Xamarin 开发人员指南

AWS Mobile SDK

AWS Mobile SDK: Xamarin 开发人员指南

Table of Contents

适用于 Xamarin 的 AWS 移动 SDK 现已包含在。 适用于 .NET 的 AWS SDK本指南参考了适用于 Xamarin 的 Mobile SDK 的存档版本。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。

什么是适用于 .NET 和 Xamarin 的 AWS Mobile SDK？

适用于 Xamarin 的 AWS 移动 SDK 包含在。 适用于 .NET 的 SDK有关更多信息，请参见适用于 .NET 的 AWS SDK 开发人员指南。

本指南不再更新，它引用了适用于 Xamarin 的 Mobile SDK 的存档版本。

相关指南和主题

- 对于前端和移动应用程序开发，建议使用 AWS Amplify。
- 有关在 Xamarin 应用程序中 适用于 .NET 的 AWS SDK 使用时的特殊注意事项，请参阅开发人员指南中的 Xamarin 支持的特殊注意事项。适用于 .NET 的 AWS SDK
- 为了便于参考，您可以在上找到适用于 Xamarin 的AWS 移动 SDK 的存档版本。 GitHub

存档的参考内容

存档的适用于 .NET 和 Xamarin 的 AWS Mobile SDK 提供了一组 .NET 库、代码示例和文档，可帮助开发人员构建适合以下平台的互连移动应用程序：

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

使用适用于.NET 和 Xamarin 的 AWS 移动软件开发工具包编写的移动应用程序调用原生 APIs 平台，因此它们具有原生应用程序的外观和感觉。软件开发工具包中的.NET 库提供了围绕 AWS RES APIs T 的 C# 封装器。

适用于 .NET 和 Xamarin 的 AWS Mobile SDK 中包含哪些组件？

支持的 AWS 服务当前包括（但不限于）以下内容：

- Amazon Cognito
- Amazon S3

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

借助上述服务，您可以验证用户身份，保存玩家和游戏数据，将对象保存在云中，接收推送通知，及收集和分析使用率数据。

此外，通过适用于 .NET 和 Xamarin 的 AWS Mobile SDK，您还可以使用适用于 .NET 的 AWS SDK 所支持的大部分 AWS 服务。本开发人员指南中介绍了特定于移动开发的 AWS 服务。要详细了解适用于 .NET 的 AWS SDK，请参阅以下内容：

- 适用于 .NET 的 AWS SDK 入门指南
- 适用于 .NET 的 AWS SDK 开发人员指南
- 适用于 .NET 的 AWS SDK API 参考

兼容性

适用于 .NET 和 Xamarin 的 AWS Mobile SDK 以可移植类库 (PCL) 的形式提供。Xamarin.Android 4.10.1 和 xamarin.iOS 7.0.4 中添加了 PCL 支持。可移植库项目内置于 Visual Studio 中。

IDEs

有关使用 IDEs 存档版本的 Xamarin SDK 的更多信息，请参阅。设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK

如何获得适用于.NET 和 Xamarin 的 AWS Mobile SDK？

要获取适用于.NET 和 Xamarin 的 AWS Mobile SDK，请参阅设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK。适用于.NET 和 Xamarin 的 AWS 移动软件开发工具包以软件包形式分发。 NuGet 您可以在适用于.NET 的 AWS 软件开发工具包 NuGet或适用于.NET 的 AWS 软件开发工具包GitHub存储库中找到 AWS 服务包的完整列表。

关于 AWS 移动服务

Amazon Cognito Identity

对 AWS 的所有调用都需要 AWS 凭证。建议您使用 Amazon Cognito Identity 向应用程序提供 AWS 凭证，而不是将凭证硬编码到应用程序中。请按照设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK 中的说明操作，通过 Amazon Cognito 获取 AWS 凭证。

Cognito 还允许您使用公共登录提供商 (如 Amazon、Facebook、Twitter 和 Google) 以及支持 OpenID Connect 的提供商对用户进行身份验证。Cognito 还支持未经身份验证的用户。Cognito 提供临时凭证，这些凭证具有有限的访问权限，您可以通过 Identity and Access Management (IAM) 角色指定这些权限。通过创建与 IAM 角色相关联的身份池，可以配置 Cognito。IAM 角色指定 resources/services 您的应用程序可以访问的内容。

要开始使用 Cognito Identity，请参阅设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK。

有关 Cognito Identity 的更多信息，请参阅 Amazon Cognito Identity。

Amazon Cognito Sync

Cognito Sync 是一种 AWS 服务和客户端库，用于跨设备同步与应用程序相关的用户数据。您可以使用 Cognito Sync API，跨设备、跨登录提供商 (Amazon、Facebook、Google 及您自己的自定义身份提供商) 同步用户配置文件数据。

