的ユースケースについて説明します。このサンプルアプリケーションを実行して、前述の AWS Mobile SDK for .NET and Xamarin を設定するには、各サンプルの README ファイルに記載された手順に従います。
- API について説明します。|sdk-xamarin-ref|_ を表示します。
- 質問する: AWS Mobile SDK フォーラムに質問を投稿するか、GitHub で問題を提起してください。

で AWS Mobile SDK for .NET and Xamarin を開始する方法

AWS Mobile SDK for .NET and Xamarin には、Xamarin アプリケーションから AWS サービスを呼び出すために必要なライブラリ、サンプル、ドキュメントが含まれています。

以下のサービスの使用を開始する前に、「AWS Mobile SDK for .NET and Xamarin をセットアップする」ページのすべての手順を完了する必要があります。

これらの「開始方法」トピックでは、以下について説明します。

トピック

- Amazon S3 でファイルの保存と取り出し
- ユーザーデータと Cognito Sync を同期する
- DynamoDB でデータの保存と取り出し
- Amazon Mobile Analytics を使用してアプリの使用データを追跡する
- SNS を使用してプッシュ通知を受信する (Xamarin iOS)
- SNS を使用してプッシュ通知を受信する (Xamarin Android)

その他の AWS Mobile SDK の詳細については、「AWS Mobile SDK」を参照してください。

Amazon S3 でファイルの保存と取り出し

「Amazon Simple Storage Service (Amazon S3)」を使用すると、モバイル開発者は、耐久性および耐障害性が高く、セキュアなオブジェクトストレージを使用できるようになります。Amazon S3 の使用方法は簡単です。シンプルなウェブサービスインターフェイスを使用して、ウェブ上のどこからでも容量に関係なくデータを保存、取得できます。

以下のチュートリアルでは、S3 をお客様のアプリで使用するための高度なユーティリティである S3 TransferUtility を統合する方法について説明します。Xamarin アプリケーションから S3 を使用する方法の詳細については、「Amazon Simple Storage Service (S3)」を参照してください。

プロジェクトのセットアップ

前提条件

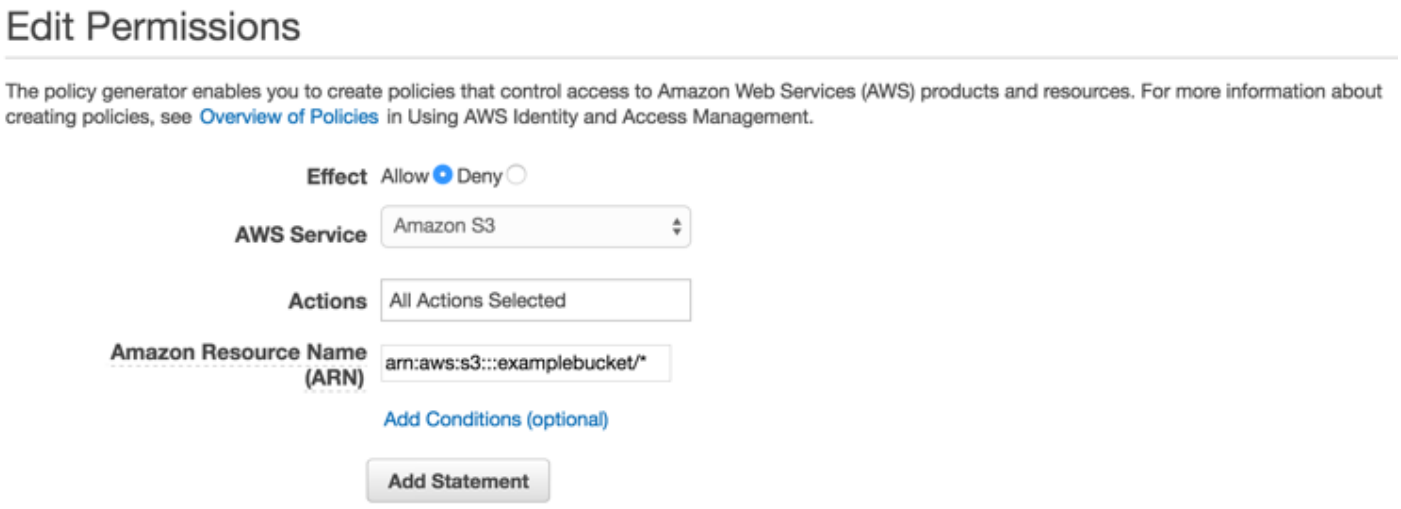
このチュートリアルを開始する前に、必ず「AWS Mobile SDK for .NET and Xamarin をセットアップする」の手順をすべて完了する必要があります。

このチュートリアルでは、S3 バケットをすでに作成していることを前提としています。S3 バケットを作成するには、S3 AWS コンソールに移動しします。

S3 のアクセス許可を設定

デフォルトの IAM ロールのポリシーでは、Amazon Mobile Analytics および Amazon Cognito Sync へのアクセス権がアプリケーションに付与されています。Cognito ID プールから Amazon S3 にアクセスできるようにするには、ID プールのロールを変更する必要があります。

1. Identity and Access Management コンソールに移動し、左側のペインの [ロール] をクリックします。
2. 検索ボックスに ID プールの名前を入力します。2 つのロール (未認証ユーザーと認証済みユーザー) が表示されます。
3. 未認証ユーザーのロールをクリックします (unauth を ID プール名に追加)。
4. [ロールポリシーの作成] をクリックして [Policy Generator] を選択し、[選択] をクリックします。
5. [アクセス許可の編集] ページで、以下のイメージに示す設定を入力し、Amazon リソースネーム (ARN) を実際の名前に置き換えます。S3 バケットの ARN は、`arn:aws:s3:::examplebucket/*` のように表示され、バケットを配置するリージョンとバケットの名前で構成されます。以下に示す設定では、ID プールから指定したバケットのすべてのアクションにフルアクセスを付与しています。



1. [ステートメントを追加] ボタンをクリックし、[次のステップ] をクリックします。
2. ウィザードに、生成した設定が表示されます。[ポリシーの適用] をクリックします。

S3 に対するアクセス権の付与の詳細については、「Amazon S3 バケットへアクセス権を付与する」を参照してください。

S3 の NuGet パッケージをプロジェクトに追加する

「AWS Mobile SDK for .NET and Xamarin のセットアップ」のステップ 4 に従って、S3 NuGet パッケージをプロジェクトに追加します。

(オプション) S3 リクエストの署名バージョンを設定する

Amazon S3 とのすべてのやり取りは認証されるか匿名で行われます。AWS では、署名バージョン 4 または署名バージョン 2 のアルゴリズムを使用して、サービスへの呼び出しを認証します。

2014 年 1 月以降に作成されたすべての新しい AWS リージョンでは、署名バージョン 4 のみをサポートしています。ただし、以前のリージョンの多くは、現在も署名バージョン 4 および署名バージョン 2 のリクエストに対応しています。

バケットが、こちらのページの署名バージョン 2 のリクエストをサポートしていないリージョンのいずれかにある場合は、次のように、AWSConfigsS3.UseSignatureVersion4 プロパティを「true」に設定する必要があります。

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

AWS 署名バージョンの詳細については、「リクエストの認証 (AWS 署名バージョン 4)」を参照してください。

S3 TransferUtility クライアントを初期化します。

S3 クライアントを作成して、次のように AWS 認証情報を渡し、続いて S3 クライアントを転送ユーティリティに渡します。

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

Amazon S3 にファイルをアップロード

S3 にファイルをアップロードするには、Transfer Utility オブジェクトの `Upload` を呼び出し、次のパラメータを渡します。

- `file` - アップロードするファイルの文字列名。

- bucketName - ファイルを保存する S3 バケットの文字列名。

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

上記のコードは、ファイルがディレクトリ (Environment.SpecialFolder.ApplicationData) にあることを前提としています。大容量ファイルをアップロードする場合は、自動的に S3 のマルチパートアップロード機能が使用されるため、スループットが向上します。

Amazon S3 からのファイルのダウンロード

S3 からファイルをダウンロードするには、Transfer Utility オブジェクトの Download を呼び出し、次のパラメータを渡します。

- file - ダウンロードするファイルの文字列名。
- bucketName - ファイルのダウンロード元の S3 バケットの文字列名。
- key - ダウンロードする S3 オブジェクト (この場合はファイル) の名前を表す文字列。

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

Xamarin アプリケーションから Amazon S3 へのアクセスの詳細については、「Amazon Simple Storage Service (S3)」を参照してください。

ユーザーデータと Cognito Sync を同期する

Amazon Cognito Sync を使用すると、アプリケーションの設定やゲームの状態などのモバイルユーザーデータを AWS クラウドに簡単に保存できます。バックエンドコードの記述やインフラストラクチャの管理は必要ありません。ユーザーのデバイスにローカルでデータを保存すれば、デバイスがオフラインの場合でもアプリケーションを使用することができます。また、ユーザーのデバイス間でデータを同期して、使用するデバイスを問わずアプリのエクスペリエンスに整合性を持たせることもできます。

以下のチュートリアルでは、Sync をアプリに統合する方法について説明します。

プロジェクトのセットアップ

前提条件

このチュートリアルを開始する前に、必ず「AWS Mobile SDK for .NET and Xamarin をセットアップする」の手順をすべて完了する必要があります。

Cognito Sync リソースにアクセス権限を付与する

セットアップ時に作成した未認証および承認済みのロールに関連付けられているデフォルトポリシーを使用して、アプリケーションから Cognito Sync にアクセスできるようにします。必要な設定はこれだけです。

Cognito Sync の NuGet パッケージをプロジェクトに追加する

「AWS Mobile SDK for .NET and Xamarin をセットアップする」のステップ 4 に従って、Cognito SyncManager の NuGet パッケージをプロジェクトに追加します。

CognitoSyncManager を初期化する

初期化した Amazon Cognito 認証情報プロバイダーを `CognitoSyncManager` コンストラクタに渡します。

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

ユーザーデータを同期する

未認証のユーザーデータを同期するには、以下のように行います。

1. データセットを作成します。
2. ユーザーデータをデータセットに追加します。
3. データセットをクラウドと同期します。

データセットを作成する

Dataset のインスタンスを作成します。openOrCreateDataset メソッドは、新しいデータセットを作成するか、デバイスにローカルで保存されている既存のデータセットのインスタンスを開くために使用されます。

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

ユーザーデータをデータセットに追加する

ユーザーデータは、キーと値のペアの形式で追加されます。

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

Cognito データセット関数は、キーによってアクセス可能な値を持つディクショナリとして機能します。

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

データセットを同期する

データセットを同期するには、その同期メソッドを呼び出します。

```
dataset.SynchronizeAsync();

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

データセットに書き込まれたデータはすべて、データセットが同期されるまでローカルに保存されます。このセクションのコードでは、未認証の Cognito ID を使用していることを前提としているため、ユーザーデータをクラウドと同期する際はデバイスごとに保存されます。デバイスには、デバイス ID が関連付けられています。クラウドと同期されると、ユーザーデータは、そのデバイス ID に関連付けられます。

Cognito Sync の詳細については、「Amazon Cognito Sync」を参照してください。

DynamoDB でデータの保存と取り出し

Amazon DynamoDB は、拡張性と可用性に優れた、費用効果の高い、高速な非リレーショナルデータベースサービスです。DynamoDB により、データストレージに対して低いレイテンシーと予測可能なパフォーマンスを維持しながら、従来の拡張性の限界を排除できます。

以下のチュートリアルでは、DynamoDB オブジェクト永続モデルを、DynamoDB にオブジェクトを格納するアプリと統合する方法について説明します。

プロジェクトのセットアップ

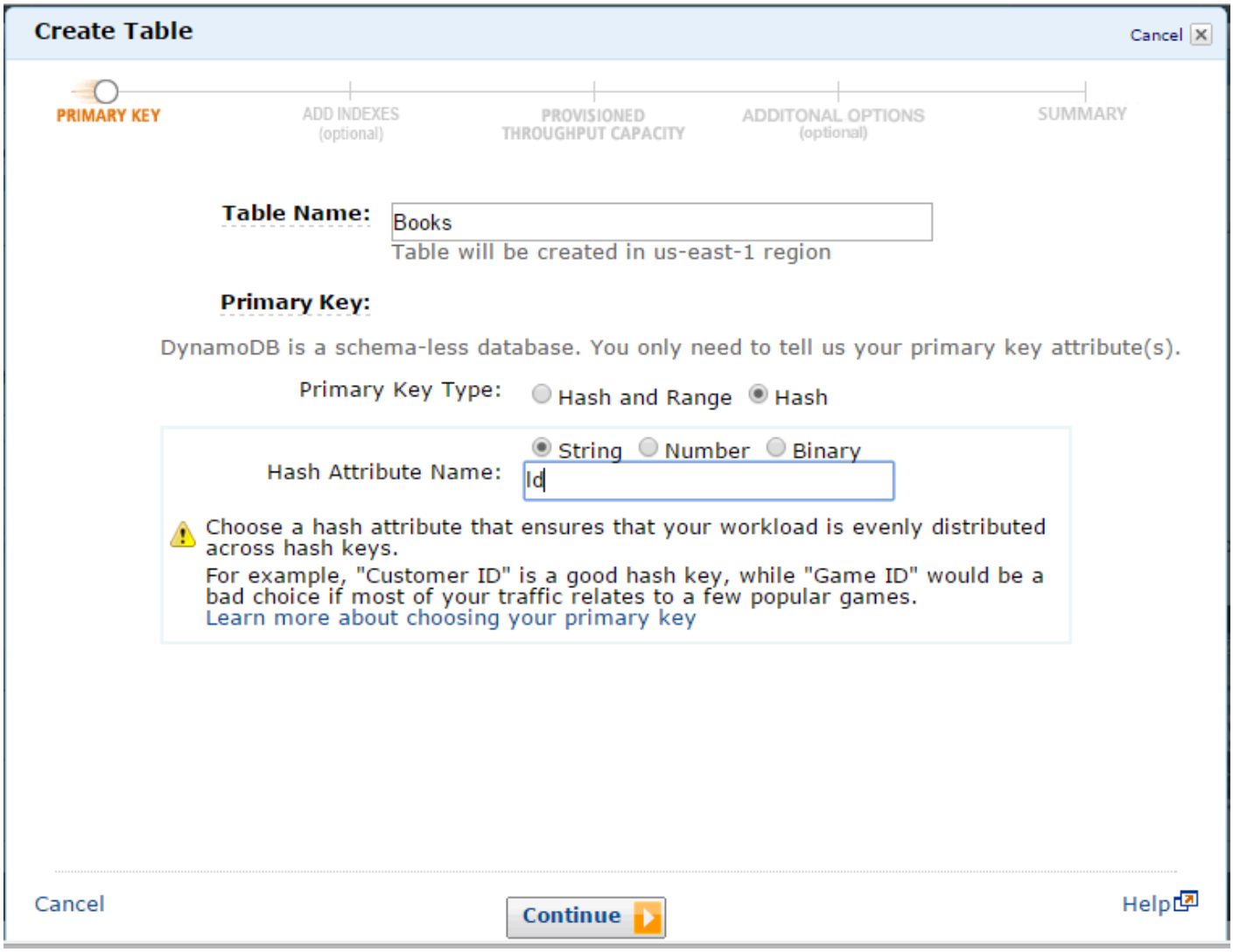
前提条件

このチュートリアルを開始する前に、必ず「AWS Mobile SDK for .NET and Xamarin をセットアップする」の手順をすべて完了する必要があります。

DynamoDB テーブルの作成

DynamoDB データベースにデータを読み書きするには、テーブルを作成する必要があります。テーブルを作成するには、プライマリキーを指定する必要があります。プライマリキーは、ハッシュ属性とオプションの範囲属性で構成されます。プライマリおよび範囲属性の使用方法の詳細については、「テーブルの操作」を参照してください。

1. DynamoDB コンソールに移動し、[テーブルの作成] をクリックします。[テーブルの作成] ウィザードが表示されます。
2. 下に示すように、テーブル名、プライマリキーのタイプ (ハッシュ)、およびハッシュ属性名 (「Id」) を指定して、[Continue (続行)] をクリックします。

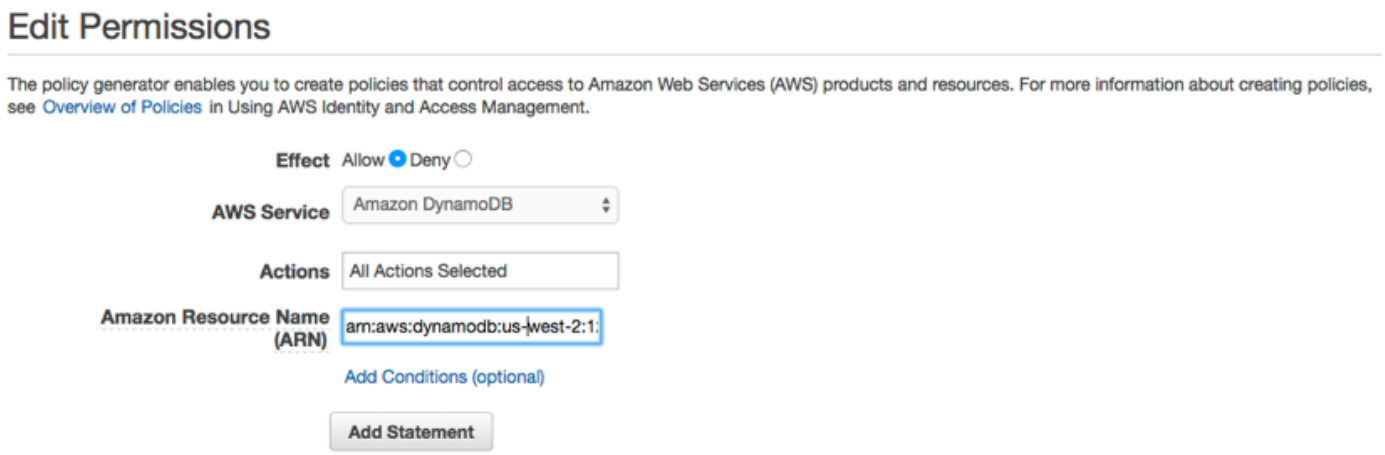


3. 次の画面の編集フィールドを空白のままにして、[続行] をクリックします。
4. [読み込みキャパシティーユニット] と [書き込みキャパシティーユニット] のデフォルト値をそのまま使用し、[続行] をクリックします。
5. 次の画面で、[通知の送信先:] テキストボックスに E メールアドレスを入力し、[続行] をクリックします。レビュー画面が表示されます。
6. [Create] (作成) をクリックします。テーブルが作成されるまでには数分かかる場合があります。

DynamoDB のアクセス許可を設定する

ID プールが Amazon DynamoDB にアクセスを可能にするには、ID プールのロールを変更する必要があります。

1. Identity and Access Management コンソールに移動し、左のペインの [ロール] をクリックします。ID プール名を検索します。未認証ユーザーと認証されたユーザーの 2 つのロールが表示されます。
2. 未認証ユーザーのロール (unauth を ID プール名に追加) をクリックし、[Create Role Policy (ロールポリシーの作成)] をクリックします。
3. [Policy Generator] を選択し、[選択] をクリックします。
4. [アクセス許可の編集] ページで、次の画像に示されている設定を入力します。DynamoDB テーブルの Amazon Resource Name (ARN) は `arn:aws:dynamodb:us-west-2:123456789012:table/Books` のようになり、テーブルが存在するリージョン、所有者の AWS アカウント番号、および `table/Books` の形式のテーブル名で構成されています。ARN 形式の詳細については、「DynamoDB の Amazon リソースネーム」を参照してください。



5. [ステートメントを追加] をクリックし、[次のステップ] をクリックします。ウィザードに、生成した設定が表示されます。
6. [ポリシーの適用] をクリックします。

DynamoDB の NuGet パッケージをプロジェクトに追加する

「AWS Mobile SDK for .NET and Xamarin のセットアップ」のステップ 4 に従って、DynamoDB NuGet パッケージをプロジェクトに追加します。

Initialize AmazonDynamoDBClient

初期化された Amazon Cognito の認証情報プロバイダーとリージョンを `AmazonDynamoDB` コンストラクタに渡し、クライアントを DynamoDBContext に渡します。

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

クラスの作成

行をテーブルに書き込むには、行データを保持するクラスを定義します。クラスは、属性を保持するプロパティも含まれていますが、コンソールで作成された DynamoDB テーブルにマップされます。次のクラス宣言はそのようなクラスを示しています。

```
[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; }
}
```

項目の保存

項目を保存するには、最初にオブジェクトを作成します。

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

次に、保存します。

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

行を更新するには、DDTableRow クラスのインスタンスを変更し、上記のように AWSDynamoObjectMapper.save() を呼び出します。

項目の取得

プライマリキーを使用して項目を取得

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

項目の更新

項目を更新するには:

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

項目の削除

項目を削除するには:

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

Xamarin アプリケーションから DynamoDB へのアクセスの詳細については、「Amazon DynamoDB」を参照してください。

Amazon Mobile Analytics を使用してアプリの使用データを追跡する

Amazon Mobile Analytics を使用すると、アプリ使用量やアプリ収益を計測できます。新規ユーザーかリピートユーザーか、アプリの収益、ユーザーの維持、そしてアプリケーション内でのカスタム動作イベントなどのキートレンドを追跡することで、データを基にした決定を下し、アプリのエンゲージメントや収益を増やすことができます。

以下のチュートリアルでは、Mobile Analytics をアプリに統合する方法について説明します。

プロジェクトのセットアップ

前提条件

このチュートリアルを開始する前に、必ず「AWS Mobile SDK for .NET and Xamarin をセットアップする」の手順をすべて完了する必要があります。

Mobile Analytics コンソールでアプリを作成する

Amazon Mobile Analytics コンソールに移動して、アプリを作成します。appId 値は、後で必要になるため記録しておきます。Mobile Analytics コンソールでアプリを作成する場合は、ID プール ID を指定する必要があります。ID プールの作成手順については、「AWS Mobile SDK for .NET and Xamarin をセットアップする」を参照してください。

コンソールでの操作方法については、Amazon Mobile Analytics ユーザーガイドを参照してください。

Mobile Analytics のアクセス許可を設定する

セットアップ時に作成したロールに関連付けられているデフォルトポリシーを使用して、アプリケーションから Mobile Analytics にアクセスできるようにします。必要な設定はこれだけです。

Mobile Analytics の NuGet パッケージをプロジェクトに追加する

「AWS Mobile SDK for .NET and Xamarin をセットアップする」のステップ 4 に従って、Mobile Analytics の NuGet パッケージをプロジェクトに追加します。

Mobile Analytics の設定

Mobile Analytics では、awsconfig.xml で設定可能な設定の一部を定義します。

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

- AllowUseDataNetwork - データネットワーク上でセッションイベントを送信するかどうかを指定するブール値。

- DBWarningThreshold - データベースのサイズ制限。この値に達すると、警告ログが生成されます。
- MaxDBSize - SQLIte データベースのサイズ。データベースの最大サイズに達すると、それ以降のイベントは削除されます。
- MaxRequestSize - HTTP リクエストで Mobile Analytics サービスに送信する必要がある、バイト単位のリクエストの最大サイズ。
- SessionTimeout - アプリケーションがバックグラウンドに移行してからセッションが終了するまでの時間間隔。

上記の設定は、各設定項目のデフォルト値です。

MobileAnalyticsManager の初期化

MobileAnalyticsManager を初期化し、`MobileAnalyticsManager` で GetOrCreateInstance を呼び出して、AWS 認証情報、リージョン、Mobile Analytics アプリケーション ID、オプションの構成オブジェクトに渡すには、次のようにします。

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

トラックセッションイベント

Xamarin Android

アクティビティの `OnPause()` および `OnResume()` メソッドを上書きし、セッションイベントを記録します。

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

protected override void OnPause()
{
```

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

これは、アプリケーションのアクティビティごとに実施する必要があります。

Xamarin iOS

AppDelegate.cs で、次のように行います。

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

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

Mobile Analytics の詳細については、「Amazon Mobile Analytics」を参照してください。

SNS を使用してプッシュ通知を受信する (Xamarin iOS)

このドキュメントでは、Amazon Simple Notification Service (SNS) および AWS Mobile SDK for .NET and Xamarin を使用して、Xamarin iOS アプリケーションにプッシュ通知を送信する方法について説明します。

プロジェクトのセットアップ

前提条件

このチュートリアルを開始する前に、必ず「AWS Mobile SDK for .NET and Xamarin をセットアップする」の手順をすべて完了する必要があります。

SNS のアクセス許可を設定

AWS Mobile SDK for .NET and Xamarin をセットアップするのステップ 2 に従って、以下のポリシーをアプリケーションのロールにアタッチします。これにより、SNS にアクセスするための適切な権限がアプリケーションに付与されます。

1. IAM コンソールに移動し、設定する IAM ロールを選択します。
2. [ポリシーのアタッチ] をクリック後、AmazonSNSFullAccess ポリシーを選択し、[ポリシーのアタッチ] をクリックします。

> **Warning**
>
> AmazonSNSFullAccess を本番稼働用環境で使用することは推奨されていません。すばやく起動して実行できるように、こちらの環境を使用します。IAM ロールのアクセス許可を指定する方法については、「IAM ロールのアクセス許可の概要」を参照してください。