要开始使用 Cognito Sync，请参阅通过 Cognito Sync 同步用户数据。

有关 Cognito Sync 的更多信息，请参阅 Amazon Cognito Sync。

Mobile Analytics

借助 Amazon Mobile Analytics，您可以收集、直观地查看并了解移动应用程序的使用情况。报告提供了有关活跃用户、会话、保留率、应用程序内收入和自定义事件的指标，可以按平台和日期范围进行筛选。内置的 Amazon Mobile Analytics 可随着您业务的发展而进行扩展，并可用于收集和处理来自数百万端点的数十亿事件。

要开始使用 Mobile Analytics，请参阅通过 Amazon Mobile Analytics 跟踪应用程序使用率数据。

有关 Mobile Analytics 的更多信息，请参阅 Amazon Mobile Analytics。

Dynamo DB

Amazon DynamoDB 是一项快速、高度可扩展、高度可用且经济实惠的非关系数据库服务。DynamoDB 消除了传统上对数据存储可扩展性的限制，同时保留了低延迟性和可预测的性能。

要开始使用 Dynamo DB，请参阅通过 DynamoDB 存储和检索数据。

有关 Dynamo DB 的更多信息，请参阅 Amazon DynamoDB。

Amazon Simple Notification Service

Amazon Simple Notification Service (SNS) 是一项快速、灵活、完全托管的推送通知服务，可让您发送单独的消息或将消息群发给大量收件人。通过 Amazon Simple Notification Service，您可以将推送通知发送给移动设备用户、电子邮件收件人，甚至可以将消息发送给其他分布式服务，既简单又经济高效。

要开始使用 Xamarin iOS 版 SNS，请参阅使用 SNS 接收推送通知 (Xamarin iOS)。

要开始使用 Xamarin Android 版 SNS，请参阅使用 SNS 接收推送通知 (Xamarin Android)。

有关 SNS 的更多信息，请参阅 Amazon Simple Notification Service (SNS)。

设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK

您可以设置 .NET 和 Xamarin 的 AWS Mobile SDK 并开始构建新项目，也可以将开发工具包与现有项目集成。您还可以克隆并运行示例，以便了解该 SDK 的工作原理。请按照以下步骤操作，设置并开始使用适用于 .NET 和 Xamarin 的 AWS Mobile SDK。

先决条件

在使用适用于 .NET 和 Xamarin 的 AWS Mobile SDK 之前，您必须先执行以下操作：

- 创建一个 AWS 账户。
- 安装 Xamarin。

具备上述条件后，执行以下操作：

1. 使用 Amazon Cognito 获取 AWS 凭证。
2. 为您将在应用程序中使用的每项 AWS 服务设置所需权限。
3. 在 IDE 中创建一个新项目。
4. 安装适用于 .NET 和 Xamarin 的 AWS Mobile SDK。
5. 配置适用于 .NET 和 Xamarin 的 AWS Mobile SDK。

第 1 步：获取 AWS 凭证

要在应用程序中调用 AWS，您必须先获取 AWS 凭证。为此，您需要使用 Amazon Cognito（一项 AWS 服务），以允许您的应用程序访问软件开发工具包中的服务，而不必在应用程序中嵌入您的私有 AWS 凭证。

要开始使用 Amazon Cognito，您必须创建一个身份池。身份池用于存储特定于您账户的信息，并通过如下所示的唯一身份池 ID 来识别：

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

1. 登录 Amazon Cognito 控制台，依次选择 Manage Federated Identities (管理联合身份) 和 Create new identity pool (创建新的身份池)。

2. 输入身份池的名称，然后选中相应复选框，以启用对未经身份验证的身份的访问权限。选择 Create Pool，以创建您的身份池。
3. 选择允许以创建两个与您的身份池关联的默认角色，一个用于未经身份验证的用户，另一个用于经过身份验证的用户。这些默认角色会向 Amazon Cognito Sync 和 Amazon Mobile Analytics 提供身份池访问权限。

通常，您只对一个应用程序使用一个身份池。

身份池创建完毕后，您需创建一个 `CognitoAWSCredentials` 对象（将其传递给您的身份池 ID），然后将其传递给 AWS 客户端的构造函数，以获取 AWS 凭证，具体操作方法如下：

```
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
);
```

第 2 步：设置权限

您必须为您要在应用程序中使用的每项 AWS 服务设置权限。首先，您需要了解 AWS 查看应用程序中用户的方式。

如果有人使用您的应用程序并调用 AWS，则 AWS 会为此用户分配一个身份。您在第 1 步中创建的身份池就是 AWS 存储这些身份的地方。其中有两种类型的身份：经过身份验证的身份和未经身份验证的身份。经过身份验证的身份属于已通过公共登录提供商（例如，Facebook、Amazon、Google）进行身份验证的用户。未经身份验证的身份属于来宾用户。