Apple iOS 開発者プログラムのメンバーシップを取得する

プッシュ通知を受け取るには、物理デバイスでアプリを実行する必要があります。デバイスでアプリを実行するには、Apple iOS 開発者プログラムのメンバーシップのメンバーシップが必要です。メンバーシップを取得したら、Xcode を使用して署名 ID を生成できます。詳細については、Apple の App ディストリビューションクイックスタートドキュメントを参照してください。

iOS 証明書を作成する

まず、iOS 証明書を作成する必要があります。次に、プッシュ通知用に設定されたプロビジョニングプロファイルを作成します。そのためには、次の操作を行います。

1. Apple 開発者メンバーセンターに移動し、[Certificates, Identifiers & Profiles] をクリックします。
2. [iOS Apps] の [Identifiers] をクリック後、その ウェブページの右上隅の + ボタンをクリックして新しい iOS App ID を追加し、App ID の説明を入力します。
3. [Add ID Suffix (ID サフィックスを追加)] セクションまで下にスクロールし、[Explicit App ID (明示的な App ID)] を選択して、バンドル ID を入力します。
4. [App Services (アプリサービス)] セクションまで下にスクロールし、[Push Notifications (プッシュ通知)] を選択します。
5. [次へ] をクリックします。
6. 送信 をクリックします。
7. [Done (完了)] をクリックします。
8. 作成した App ID を選択し、[Edit] をクリックします。
9. [Push Notifications (プッシュ通知)] セクションまで下にスクロールします。[Development SSL Certificate] の [Create Certificate] を選択します。

10.手順に従って、証明書署名リクエスト (CSR) を作成してアップロードし、Apple Notification Service (APNS) との通信に使用する SSL 証明書をダウンロードします。

11.[Certificates, Identifiers & Profiles] ページに戻ります。[Provisioning Profiles] の [All] をクリックします。

12.右上隅の + ボタンをクリックし、新しいプロビジョニングプロファイルを追加します。

13.[iOS App Development] を選択し、[Continue] をクリックします。

14.App ID を選択し、[Continue] をクリックします。

15.開発者の証明書を選択して、[Continue] をクリックします。

16.デバイスを選択し、[Continue] をクリックします。

17.プロファイル名を入力し、[Generate] をクリックします。

18.プロビジョニングプロファイルをダウンロードしてダブルクリックし、インストールします。

プッシュ通知用に設定されたプロファイルのプロビジョニングに関する詳細については、Apple の Configuring Push Notifications ドキュメントを参照してください。

SNS コンソールで証明書を使用してプラットフォーム ARN を作成する

1. KeyChain access アプリを実行して、画面左下の [My Certificates] を選択したら、APNS に接続するために生成した SSL 証明書を右クリックして、[Export] を選択します。証明書を保護するためのファイル名とパスワードを指定するよう求められます。証明書は、P12 ファイルに保存されます。
2. SNS コンソールに移動し、画面左側の [Applications] をクリックします。
3. [プラットフォームアプリケーションの作成] をクリックして、新しい SNS プラットフォームアプリケーションを作成します。
4. アプリケーション名を入力します。
5. [Push notification platform] の [Apple Development] を選択します。
6. [ファイルの選択] をクリックし、SSL 証明書をエクスポートした際に作成した P12 ファイルを選択します。
7. SSL 証明書をエクスポートした際に指定したパスワードを入力し、[認証情報をファイルから読み込み] をクリックします。
8. [プラットフォームアプリケーションの作成] をクリックします。
9. 作成したプラットフォームアプリケーションを選択して、アプリケーション ARN をコピーします。このアプリケーション ARN は、この後の手順で必要になります。

SNS の NuGet パッケージをプロジェクトに追加する

「AWS Mobile SDK for .NET and Xamarin をセットアップする」のステップ 4 に従って、Amazon Simple Notification Service の NuGet パッケージをプロジェクトに追加します。

SNS クライアントを作成する

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

アプリケーションにリモート通知を登録する

アプリケーションを登録するには、以下に示すように、UIApplication オブジェクトの RegisterForRemoteNotifications を呼び出します。AppDelegate.cs に以下のコードを配置し、以下に表示されているプラットフォームアプリケーションの ARN を挿入します。

```
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 コンソールからエンドポイントにメッセージを送信する

1. [SNS コンソール] > [アプリケーション] の順に移動します。
2. プラットフォームアプリケーションを選択し、エンドポイントを選択したら、[エンドポイントへの発行] をクリックします。
3. テキストボックスにテキストメッセージを入力し、[メッセージの発行] をクリックしてメッセージを発行します。

SNS を使用してプッシュ通知を受信する (Xamarin Android)

このチュートリアルでは、Amazon Simple Notification Service (SNS) および AWS Mobile SDK for .NET and Xamarin を使用して、Xamarin Android アプリケーションにプッシュ通知を送信する方法について説明します。

プロジェクトのセットアップ

前提条件

このチュートリアルを開始する前に、必ず「AWS Mobile SDK for .NET and Xamarin をセットアップする」の手順をすべて完了する必要があります。

SNS のアクセス許可を設定

AWS Mobile SDK for .NET and Xamarin をセットアップするのステップ 2 に従って、以下のポリシーをアプリケーションのロールにアタッチします。これにより、SNS にアクセスするための適切な権限がアプリケーションに付与されます。

1. IAM コンソールに移動し、設定する IAM ロールを選択します。
2. [ポリシーのアタッチ] をクリック後、AmazonSNSFullAccess ポリシーを選択し、[ポリシーのアタッチ] をクリックします。

> **Warning**
>
> AmazonSNSFullAccess を本番稼働用環境で使用することは推奨されていません。すばやく起動して実行できるように、こちらの環境を使用します。IAM ロールのアクセス許可を指定する方法については、「IAM ロールのアクセス許可の概要」を参照してください。

Google Cloud のプッシュ通知を有効にする

まず、新しい Google API プロジェクトを追加します。

1. Google 開発者コンソールに移動します。
2. [Create Project] をクリックします。
3. [New Project] ボックスにプロジェクト名を入力し、プロジェクト ID をメモしたら (後で使用します)、[Create] をクリックします。

次に、プロジェクトの Google クラウド メッセージング (GCM) サービスを有効にします。

1. Google 開発者コンソールで、新しいプロジェクトが既に選択されています。選択されていない場合は、ページ上部にあるドロップダウンでプロジェクトを選択します。
2. ページ左側のサイドバーから [APIs & auth (API と Auth)] を選択します。
3. 検索ボックスに「Android 用 Google クラウドメッセージング」と入力し、Google Cloud Messaging for Android リンクをクリックします。
4. [Enable API (API を有効化)] をクリックします。

最後に、API キーを取得します。

1. Google 開発者コンソールで、[APIs & auth (API と Auth)]>[Credentials (認証情報)] の順に選択します。
2. [Public API access (パブリック API アクセス)] で、[Create new key (新しいキーを作成)] をクリックします。
3. [Create a new key (新しいキーの作成)] ダイアログで、[Server key (サーバーキー)] をクリックします。
4. 表示されたダイアログボックスで [Create (作成)] を選択し、表示された API キーをコピーします。この API キーは、後で認証を実行するために使用します。

SNS コンソールでプロジェクト ID を使用してプラットフォーム ARN を作成する

1. SNS コンソールに移動します。
2. 画面の左側にある [アプリケーション] をクリックします。
3. [プラットフォームアプリケーションの作成] をクリックして、新しい SNS プラットフォームアプリケーションを作成します。
4. アプリケーション名を入力します。
5. [Push notification platform] で [Google Cloud Messaging (GCM)] を選択します。
6. [API キー] と表示されているテキストボックスに API キーを貼り付けます。
7. [プラットフォームアプリケーションの作成] をクリックします。
8. 作成したプラットフォームアプリケーションを選択して、アプリケーション ARN をコピーします。

SNS の NuGet パッケージをプロジェクトに追加する

「AWS Mobile SDK for .NET and Xamarin をセットアップする」のステップ 4 に従って、Amazon Simple Notification Service の NuGet パッケージをプロジェクトに追加します。

SNS クライアントを作成する

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

アプリケーションにリモート通知を登録する

Android にリモート通知を登録するには、BroadcastReceiver を作成する必要があります。これにより、Google クラウドメッセージを受信できるようになります。以下のパッケージ名を変更します。ここで、次のように表示されます。

```
[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);
      }
}
```

以下は、BroadcastReceiver よりプッシュ通知を受け取り、デバイスの通知バーに通知を表示するサービスです。

```
[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 コンソールからエンドポイントにメッセージを送信する

1. [SNS コンソール] > [アプリケーション] の順に移動します。
2. プラットフォームアプリケーションを選択し、エンドポイントを選択したら、[エンドポイントへの発行] をクリックします。
3. テキストボックスにテキストメッセージを入力し、[メッセージの発行] をクリックしてメッセージを発行します。

Amazon Cognito ID

Amazon Cognito ID とは

Amazon Cognito ID では、ユーザーの一意の ID を作成し、ID プロバイダーで認証することができます。ID により、権限が制限された一時的な AWS 認証情報を取得して、Amazon Cognito Sync とデータを同期するか、他の AWS サービスに直接アクセスできます。Amazon Cognito ID はパブリック ID プロバイダー (Amazon、Facebook、Google) および認証されていない ID をサポートします。また、独自のバックエンド認証プロセスを通じてユーザーを登録し、認証することができる、開発者が認証した ID をサポートします。

Cognito ID の詳細については、Amazon Cognito 開発者ガイドを参照してください。

Cognito 認証リージョンの可用性の詳細については、「AWS サービスリージョンの可用性」を参照してください。

パブリックプロバイダーを使用してユーザーの認証

Amazon Cognito ID を使用すると、ユーザーに固有の ID を作成し、Amazon S3 や Amazon DynamoDB などの AWS リソースへの安全なアクセスを認証できます。Amazon Cognito ID は、パブリック ID プロバイダー (Amazon、Facebook、Twitter/Digits、Google、OpenID Connect と互換性のあるプロバイダー) および未認証 ID をサポートします。

Amazon、Facebook、Twitter/Digits、Google などのパブリック ID プロバイダーを使用してユーザーを認証する方法については、Amazon Cognito 開発者ガイドの「外部プロバイダー」を参照してください。

開発者が認証した ID の使用

Amazon Cognito は、Facebook、Google、Amazon を通じたウェブ ID フェデレーションに加えて、開発者が認証した ID をサポートします。開発者が認証した ID では、引き続き「Amazon Cognito Sync」を使用してユーザーデータを同期し、AWS リソースにアクセスしながら、独自の既存の認証プロセスを通じてユーザー登録や認証ができます。デベロッパーが認証したアイデンティティの使用には、エンドユーザーのデバイス、認証のバックエンド、および Amazon Cognito 間の対話が関連します。

開発者が認証した ID の詳細については、Amazon Cognito 開発者ガイドの「開発者が認証した ID」を参照してください。

Amazon Cognito Sync

Amazon Cognito Sync とは

Cognito Sync は、ユーザーデータ (例: ゲームスコア、ユーザー設定、ゲームのステータス) をデバイス間で同期できるようにする、AWS サービスかつクライアントライブラリです。Cognito Sync API を使用すると、デバイス間でユーザーデータを同期することができます。Cognito Sync をアプリで使用するには、プロジェクトに「 | 」を含む必要があります。

Amazon Cognito Sync をアプリケーションに統合する方法については、Amazon Cognito Sync 開発者ガイドを参照してください。

Amazon Mobile Analytics

Amazon Mobile Analytics は、アプリの使用状況データを大規模に収集、視覚化、把握、抽出できるサービスです。Mobile Analytics は、標準のデバイスデータとカスタムイベントの両方を簡単に取得し、自動的にレポートを計算します。以下の集計レポートに加えて、Redshift と S3 にエクスポートするデータを自動的に設定して、さらに分析することもできます。

Amazon Mobile Analytics を使用すると、顧客の行動を追跡し、メトリクスを集約して、データを視覚化できるほか、有意義なパターンを特定することができます。

主なコンセプト

レポートタイプ

Mobile Analytics は、初期状態で、Mobile Analytics コンソールで次のレポートを提供します。

- Daily Active Users (DAU)、Monthly Active Users (MAU)、および新しいユーザー
- Sticky Factor (DAU を MAU で割ったもの)
- Daily Active User ごとの Session Count および Average Sessions
- Average Revenue per Daily Active User (ARPDAU) および Average Revenue per Daily Paying Active User (ARPPDAU)
- 1 日、3 日および 7 日ごとの保持期間と、1 週間、2 週間、および 3 週間ごとの保持期間
- カスタムイベント:

これらのレポートは、コンソールに 6 つのレポートタブで表示されます。

- 概要 - 事前に選択された次の 9 つのレポートを、見やすいダッシュボードで追跡し、簡単に確認することができます。MAU、DAU、New Users、Sticky Factor、Daily Sessions、1-Day Retention、ARPDAU、Daily Paying Users、ARPPDAU。
- アクティブなユーザー - アプリを毎日、毎月どのくらいのユーザーが利用しているかを追跡し、エンゲージメント、アピール、収益化を測定するためのスティッキー係数を監視します。
- セッション - 特定の日にアプリを使用する頻度と、1 日に各ユーザーがアプリを開く頻度を追跡します。
- 保持期間 - 顧客が、毎日、毎週、アプリに戻ってくる率を追跡します。
- 収益 - アプリ内の収益動向を追跡して、収益性向上のための領域を特定します。

- カスタムイベント – アプリに特化したカスタムのユーザーアクションを追跡します。

Mobile Analytics レポートと Mobile Analytics コンソールでの作業の詳細については、Mobile Analytics 開発者ガイドの「Mobile Analytics コンソールレポートの概要」を参照してください。

プロジェクトのセットアップ

前提条件

アプリケーションで Mobile Analytics を使用するには、プロジェクトに SDK を追加する必要があります。「AWS Mobile SDK for .NET and Xamarin をセットアップする」の手順に従って、追加します。

Mobile Analytics の設定

Mobile Analytics では、awsconfig.xml で設定可能な設定の一部を定義します。

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

- SessionTimeout - SessionTimeout より長い時間バックグラウンドにとどまると、Mobile Analytics クライアントは現在のセッションを終了し、アプリがフォアグラウンドに戻ったときに新しいセッションが作成されます。5 ~ 10 の範囲の値を使用することをお勧めします。デフォルト値は 5 です。
- MaxDBSize - イベントのローカルストレージに使用されるデータベースの最大サイズ (バイト単位)。データベースのサイズがこの値を超えた場合は、追加のイベントは無視されます。1 MB ~ 10 MB の範囲の値を使用することをお勧めします。デフォルト値は 5,242,880 (5 MB) です。
- DBWarningThreashold - 警告のしきい値。有効な値の範囲は 0 ~ 1 です。値がしきい値を超える場合は警告ログが生成されます。デフォルト値は 0.9 です。
- MaxRequestSize - Mobile Analytics サービスに対して行われた HTTP リクエストの最大サイズ。値はバイト単位で指定され、1 ～ 512 KB の範囲で指定できます。デフォルト値は 102,400 (100 KB) です。512 KB を超える値を使用しないでください。サービスによって HTTP リクエストが拒否される可能性があります。

- AllowUseDataNetwork - 携帯電話データネットワーク上でサービスコールが許可されるかどうかを示す値。このオプションは慎重に使用してください。お客様の使用状況データが増える可能性があります。

上記の設定は、各設定項目のデフォルト値です。

Mobile Analytics とお客様のアプリケーションの統合

以下のセクションでは、Mobile Analytics をアプリに統合する方法について説明します。

Mobile Analytics コンソールでアプリを作成する

Amazon Mobile Analytics コンソールに移動して、アプリを作成します。appId 値は、後で必要になるため記録しておきます。Mobile Analytics コンソールでアプリを作成する場合は、ID プール ID を指定する必要があります。ID プールの作成手順については、「AWS Mobile SDK for .NET and Xamarin をセットアップする」を参照してください。

Mobile Analytics コンソールでの作業の詳細については、Mobile Analytics 開発者ガイドの「Mobile Analytics コンソールレポートの概要」を参照してください。

MobileAnalyticsManager クライアントを作成

MobileAnalyticsManager を初期化し、MobileAnalyticsManager で GetOrCreateInstance を呼び出して、AWS 認証情報、リージョン、Mobile Analytics アプリケーション ID、オプションの構成オブジェクトに渡すには、次のようにします。

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

APP_ID は、アプリの作成ウィザードで生成されます。これらの値はいずれも、Mobile Analytics コンソールの値と一致する必要があります。APP_ID は、Mobile Analytics コンソールのデータをグループ化するために使用されます。Mobile Analytics コンソールでアプリ作成後にアプリ ID を検索するには、Mobile Analytics コンソールに移動して、画面右上隅の歯車アイコンをクリックします。

これにより、登録済みのアプリとアプリ ID がすべて表示されたアプリの管理ページが表示されます。

収益化イベントの記録

AWS Mobile SDK for .NET and Xamarin には、MonetizationEventクラスが用意されています。このクラスを使用して、モバイルアプリケーション内の購入を追跡する収益化イベントを生成できます。収益化イベントの作成方法を示すコードスニペットは、以下のとおりです。

```
// 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);
```

カスタムイベントの記録

Mobile Analytics では、カスタムイベントを定義できます。カスタムイベントは全体的に定義されています。そのため、アプリやゲーム固有のユーザーアクションを追跡しやすくなります。カスタムイベントの詳細については、「カスタムイベント」を参照してください。

この例では、ゲームアプリとして、ユーザーがレベルを達成するとイベントを記録します。新しい AmazonMobileAnalyticsEvent インスタンスを作成して、「LevelComplete」イベントを作成します。

```
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);
```

セッションの記録

Xamarin iOS

アプリケーションがフォーカスを失った場合は、セッションを一時停止することができます。iOS アプリの場合、AppDelegate.cs ファイルで、DidEnterBackground および WillEnterForeground をオーバーライドして、次のスニペットに示すように MobileAnalyticsManager.PauseSesion および MobileAnalyticsManager.ResumeSession を呼び出します。

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

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

Xamarin Android

Android アプリの場合、次のコードスニペットに示すように、OnPause () メソッドでは MobileAnalyticsManager.PauseSesion を呼び出し、OnResume () メソッドでは MobileAnalyticsManager.ResumeSession を呼び出します。

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

```
}

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

デフォルトでは、5 秒以内にアプリから焦点を切り替えて戻した場合、セッションは再開されます。ユーザーが焦点から離れて 5 秒以上経過した場合は、新しいセッションが作成されます。この設定は、aws_mobile_analytics.json 設定ファイルで、「SESSION_DELTA」プロパティを新しいセッションを作成するまで待機する秒数に設定することで設定できます。

Amazon Simple Storage Service (S3)

S3 とは

Amazon Simple Storage Service (Amazon S3) を使用すると、開発者は、耐久性および耐障害性が高く、セキュアなオブジェクトストレージを使用できるようになります。Amazon S3 の使用方法は簡単です。シンプルなウェブサービスインターフェイスを使用して、ウェブ上のどこからでも容量に関係なくデータを保存、取得できます。Amazon S3 のお支払いは、実際に使用したストレージ分のみです。最低料金もセットアップするための費用も不要です。

Amazon S3 は、クラウドアプリケーション、コンテンツ配信、バックアップとアーカイブ、災害対策、ビッグデータ分析など、さまざまなユースケースにおいてコスト効率のいいオブジェクトストレージです。

AWS S3 リージョンの可用性の詳細については、「AWS サービスリージョンの可用性」を参照してください。

主なコンセプト

バケット

Amazon S3 に保存したオブジェクトはすべて、バケット内にあります。ファイルシステムでディレクトリを使用してファイルをグループ化するのと同じ方法で、バケットを使用して関連するオブジェクトをグループ化できます。バケットにはアクセス許可やバージョニングステータスなどのプロパティがあり、バケットを保存するリージョンを指定できます。

S3 バケットの詳細については、S3 開発者ガイドの「バケットの使用」を参照してください。

オブジェクト

オブジェクトは、Amazon S3 に保存するデータです。すべてのオブジェクトは、特定の AWS リージョンに作成したバケット内に存在します。

明示的に別のリージョンに移動する場合を除き、特定のリージョンに保存されたオブジェクトは、そのリージョンから移動されることはありません。たとえば、欧州 (アイルランド) リージョンに格納されたオブジェクトは、ずっとそのリージョンに置かれたままです。Amazon S3 リージョンに格納されたオブジェクトはそのリージョンに物理的に残ります。Amazon S3 ではコピーが保持される

ことも、他のいずれのリージョンに移動されることもありません。ただし、それらのオブジェクトには、必要なアクセス許可がある限り、どこからでもアクセスできます。

オブジェクトのファイル形式は任意です (画像、バックアップデータ、動画など)。オブジェクトのサイズは最大 5 TB です。バケット内のオブジェクトの数に制限はありません。

Amazon S3 にオブジェクトをアップロードする前に、バケットに対する書き込みアクセス許可が必要です。バケットのアクセス許可を設定する方法については、S3 開発者ガイドの「バケット許可の編集」を参照してください。

S3 オブジェクトの詳細については、S3 開発者ガイドの「オブジェクトの使用」を参照してください。

オブジェクトメタデータ

Amazon S3 内の各オブジェクトには、メタデータを表す一連のキーと値のペアがあります。メタデータには、以下の 2 種類があります。

- システムメタデータ - Amazon S3 によって処理されることがあります (例: Content-Type、Content-Length)。
- ユーザーメタデータ – Amazon S3 によって処理されることはありません。ユーザーメタデータはオブジェクトと共に保存され、返されます。ユーザーメタデータは最大 2 KB です。キーもその値も US-ASCII 規格に準拠している必要があります。

S3 オブジェクトのメタデータの詳細については、「オブジェクトのメタデータの編集」を参照してください。

プロジェクトのセットアップ

前提条件

アプリケーションで Amazon S3 を使用するには、プロジェクトに SDK を追加する必要があります。「AWS Mobile SDK for .NET and Xamarin をセットアップする」の手順に従って、追加します。

S3 バケットの作成

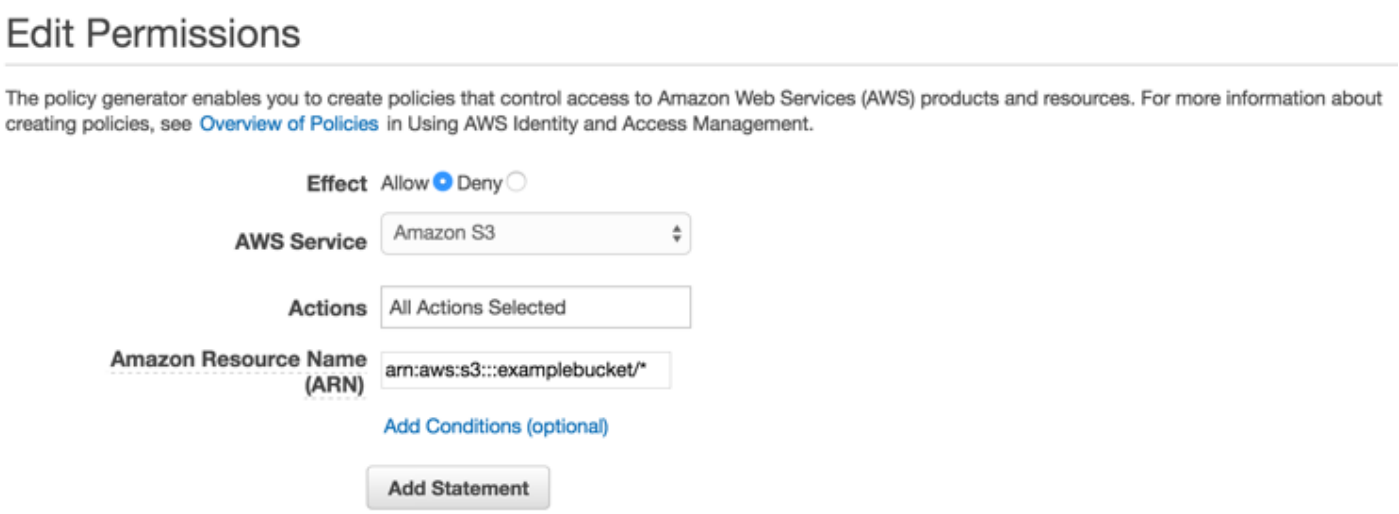
Amazon S3 は、特定のリージョンのクラウドストレージコンテナである Amazon S3 バケットにアプリケーションのリソースを保存します。Amazon S3 バケットの名前は、それぞれグローバルに一意である必要があります。Amazon S3 コンソールを使用して、バケットを作成します。

1. Amazon S3 コンソールにサインインし、[バケットを作成する] をクリックします。
2. バケット名を入力してリージョンを選択したら [作成] をクリックします。

S3 のアクセス許可を設定

デフォルトの IAM ロールのポリシーでは、Amazon Mobile Analytics および Amazon Cognito Sync へのアクセス権がアプリケーションに付与されています。Cognito ID プールから Amazon S3 にアクセスできるようにするには、ID プールのロールを変更する必要があります。

1. Identity and Access Management コンソールに移動し、左側のペインの [ロール] をクリックします。
2. 検索ボックスに ID プールの名前を入力します。2 つのロール (未認証ユーザーと認証済みユーザー) が表示されます。
3. 未認証ユーザーのロールをクリックします (unauth を ID プール名に追加)。
4. [ロールポリシーの作成] をクリックして [Policy Generator] を選択し、[選択] をクリックします。
5. [アクセス許可の編集] ページで、以下のイメージに示す設定を入力し、Amazon リソースネーム (ARN) を実際の名前に置き換えます。S3 バケットの ARN は、`arn:aws:s3:::examplebucket/*` のように表示され、バケットを配置するリージョンとバケットの名前で構成されます。以下に示す設定では、ID プールから指定したバケットのすべてのアクションにフルアクセスを付与しています。



1. [ステートメントを追加] ボタンをクリックし、[次のステップ] をクリックします。
2. ウィザードに、生成した設定が表示されます。[ポリシーの適用] をクリックします。

S3 に対するアクセス権の付与の詳細については、「Amazon S3 バケットへアクセス権を付与する」を参照してください。

(オプション) S3 リクエストの署名バージョンを設定する

Amazon S3 とのすべてのやり取りは認証されるか匿名で行われます。AWS では、署名バージョン 4 または署名バージョン 2 のアルゴリズムを使用して、サービスへの呼び出しを認証します。

2014 年 1 月以降に作成されたすべての新しい AWS リージョンでは、署名バージョン 4 のみをサポートしています。ただし、以前のリージョンの多くは、現在も署名バージョン 4 および署名バージョン 2 のリクエストに対応しています。

バケットが、こちらのページの署名バージョン 2 のリクエストをサポートしていないリージョンのいずれかにある場合は、次のように、AWSConfigsS3.UseSignatureVersion4 プロパティを「true」に設定する必要があります。

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

AWS 署名バージョンの詳細については、「リクエストの認証 (AWS 署名バージョン 4)」を参照してください。

S3 とアプリケーションの統合

Xamarin アプリケーションで S3 を操作する方法は 2 種類あります。この 2 種類のメソッドについては、次のトピックで詳しく説明します。

S3 転送ユーティリティを使用する

S3 転送ユーティリティを使用すると、Xamarin アプリケーションから S3 にファイルをアップロードおよびダウンロードしやすくなります。

TransferUtility を初期化する

S3 クライアントを作成して、次のように AWS 認証情報を渡し、続いて S3 クライアントを転送ユーティリティに渡します。

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

(オプション) TransferUtility を設定する

設定可能なオプションのプロパティは 3 種類あります。

- ConcurrentServiceRequests - ファイルのアップロード/ダウンロードに使用するアクティブスレッド数または同時 (非同期) ウェブリクエスト数を決定します。デフォルト値は 10 です。
- MinSizeBeforePartUpload - アップロードパートの最小パートサイズ (バイト単位) を取得または設定します。デフォルトでは 16 MB です。最小パートサイズを小さくすると、マルチパートアップロードは、多数の小さいパーツに分割されます。この値の設定が低すぎると、転送速度に悪影響が及び、各パートのレイテンシーおよびネットワーク通信は増加します。
- NumberOfUploadThreads - 実行スレッド数を取得または設定します。このプロパティでは、ファイルのアップロードに使用するアクティブなスレッド数を決定します。デフォルトのスレッド数は 10 です。

S3 TransferUtility クライアントを設定するには、構成オブジェクトを作成してプロパティを設定し、オブジェクトを TransferUtility コンストラクタに渡します。

```
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);
```

ファイルをダウンロードする

S3 からファイルをダウンロードするには、Transfer Utility オブジェクトの Download を呼び出し、次のパラメータを渡します。

- file - ダウンロードするファイルの文字列名。
- bucketName - ファイルのダウンロード元の S3 バケットの文字列名。
- key - ダウンロードする S3 オブジェクト (この場合はファイル) の名前を表す文字列。

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

ファイルのアップロード

S3 にファイルをアップロードするには、Transfer Utility オブジェクトの Upload を呼び出し、次のパラメータを渡します。

- file - アップロードするファイルの文字列名。
- bucketName - ファイルを保存する S3 バケットの文字列名。

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

上記のコードは、ファイルがディレクトリ (Environment.SpecialFolder.ApplicationData) にあることを前提としています。大容量ファイルをアップロードする場合は、自動的に S3 のマルチパートアップロード機能が使用されるため、スループットが向上します。

サービスレベル S3 API を使用する

S3 TransferUtility を使用することに加えて、低レベルの S3 API を使用して S3 と連携することもできます。

Amazon S3 クライアントの初期化

Amazon S3 を使用するには、まず、以前に作成した CognitoAWSCredentials インスタンスおよびリージョンを参照する AmazonS3Client インスタンスを作成する必要があります。

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

ファイルをダウンロードする

S3 からファイルをダウンロードするには、以下を行います。

```
// 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);
    }
}
```

ファイルのアップロード

S3 にファイルをアップロードするには、以下を行います。

```
// 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);
```

項目の削除

S3 の項目を削除するには、以下を行います。

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

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

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

複数の項目の削除

単一の HTTP リクエストを使用して、バケットから複数のオブジェクトを削除するには、以下を行います。

```
// 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);
    }
}
```

最大 1,000 個のキーを指定できます。

バケットの一覧表示

認証されたリクエスト送信者が所有するすべてのバケットのリストを返すには、以下を行います。

```
// 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);
}
```

オブジェクトのリスト化

S3 バケットに保存されているオブジェクトの一部またはすべて (最大 1000) を返すことができます。設定するには、バケットの読み取りアクセス許可が必要です。

```
// 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);
  }
}
```

バケットのリージョンを取得する

バケットが含まれているリージョンを取得するには、以下を行います。

```
// 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);
```

バケットのポリシーの取得

バケットのポリシーを取得するには、以下を行います。

```
// 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

Amazon DynamoDB とは

Amazon DynamoDB は、拡張性に優れた、高速な非リレーショナルデータベースサービスです。DynamoDB により、データストレージに対して低いレイテンシーと予測可能なパフォーマンスを維持しながら、従来の拡張性の限界を排除できます。

主なコンセプト

DynamoDB データモデルのコンセプトには、テーブル、項目、属性が含まれます。

テーブル

Amazon DynamoDB では、データベースはテーブルの集合体です。テーブルは項目の集合であり、各項目は属性の集合です。

リレーショナルデータベースのテーブルには、テーブル名、プライマリキー、列名とデータ型のリストなど、事前定義されたスキーマがあります。テーブルに保存されているすべてのレコードは、同じ一連の列で構成されている必要があります。それに対し、DynamoDB では、テーブルに 1 つのプライマリキーが設定されていることのみが必要であり、すべての属性名とデータ型を事前に定義する必要はありません。

テーブルの操作方法の詳細については、「DynamoDB のテーブルの操作」を参照してください。

項目と属性

DynamoDB テーブル内の個々の項目には多数の属性があります。ただし、項目サイズに対する 400KB の制限があります。項目のサイズは、属性名と値の長さの合計です (バイナリおよび UTF-8 の長さ) 。

項目の各属性は、名前と値のペアです。属性には単一値または多値のセットを指定できます。たとえば書籍項目には、題名と著者の属性を指定できます。それぞれの書籍の題名は 1 つですが、多数の著者を指定できます。多値属性は 1 つのセットであり、値の重複は許可されていません。

たとえば、製品カタログを DynamoDB に保存するとします。ID 属性をプライマリキーとして ProductCatalog テーブルを作成します。プライマリキーは各項目を一意に識別するため、テーブル内の 2 つの製品が同じ ID を持つことはできません。

項目の操作に関する詳細については、「DynamoDB の項目の操作」を参照してください。

データ型

Amazon DynamoDB では、次のデータ型をサポートしています。

- スカラー型 – 数値、文字列、バイナリ、ブール、Null。
- 多値型 – 文字列セット、数値セット、バイナリセット。
- ドキュメント型 – リスト、マップ。

データ型 (スカラー型、多値型、ドキュメント型) の詳細については、「DynamoDB データ型」を参照してください。

プライマリキー

テーブルを作成する場合には、テーブル名に加えて、テーブルのプライマリキーを指定する必要があります。プライマリキーはテーブルの各項目を一意に識別するため、テーブル内の 2 つの項目が同じキーを持つことはありません。DynamoDB では、次の 2 つのタイプのプライマリキーがサポートされています。

- ハッシュキー: プライマリキーは、ハッシュ属性という 1 つの属性で構成されています。DynamoDB では、このプライマリキー属性について、順序を指定しないハッシュインデックスを構築します。テーブルの各項目は、ハッシュキー値によって一意に識別されます。
- ハッシュキーおよびレンジキー: プライマリキーは、2 つの属性で構成されています。最初の属性はハッシュ属性で、2 番目の属性が範囲属性です。DynamoDB では、ハッシュプライマリキー属性に基づいて、順序を指定しないハッシュインデックスを構築し、レンジプライマリキー属性に基づいて、ソートされたレンジインデックスを構築します。テーブルの各項目は、ハッシュキーと範囲キーの値の組み合わせによって識別されます。2 つの項目でハッシュキー値を同じにすることは可能ですが、これらの 2 つの項目の範囲キー値は異なっている必要があります。

セカンダリインデックス

ハッシュキーおよびレンジキーを使用してテーブルを作成するときに、必要に応じてそのテーブルで 1 つ以上のセカンダリインデックスを定義できます。セカンダリインデックスでは、プライマリキーに対するクエリに加えて、代替キーを使用して、テーブル内のデータのクエリを行うことができます。

DynamoDB では、2 種類のセカンダリインデックス (ローカルセカンダリインデックスおよびグローバルセカンダリインデックス) をサポートしています。

- ローカルセカンダリインデックス: テーブルと同じハッシュキーと、異なるレンジキーを持つインデックス。
- グローバルセカンダリインデックス: テーブルと異なるハッシュキーとレンジキーを持つインデックス。

テーブルあたり最大 5 個のグローバルセカンダリインデックスと 5 個のローカルセカンダリインデックスを定義できます。詳細については、DynamoDB 開発者ガイドの「DynamoDB でのセカンダリインデックスを使用したデータアクセス性の向上」を参照してください。

クエリおよびスキャン

Amazon DynamoDB では、プライマリキーを使用した項目へのアクセスに加えて、データの検索用に Query と Scan という 2 つの API も用意されています。いくつかのベストプラクティスを理解するために、DynamoDB 開発者ガイドの「クエリおよびスキャンのガイドライン」をお読みになることをお勧めします。

Query

Query オペレーションでは、プライマリキーの属性値を使用してテーブルまたはセカンダリインデックスの項目を検索します。ハッシュキーの属性名と検索対象の個別の値を指定する必要があります。必要に応じて、レンジキーの属性名と値を指定し、比較演算子を使用して、検索結果をさらに絞り込むことができます。

サンプルクエリについては、以下を参照してください。

- ドキュメントモデルを使用する
- オブジェクト永続性モデルの使用
- DynamoDB サービスレベル API を使用する

Query の詳細については、DynamoDB 開発者ガイドの「Query」を参照してください。

Scan

Scan オペレーションでは、テーブルまたはセカンダリインデックスのすべての項目を読み込みます。デフォルトでは、Scan オペレーションはテーブルまたはインデックスのすべての項目のデータ

属性を返します。ProjectionExpression パラメータを使用し、Scan がすべての属性ではなく一部のみを返すようにできます。

サンプルの Scan については、以下を参照してください。

- ドキュメントモデルを使用する
- オブジェクト永続性モデルの使用
- DynamoDB サービスレベル API を使用する

Scan の詳細については、DynamoDB 開発者ガイドの「Scan」を参照してください。

プロジェクトのセットアップ

前提条件

アプリケーションで DynamoDB を使用するには、プロジェクトに SDK を追加する必要があります。「AWS Mobile SDK for .NET and Xamarin をセットアップする」の手順に従って、追加します。

DynamoDB テーブルの作成

テーブルを作成するには、DynamoDB コンソールに移動し、次の手順に従って操作します。

1. [テーブルの作成] をクリックします。
2. テーブルの名前を入力します。
3. プライマリキーのタイプとして、[ハッシュ] を選択します。
4. タイプを選択し、ハッシュ属性名の値を入力します。[次へ] をクリックします。
5. [Add Indexes (インデックスの追加)] ページで、グローバルセカンダリインデックスを使用する予定の場合は、[Index Type (インデックスタイプ)] に「グローバルセカンダリインデックス」を設定し、[Index Hash Key (インデックスハッシュキー)] で、セカンダリインデックスの値を入力します。これにより、プライマリインデックスとセカンダリインデックスの両方を使用して、クエリとスキャンを行うことができます。[Add Index To Table (テーブルにインデックスを追加)] をクリックし、続いて [続行] をクリックします。グローバルセカンダリインデックスを使用してスキップするには、[続行] をクリックします。
6. 読み込みおよび書き込みキャパシティーを目的のレベルに設定します。キャパシティー設定の詳細については、「Provisioned Throughput in Amazon DynamoDB」を参照してください。[次へ] をクリックします。

7. 次の画面で、通知 E メールを入力し、必要に応じてスループットアラームを作成します。[次へ] をクリックします。
8. [概要] ページで、[作成] をクリックします。DynamoDB にデータベースが作成されます。

DynamoDB のアクセス許可を設定する

DynamoDB をアプリケーションで使用するには、適切なアクセス許可を設定する必要があります。次の IAM ポリシーでは、ユーザーは特定の DynamoDB テーブルの項目を削除、取得、挿入、クエリ、スキャン、更新することができます。このポリシーは、ARN によって識別されます。

```
{
  "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"
    }
  ]
}
```

ポリシーは、IAM コンソールで変更できます。アプリケーションのニーズに応じて、許可されているアクションを追加または削除する必要があります。

IAM ポリシーの詳細については、「IAM の使用」を参照してください。

DynamoDB 固有のポリシーの詳細については、DynamoDB 開発者ガイドの「IAM を使用して DynamoDB リソースへのアクセスをコントロールする」を参照してください。

DynamoDB とアプリケーションの統合

AWS Mobile SDK for .NET and Xamarin には、DynamoDB を操作するための高レベルのライブラリが用意されています。低レベルの DynamoDB API に対してリクエストすることもできますが、ほとんどのユースケースにおいて、ハイレベルのライブラリが推奨されています。特

に、AmazonDynamoDBClient は高レベルライブラリの便利な部分です。このクラスを使用することで、さまざまな (CRUD) 操作の作成、読み取り、更新、削除、およびクエリの実行を行うことができます。

AWS Mobile SDK for .NET and Xamarin では、DynamoDB を操作するために、AWS SDK for .NET から API を使用して呼び出すことができます。API はすべて、AWSSDK.dll で利用できます。AWS SDK for .NET のダウンロードの詳細については、「AWS SDK for .NET」を参照してください。

Xamarin アプリケーションで DynamoDB を操作する方法は、3 種類あります。

- ドキュメントモデル: この API には、お客様のプログラミングタスクを簡素化する低レベル DyanmoDB API のラッパークラスが用意されています。テーブルとドキュメントが主要なラッパークラスです。ドキュメントモデルは、項目の作成、取得、更新、削除などのデータオペレーションに使用できます。この API は、Amazon.DynamoDB.DocumentModel 名前空間で利用できます。
- オブジェクト永続性モデル: オブジェクト永続性 API では、クライアント側のクラスを DynamoDB テーブルにマッピングすることができます。各オブジェクトインスタンスが、対応するテーブルの項目にマッピングされます。この API 内にある DynamoDBContext クラスによって、クライアント側オブジェクトをテーブルに保存し、項目をオブジェクトとして取得し、クエリおよびスキャンを実行するメソッドが提供されます。オブジェクト永続性モデルは、項目の作成、取得、更新、削除などのデータオペレーションに使用できます。まず、サービスクライアント API を使用してテーブルを作成する必要があります。次に、オブジェクト永続性モデルを使用して、クラスをテーブルにマッピングします。この API は、Amazon.DynamoDB.DataModel 名前空間で利用できます。
- サービスクライアント API: これは、プロトコルレベルの API であり、DynamoDB API に緊密にマッピングされます。この低レベル API は、テーブルおよび項目のすべてのオペレーション (テーブルおよび項目の作成、更新、削除など) に使用できます。テーブルに対するクエリおよびスキャンも可能です。この API は、Amazon.DynamoDB 名前空間で利用できます。

これらの 3 つのモデルは、次のトピックで詳しく説明します。

ドキュメントモデルの使用

ドキュメントモデルは、低レベル .NET API の周囲にラッパークラスを提供しています。テーブルとドキュメントが主要なラッパークラスです。ドキュメントモデルを使用して、項目を作成、検索、更新、および削除することができます。テーブルを作成、更新、および削除するには、低レベル API を使用する必要があります。低レベル API の使用方法については、「DynamoDB サービスレベル

API を使用する」を参照してください。低レベル API は、Amazon.DynamoDB.DocumentModel 名前空間で利用できます。

ドキュメントモデルの詳細については、「.NET ドキュメントモデル」を参照してください。

DynamoDB クライアントの作成

DynamoDB クライアントを作成するには、以下を行います。

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

CRUD オペレーション

項目の保存

項目の作成:

```
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";
```

項目を DynamoDB テーブルに保存:

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

項目の取得

項目を取得するには、以下を行います。

```
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);
}
```

項目の更新

項目を更新するには:

```
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);
}
```

項目を条件付きで更新するには:

```
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);
}
```

項目の削除

項目を削除するには:

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

```
  await books.DeleteItemAsync(id);
}
```

クエリおよびスキャン

作成者が「Mark Twain」であるすべての書籍を検索して取得するには、以下を行います。

```
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);
  });
}
```

以下のスキャンのサンプルコードでは、テーブルに含まれるすべての本を返します。

```
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);
      });
}
```

オブジェクト永続性モデルの使用

AWS Mobile SDK for .NET and Xamarin には、クライアント側クラスを DynamoDB テーブルにマッピングできる、オブジェクト永続性モデルが用意されています。各オブジェクトインスタンスが、対応するテーブルの項目にマッピングされます。クライアント側オブジェクトをテーブルに保存するた

めに、オブジェクト永続性モデルでは、DynamoDB のエントリポイントとなる DynamoDBContext クラスを使用できます。このクラスでは、DynamoDB に接続してテーブルにアクセスし、各種の CRUD オペレーションやクエリを実行することができます。

オブジェクト永続性モデルには、テーブルを作成、更新、または削除するための API はありません。データオペレーションだけが可能になっています。テーブルを作成、更新、および削除するには、低レベル API を使用する必要があります。低レベル API の使用方法については、「DynamoDB サービスレベル API を使用する」を参照してください。

概要

オブジェクト永続性モデルには、クライアント側クラスをテーブルにマッピングし、プロパティ/フィールドを属性にマッピングする、属性のセットが用意されています。オブジェクト永続性モデルでは、クラスプロパティとテーブル属性との間で、明示的なマッピングとデフォルトのマッピングの両方がサポートされています。

- 明示的なマッピング: プライマリキーにプロパティをマッピングするには、オブジェクト永続性モデル属性の DynamoDBHashKey および DynamoDBRangeKey を使用する必要があります。さらに、非プライマリキー属性については、クラス内のプロパティ名と、マッピング先の対応するテーブル属性が同じでない場合は、DynamoDBProperty 属性を明示的に追加してマッピングを定義する必要があります。
- デフォルトのマッピング - デフォルトでは、オブジェクト永続性モデルによって、クラスプロパティがテーブル内の同じ名前の属性にマッピングされます。

すべてのクラスプロパティをマッピングする必要はありません。これらのプロパティを特定するには、DynamoDBIgnore 属性を追加します。オブジェクトのインスタンスを保存して取得すると、この属性でマークされたプロパティはすべて省略されます。

サポートされるデータ型

オブジェクト永続性モデルでは、プリミティブな .NET データ型、コレクション、および任意のデータ型のセットがサポートされています。このモデルでは、次のプリミティブデータ型がサポートされています。

- ブール
- バイト
- char

- DateTime
- 小数点、倍精度浮動小数点型、浮動小数点型
- Int16、Int32、Int64
- SByte
- 文字列
- UInt16、UInt32、UInt64

オブジェクト永続性モデルでは、次の制限の下で .NET コレクション型もサポートされています。

- コレクション型は、ICollection インターフェイスを実装する必要があります。
- コレクション型は、サポートされているプリミティブ型で構成されている必要があります。たとえば、ICollection<string>、ICollection<bool> を参照してください。
- コレクション型では、パラメータがないコンストラクタを使用できなければなりません。

オブジェクト永続性モデルの詳細については、「.NET オブジェクト永続性モデル」を参照してください。

DynamoDB クライアントの作成

DynamoDB クライアントを作成するには、以下を行います。

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

CRUD オペレーション

オブジェクトの保存

オブジェクトを作成します。

```
[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 テーブルにオブジェクトを保存します。

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

オブジェクトの取得

オブジェクトを取得するには:

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

オブジェクトを更新

オブジェクトを更新するには:

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

オブジェクトの削除

オブジェクトを削除するには:

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

クエリおよびスキャン

作成者が「Charles Dickens」であるすべての書籍を検索して取得するには:

```
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());
  });
}
```

以下のスキャンのサンプルコードでは、テーブルに含まれるすべての本を返します。

```
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 サービスレベル API を使用する

Dynamo サービスレベルの API を使用して、テーブルを作成、更新、削除することができます。この API を使用して、テーブル内の項目に対して、一般的な作成、読み込み、更新、削除 (CRUD) のオペレーションを実行することもできます。

DynamoDB クライアントの作成

DynamoDB クライアントを作成するには、以下を行います。

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

CRUD オペレーション

項目の保存

DynamoDB テーブルに項目を保存するには、以下を行います。

```
// 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);
```

項目の取得

項目を取得するには、以下を行います。

```
// 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"]);
}
```

項目の更新

項目を更新するには:

```
// 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);
```

項目の削除

項目を削除するには:

```
// 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);
```

クエリおよびスキャン

作成者が「Mark Twain」であるすべての書籍を検索して取得するには、以下を行います。

```
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);
    });
```

```
  }
}
```

以下のスキャンのサンプルコードでは、テーブルに含まれるすべての本を返します。

```
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)

SNS および AWS Mobile SDK for .NET and Xamarin を使用して、モバイルプッシュ通知を受信するアプリケーションを書き込むことができます。SNS の詳細については、「Amazon Simple Notification Service」を参照してください。

主なコンセプト

Amazon SNS を使用することで、アプリケーションやエンドユーザーは、デバイスが異なっても、モバイルプッシュ通知 (Apple、Google、Kindle Fire デバイス)、HTTP/HTTPS、E メール/JSON 形式のメール、SMS、Amazon Simple Queue Service (SQS) キュー、AWS Lambda 関数を使用して通知を受け取ることができます。SNS を使用すると、別々にメッセージを送信したり、1 つのトピックに受信登録されている多数の受信者にメッセージをファンアウトしたりできます。

トピック

トピックは、受信者が同じ通知の同一コピーを動的に購読できるようにする「アクセスポイント」です。1 つのトピックで複数のエンドポイントタイプへの配信をサポートできます。たとえば、iOS、Android、および SMS の受取人をまとめて 1 つのグループにすることができます。

サブスクリプション

トピックに対して発行されたメッセージを受信するには、そのトピックへのエンドポイントを受信する必要があります。エンドポイントは、Amazon SNS から通知メッセージを受信できるモバイルアプリ、ウェブサーバー、E メールアドレス、または Amazon SQS キューを指します。任意のトピックにエンドポイントをサブスクライブし、サブスクリプションが確認されると、エンドポイントはそのトピックに発行されたすべてのメッセージを受信するようになります。

公開

トピックに発行すると、適切に書式設定されたメッセージのコピーが、SNS よりそのトピックの各受信者に配信されます。モバイルプッシュ通知では、エンドポイントに直接発行するか、エンドポイントをトピックに受信登録できます。

プロジェクトのセットアップ

前提条件

アプリケーションで SNS を使用するには、プロジェクトに SDK を追加する必要があります。「AWS Mobile SDK for .NET and Xamarin をセットアップする」の手順に従って、追加します。

SNS のアクセス許可を設定

SNS のアクセス権限の設定の詳細については、「Amazon SNS トピックへのアクセスの管理」を参照してください。

SNS の NuGet パッケージをプロジェクトに追加する

「AWS Mobile SDK for .NET and Xamarin をセットアップする」のステップ 4 に従って、Amazon Simple Notification Service の NuGet パッケージをプロジェクトに追加します。

SNS とアプリケーションを統合する

Xamarin アプリケーションで SNS を操作する方法は多数あります。

プッシュ通知の送信 (Xamarin Android)

このドキュメントでは、Amazon Simple Notification Service (SNS) および AWS Mobile SDK for .NET and Xamarin を使用して、Xamarin Android アプリケーションにプッシュ通知を送信する方法について説明します。

プロジェクトのセットアップ

前提条件

このチュートリアルを開始する前に、必ず「AWS Mobile SDK for .NET and Xamarin をセットアップする」の手順をすべて完了する必要があります。

SNS のアクセス許可を設定

AWS Mobile SDK for .NET and Xamarin をセットアップするのステップ 2 に従って、以下のポリシーをアプリケーションのロールにアタッチします。これにより、SNS にアクセスするための適切な権限がアプリケーションに付与されます。

1. IAM コンソールに移動し、設定する IAM ロールを選択します。
2. [ポリシーのアタッチ] をクリック後、AmazonSNSFullAccess ポリシーを選択し、[ポリシーのアタッチ] をクリックします。

> **Warning**
>
> AmazonSNSFullAccess を本番稼働用環境で使用することは推奨されていません。すばやく起動して実行できるように、こちらの環境を使用します。IAM ロールのアクセス許可を指定する方法については、「IAM ロールのアクセス許可の概要」を参照してください。

Google Cloud のプッシュ通知を有効にする

まず、新しい Google API プロジェクトを追加します。

1. Google 開発者コンソールに移動します。
2. [Create Project] をクリックします。
3. [New Project] ボックスにプロジェクト名を入力し、プロジェクト ID をメモしたら (後で使用します)、[Create] をクリックします。

次に、プロジェクトの Google クラウド メッセージング (GCM) サービスを有効にします。

1. Google 開発者コンソールで、新しいプロジェクトが既に選択されています。選択されていない場合は、ページ上部にあるドロップダウンでプロジェクトを選択します。
2. ページ左側のサイドバーから [APIs & auth (API と Auth)] を選択します。
3. 検索ボックスに「Android 用 Google クラウドメッセージング」と入力し、Google Cloud Messaging for Android リンクをクリックします。
4. [Enable API (API を有効化)] をクリックします。

最後に、API キーを取得します。

1. Google 開発者コンソールで、[APIs & auth (API と Auth)]>[Credentials (認証情報)] の順に選択します。
2. [Public API access (パブリック API アクセス)] で、[Create new key (新しいキーを作成)] をクリックします。

3. [Create a new key (新しいキーの作成)] ダイアログで、[Server key (サーバーキー)] をクリックします。
4. 表示されたダイアログボックスで [Create (作成)] を選択し、表示された API キーをコピーします。この API キーは、後で認証を実行するために使用します。

SNS コンソールでプロジェクト ID を使用してプラットフォーム ARN を作成する

1. SNS コンソールに移動します。
2. 画面の左側にある [アプリケーション] をクリックします。
3. [プラットフォームアプリケーションの作成] をクリックして、新しい SNS プラットフォームアプリケーションを作成します。
4. アプリケーション名を入力します。
5. [Push notification platform] で [Google Cloud Messaging (GCM)] を選択します。
6. [API キー] と表示されているテキストボックスに API キーを貼り付けます。
7. [プラットフォームアプリケーションの作成] をクリックします。
8. 作成したプラットフォームアプリケーションを選択して、アプリケーション ARN をコピーします。

SNS の NuGet パッケージをプロジェクトに追加する

「AWS Mobile SDK for .NET and Xamarin をセットアップする」のステップ 4 に従って、Amazon Simple Notification Service の NuGet パッケージをプロジェクトに追加します。

SNS クライアントを作成する

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

アプリケーションにリモート通知を登録する

Android にリモート通知を登録するには、BroadcastReceiver を作成する必要があります。これにより、Google クラウドメッセージを受信できるようになります。以下のパッケージ名を変更します。ここで、次のように表示されます。

```
[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);
      }
}
```

以下は、BroadcastReceiver よりプッシュ通知を受け取り、デバイスの通知バーに通知を表示するサービスです。

```
[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 コンソールからエンドポイントにメッセージを送信する

1. [SNS コンソール] > [アプリケーション] の順に移動します。
2. プラットフォームアプリケーションを選択し、エンドポイントを選択したら、[エンドポイントへの発行] をクリックします。
3. テキストボックスにテキストメッセージを入力し、[メッセージの発行] をクリックしてメッセージを発行します。

プッシュ通知の送信 (Xamarin iOS)

このドキュメントでは、Amazon Simple Notification Service (SNS) および AWS Mobile SDK for .NET and Xamarin を使用して、Xamarin iOS アプリケーションにプッシュ通知を送信する方法について説明します。

プロジェクトのセットアップ

前提条件

このチュートリアルを開始する前に、必ず「AWS Mobile SDK for .NET and Xamarin をセットアップする」の手順をすべて完了する必要があります。

SNS のアクセス許可を設定

AWS Mobile SDK for .NET and Xamarin をセットアップするのステップ 2 に従って、以下のポリシーをアプリケーションのロールにアタッチします。これにより、SNS にアクセスするための適切な権限がアプリケーションに付与されます。

1. IAM コンソールに移動し、設定する IAM ロールを選択します。
2. [ポリシーのアタッチ] をクリック後、AmazonSNSFullAccess ポリシーを選択し、[ポリシーのアタッチ] をクリックします。

> **Warning**
>
> AmazonSNSFullAccess を本番稼働用環境で使用することは推奨されていません。すばやく起動して実行できるように、こちらの環境を使用します。IAM ロールのアクセス許可を指定する方法については、「IAM ロールのアクセス許可の概要」を参照してください。

Apple iOS 開発者プログラムのメンバーシップを取得する

プッシュ通知を受け取るには、物理デバイスでアプリを実行する必要があります。デバイスでアプリを実行するには、Apple iOS 開発者プログラムのメンバーシップのメンバーシップが必要です。メンバーシップを取得したら、Xcode を使用して署名 ID を生成できます。詳細については、Apple の App ディストリビューションクイックスタートドキュメントを参照してください。

iOS 証明書を作成する

まず、iOS 証明書を作成する必要があります。次に、プッシュ通知用に設定されたプロビジョニングプロファイルを作成します。そのためには、次の操作を行います。

1. Apple 開発者メンバーセンターに移動し、[Certificates, Identifiers & Profiles] をクリックします。
2. [iOS Apps] の [Identifiers] をクリック後、その ウェブページの右上隅の + ボタンをクリックして新しい iOS App ID を追加し、App ID の説明を入力します。
3. [Add ID Suffix (ID サフィックスを追加)] セクションまで下にスクロールし、[Explicit App ID (明示的な App ID)] を選択して、バンドル ID を入力します。
4. [App Services (アプリサービス)] セクションまで下にスクロールし、[Push Notifications (プッシュ通知)] を選択します。
5. [次へ] をクリックします。
6. 送信 をクリックします。
7. [Done (完了)] をクリックします。
8. 作成した App ID を選択し、[Edit] をクリックします。
9. [Push Notifications (プッシュ通知)] セクションまで下にスクロールします。[Development SSL Certificate] の [Create Certificate] を選択します。
10. 手順に従って、証明書署名リクエスト (CSR) を作成してアップロードし、Apple Notification Service (APNS) との通信に使用する SSL 証明書をダウンロードします。
11. [Certificates, Identifiers & Profiles] ページに戻ります。[Provisioning Profiles] の [All] をクリックします。
12. 右上隅の + ボタンをクリックし、新しいプロビジョニングプロファイルを追加します。
13. [iOS App Development] を選択し、[Continue] をクリックします。
14. App ID を選択し、[Continue] をクリックします。
15. 開発者の証明書を選択して、[Continue] をクリックします。
16. デバイスを選択し、[Continue] をクリックします。
17. プロファイル名を入力し、[Generate] をクリックします。
18. プロビジョニングプロファイルをダウンロードしてダブルクリックし、インストールします。

プッシュ通知用に設定されたプロファイルのプロビジョニングに関する詳細については、Apple のConfiguring Push Notifications ドキュメントを参照してください。

SNS コンソールで証明書を使用してプラットフォーム ARN を作成する

1. KeyChain access アプリを実行して、画面左下の [My Certificates] を選択したら、APNS に接続するために生成した SSL 証明書を右クリックして、[Export] を選択します。証明書を保護するためのファイル名とパスワードを指定するよう求められます。証明書は、P12 ファイルに保存されます。
2. SNS コンソールに移動し、画面左側の [Applications] をクリックします。
3. [プラットフォームアプリケーションの作成] をクリックして、新しい SNS プラットフォームアプリケーションを作成します。
4. アプリケーション名を入力します。
5. [Push notification platform] の [Apple Development] を選択します。
6. [ファイルの選択] をクリックし、SSL 証明書をエクスポートした際に作成した P12 ファイルを選択します。
7. SSL 証明書をエクスポートした際に指定したパスワードを入力し、[認証情報をファイルから読み込み] をクリックします。
8. [プラットフォームアプリケーションの作成] をクリックします。
9. 作成したプラットフォームアプリケーションを選択して、アプリケーション ARN をコピーします。このアプリケーション ARN は、この後の手順で必要になります。

SNS の NuGet パッケージをプロジェクトに追加する

「AWS Mobile SDK for .NET and Xamarin をセットアップする」のステップ 4 に従って、Amazon Simple Notification Service の NuGet パッケージをプロジェクトに追加します。

SNS クライアントを作成する

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

アプリケーションにリモート通知を登録する

アプリケーションを登録するには、以下に示すように、UIApplication オブジェクトの RegisterForRemoteNotifications を呼び出します。AppDelegate.cs に以下のコードを配置し、以下に表示されているプラットフォームアプリケーションの ARN を挿入します。

```
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 コンソールからエンドポイントにメッセージを送信する

1. [SNS コンソール] > [アプリケーション] の順に移動します。
2. プラットフォームアプリケーションを選択し、エンドポイントを選択したら、[エンドポイントへの発行] をクリックします。
3. テキストボックスにテキストメッセージを入力し、[メッセージの発行] をクリックしてメッセージを発行します。

SMS 通知の送受信

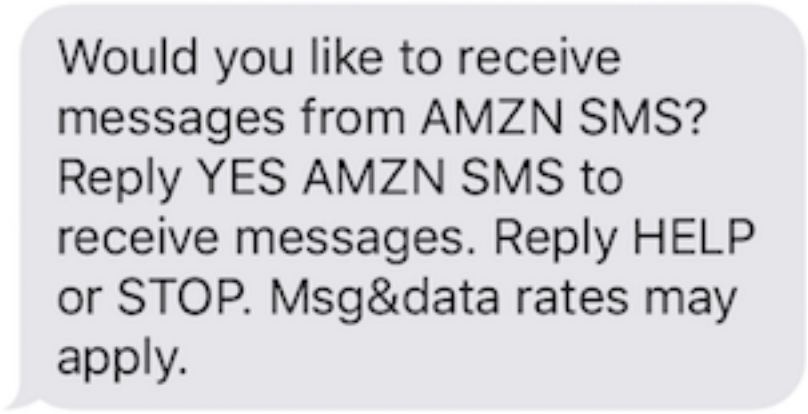
Amazon Simple Notification Service (Amazon SNS) を使用して、SMS 対応の携帯電話やスマートフォンとの間でショートメッセージサービス (SMS) 通知を送受信することができます。

> **Note**
>
> SMS 通知は、現在、米国内の電話番号でサポートされています。SMS メッセージは、米国東部 (バージニア北部) リージョンで作成されたトピックからのみ送信できます。ただし、米国東部 (バージニア北部) リージョンで作成したトピックに、他の任意のリージョンからメッセージを発行できます。

トピックの作成

トピックを作成するには:

1. Amazon SNS コンソールで、[新しいトピックの作成] をクリックします。[Create new topic] ダイアログボックスが表示されます。
2. [Topic name] ボックスにトピック名を入力します。
3. [Display name] ボックスに、表示名を入力します。トピックには、表示名が割り当てられている必要があります。表示名の先頭の 10 文字が、テキストメッセージプレフィックスの最初の部分として使用されるためです。入力した表示名が、SNS からユーザーへ送信される確認メッセージに表示されます (以下の表示名は「AMZN SMS」)。



1. [トピックの作成] をクリックします。新しいトピックが [Topics] ページに表示されます。
2. 新しいトピックを選択し、トピックの ARN をクリックします。[トピックの詳細] ページが表示されます。
3. 次のステップで、トピックに受信登録する際に必要になるため、トピック ARN をコピーします。

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

SMS プロトコルを使用してトピックに受信登録する

SNS クライアントを作成し、ID プールの認証情報オブジェクトおよびリージョンを渡します。

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

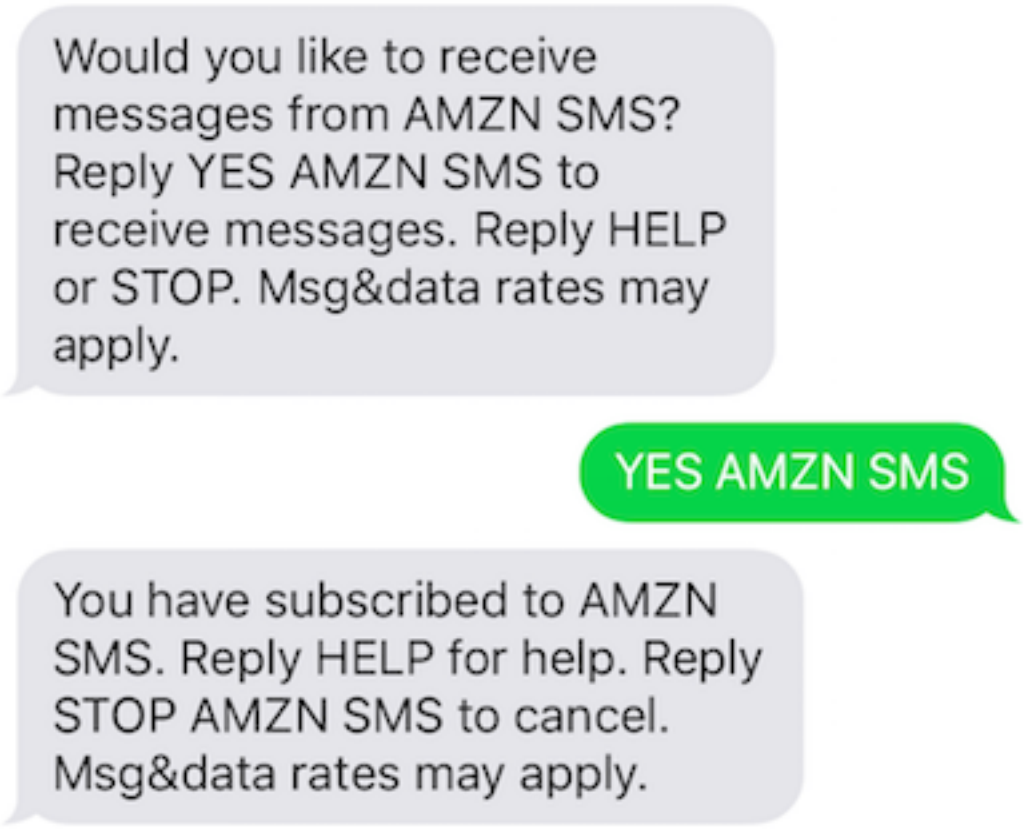
トピックに受信登録するには、SubscribeAsync を呼び出し、受信登録するトピックの ARN (例: プロトコル (「sms」)) と電話番号を渡します。

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

受信登録の ARN が受信登録の応答オブジェクトで送信されます。受信登録の ARN は次のようになります。

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

デバイスがトピックに受信登録すると、SNS よりデバイスに確認メッセージが送信されるため、ユーザーは、以下に示すように、通知を受け取ることを確認する必要があります。

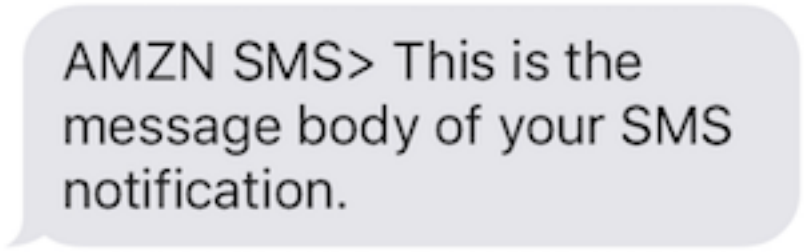


トピックに受信登録したら、そのトピックに発行したときに SMS メッセージが受信されます。

メッセージの発行

トピックにメッセージを発行するには:

1. AWS マネジメントコンソールにサインインし、Amazon SNS コンソールを開きます。
2. 左のナビゲーションペインで、[トピック] をクリックし、発行先のトピックを選択します。
3. [トピックの発行] をクリックします。
4. [Subject] ボックスに、件名を入力します。
5. [Message] ボックスにメッセージを入力します。Amazon SNS は、[Subject] ボックスにテキストが入力されている場合を除き、[Message] ボックスに入力されているテキストを SMS の受信者に送信します。Amazon SNS は送信されるすべての SMS メッセージに表示名プレフィックスを含めるため、表示名プレフィックスとメッセージペイロードの合計が ASCII 文字で 140 個または Unicode 文字で 70 個を超えることはできません。この制限を超えたメッセージは、Amazon SNS によって切り捨てられます。
6. [メッセージの発行] をクリックします。Amazon SNS で確認ダイアログボックスが表示されます。次に示すように、SMS メッセージが SMS 対応デバイスに表示されます。



HTTP/HTTPS エンドポイントへのメッセージの送信

Amazon SNS を使用して、1 つ以上の HTTP または HTTPS エンドポイントに通知メッセージを送信できます。手順は次のとおりです。

1. Amazon SNS メッセージを受け取るようにエンドポイントを設定します。
2. HTTP/HTTPS エンドポイントをトピックに受信登録します。
3. サブスクリプションを確認します。
4. トピックに通知を発行します。Amazon SNS より、受信登録したエンドポイントに、通知の内容を送信する HTTP POST リクエストが送信されます。

Amazon SNS メッセージを受け取るように HTTP/HTTPS エンドポイントを設定する

「HTTP/HTTPS エンドポイントへの Amazon SNS メッセージの送信」のステップ 1 の手順に従って、エンドポイントを設定します。

HTTP/HTTPS エンドポイントを Amazon SNS トピックに受信登録する

SNS クライアントを作成し、ID プールの認証情報オブジェクトおよびリージョンを渡します。

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

トピックを通じて HTTP または HTTPS エンドポイントにメッセージを送信するには、エンドポイントを Amazon SNS トピックにサブスクライブする必要があります。エンドポイントを指定するには、その URL を使用します。

```
var response = await snsClient.SubscribeAsync(
  "topicArn",
```

```
  "http",  /* "http" or "https" */
  "endpointUrl" /* endpoint url beginning with http or https */
);
```

サブスクリプションの確認

エンドポイントを受信登録すると、Amazon SNS はエンドポイントに受信登録の確認メッセージを送信します。エンドポイントのコードは、受信登録の確認メッセージから SubscribeURL 値を取得し、SubscribeURL そのもので指定された場所にアクセスするか、この場所を利用可能にして、手動で SubscribeURL にアクセスできるようにする必要があります (例: ウェブブラウザを使用している場合)。

Amazon SNS は、受信登録が確認されるまでエンドポイントにメッセージを送信しません。SubscribeURL にアクセスすると、応答にはサブスクリプションの ARN を指定する SubscriptionArn 要素を含む XML ドキュメントが含まれます。

HTTP/HTTPS エンドポイントへのメッセージの送信

トピックに発行することで、トピックの受信登録にメッセージを送信できます。PublishAsync を呼び出し、トピック ARN とメッセージを渡します。

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

SNS のトラブルシューティング

Amazon SNS コンソールで配信ステータスを使用する

Amazon SNS コンソールには、配信ステータス機能が搭載されており、モバイルプッシュ通知プラットフォーム (Apple (APNS)、Google (GCM)、Amazon (ADM)、Windows (WNS および MPNS)、Baidu) へのメッセージ配信結果 (成功および失敗) に関するフィードバックを収集することができます。

他にも、Amazon SNS のドウェル時間などの重要な情報が表示されます。この情報は、Amazon CloudWatch ロググループにキャプチャされます。このグループは、Amazon SNS コンソールまたは Amazon SNS API からこの機能を有効にした場合に、Amazon SNS によって自動的に作成されます。

配信ステータス機能の使用手順については、AWS モバイルブログの「Amazon SNS の配信ステータス機能を使用する」を参照してください。

AWS Mobile SDK for .NET and Xamarin を使用したのベストプラクティス

AWS Mobile SDK for .NET and Xamarin を使用する際に役立つ、いくつかの基礎とベストプラクティスがあります。

- Amazon Cognito を使用して、アプリケーションで認証情報をハードコードする代わりに AWS 認証情報を取得します。アプリケーションでハードウェアコードをハードコードすると、公開される可能性があります。他人がお客様の認証情報を使用して AWS を呼び出すことができます。Amazon Cognito を使用して AWS 認証情報を取得する方法については、「AWS Mobile SDK for .NET and Xamarin の設定」を参照してください。
- S3 の使用に関するベストプラクティスについては、「AWS ブログの記事」を参照してください。
- DynamoDB の使用に関するベストプラクティスについては、DynamoDB 開発者ガイドの「DynamoDB のベストプラクティス」を参照してください。

当社は常にお客様の成功を支援するとともに、フィードバックを歓迎しています。AWS フォーラムに投稿するか、Github で問題を提議してください。

AWS サービスドキュメントのライブラリ

AWS Mobile SDK for .NET and Xamarin のすべてのサービスには、個別のデベロッパーガイドとサービス API リファレンスがあり、役に立つ参考情報を提供します。

Amazon Cognito ID

- Cognito 開発者ガイド
- Cognito ID の サービス API リファレンス

Amazon Cognito Sync

- Cognito 開発者ガイド
- Cognito Sync の サービス API リファレンス

Amazon Mobile Analytics

- Mobile Analytics 開発者ガイド
- Mobile Analytics サービス API リファレンス

Amazon S3

- S3 開発者ガイド
- S3 入門ガイド
- S3 サービス API リファレンス

Amazon DynamoDB

- (DynamoDB 開発者ガイド) を参照してください
- DynamoDB 入門ガイド
- DynamoDB サービス API リファレンス

Amazon Simple Notification Service (SNS)

- SNS 開発者ガイド
- SNS サービス API リファレンス

その他の役立つリンク

- AWS 用語集
- AWS 認証情報について

トラブルシューティング

このトピックでは、AWS Mobile SDK for .NET and Xamarin を使用する際に発生する可能性のある問題のトラブルシューティングについて説明します。

IAM ロールに必要なアクセス許可が付与されていることを確認する

アプリから AWS サービスを呼び出す際は、Cognito ID プールの ID を使用する必要があります。プールの各 ID は、IAM (Identity and Access Management) ロールに関連付いています。

ロールには、1 つ以上のポリシーが関連付けられており、そのロールに割り当てられたユーザーがアクセスできる AWS リソースを指定しています。デフォルトでは、2 種類のロール (認証済みユーザー用と未認証ユーザー用) が作成されています。

既存のポリシーファイルを変更するか、新しいポリシーファイルをアプリで必要なアクセス許可に関連付ける必要があります。認証済みユーザーと未認証ユーザーがいずれもアプリケーションにアクセスできるようにするには、アプリで必要な AWS リソースにアクセスするためのアクセス許可を両方のロールに付与する必要があります。

以下のポリシーファイルは、S3 バケットへのアクセス権限を付与する方法を示しています。

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

以下のポリシーファイルは、DynamoDB データベースへのアクセス権限を付与する方法を示しています。

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

ポリシーの指定に関する詳細については、「IAM ポリシー」を参照してください。

HTTP プロキシデバッガーを使用する

アプリで呼び出す AWS サービスに HTTP または HTTPS エンドポイントがある場合、HTTP/HTTPS プロキシデバッガーを使用してリクエストおよびレスポンスを表示し、動作の詳細を確認できます。利用できる HTTP プロキシデバッガーは多数あります。

- Charles - Windows および OSX 用のウェブデバッギングプロキシ
- Fiddler - Windows 用のウェブデバッギングプロキシ

Charles と Fiddler ではいずれも、一部の設定で、SSL 暗号化トラフィックを表示できるようにする必要があります。これらのツールの詳細については、該当するドキュメントをお読み下さい。ウェブデバッギングプロキシを使用しており、暗号化トラフィックを表示するように設定できない場合は、aws_endpoints_json ファイルを開き、デバッグする必要のある AWS サービスの HTTP タグを true に設定します。

ドキュメント履歴

以下の表に、AWS Mobile SDK for .NET and Xamarin の前回のリリース以降に行われた、ドキュメントの重要な変更を示します。

- API バージョン: 2015-08-27
- 前回のドキュメントの更新: 2021 年 02 月 23 日

変更	API バージョン	説明	リリース日
アーカイブ済み	2015-08-27	AWSの Mobile SDK for Xamarin がAWS SDK for .NETに含まれるようになりました。このガイドでは、Mobile SDK for Xamarin のアーカイブバージョンについて説明します。	2021 年 2 月 12 日
GA リリース	2015-08-27	GA リリース	2015-08-27
ベータリリース	2015-07-28	ベータリリース	rn2015-07-28