每个身份都与一个 AWS Identity and Access Management 角色相关联。在第 1 步中，您创建了两个 IAM 角色，一个用于经过身份验证的用户，另一个用于未经身份验证的用户。每个 IAM 角色都有一个或多个附加到自身的策略，用于指定分配给相应角色的身份可以访问哪些 AWS 服务。例如，以下策略示例会授予对 Amazon S3 存储桶的访问权限：

```
{
```

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

要为您要在应用程序中使用的 AWS 服务设置权限，只需修改附加到相应角色的策略即可。

1. 转到 IAM 控制台并选择“Roles (角色)”。在搜索框中键入您的身份池名称。选择要配置的 IAM 角色。如果您的应用程序允许经过身份验证和未经身份验证的用户，则您需要为这两个角色授予权限。
2. 单击附加策略，选择所需的策略，然后单击附加策略。您为 IAM 角色创建的默认策略会提供对 Amazon Cognito Sync 和 Mobile Analytics 的访问权限。

有关如何创建策略或从现有策略列表中选择策略的更多信息，请参阅 IAM 策略。

步骤 3：创建新的 项目

Windows

您可以使用 Visual Studio 来开发您的应用程序。

OS X

您可以使用 Visual Studio 开发应用程序。使用 Xamarin 进行 iOS 开发需要访问 Mac 才能运行您的应用程序。有关更多信息，请参阅在 Windows 上安装 Xamarin.iOS。

> **Note**
>
> 来自 JetBrains的跨平台商业 IDE Rider 在 Windows 和 Mac 平台上都支持 Xamarin。

第 4 步：安装适用于.NET 和 Xamarin 的 AWS Mobile SDK

Windows

方法 1：使用 Package Manager 控制台安装

适用于.NET 和 Xamarin 的 AWS Mobile SDK 由一组.NET 程序集组成。要安装适用于.NET 和 Xamarin 的 AWS Mobile SDK，请在 Package Manager 控制台中为每个程序包运行 install-package 命令。例如，要安装 Cognito Identity，请运行以下命令：

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

所有项目都需要 AWS Core Runtime 和 Amazon Cognito 程序包。下表完整列出了每项服务对应的程序包名称。

服务	软件包名称
AWS Core Runtime	AWSSDK.Core
Amazon Cognito Sync	AWSSDK.CognitoSync
Amazon Cognito Identity	AWSSDK.CognitoIdentity
Amazon DynamoDB	AWSSDK.dynamo DBv2
Amazon Mobile Analytics	AWSSDK.MobileAnalytics
Amazon S3	AWSSDK.S3
Amazon SNS	AWSSDK.SimpleNotificationService

要添加预发行程序包，请在安装程序包时添加 `-Pre` 命令行参数，方法如下所示：

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

您可以在适用于.NET 的 AWS 软件开发工具包 NuGet或适用于.NET 的 AWS 软件开发工具包 GitHub 存储库中找到 AWS 服务包的完整列表。

方法 2：使用 IDE 安装

在 Visual Studio 中

1. 右键单击该项目，然后单击 “管理 NuGet包”。
2. 搜索您要添加到项目中的程序包名称。要包含预售 NuGet 套餐，请选择 “包括预售”。您可以在上的 AWS 开发工具包包中找到 AWS 服务包的完整列表 NuGet。
3. 依次选择此程序包和安装。

Mac (OS X)

在 Visual Studio 中

1. 右键单击软件包文件夹，然后选择 Add Packages (添加软件包)。
2. 搜索您要添加到项目中的程序包名称。要包含预发行 NuGet 包，请选择 “显示预发行包”。您可以在上的 AWS 开发工具包包中找到 AWS 服务包的完整列表 NuGet。
3. 选中所需软件包旁边的复选框，然后选择 Add Package (添加软件包)。

> **Important**
>
> 如果您使用可移植类库进行开发，则还必须将 AWSSDK .Core NuGet 包添加到所有源自可移植类库的项目中。

第 5 步：配置适用于.NET 和 Xamarin 的 AWS Mobile SDK

设置日志记录

您可以使用 `Amazon.AWSConfigs` 类和 `Amazon.Util.LoggingConfig` 类，配置日志记录设置。这些类可在通过 Visual Studio 中的 Nuget Package Manager 提供的 `AWSSdk.Core` 程序集中找到。您可以将日志记录设置代码放置在 `OnCreate` 文件 (适用于 Android 应用程序) 或 `MainActivity.cs` 文件 (适用于 iOS 应用程序) 的 `AppDelegate.cs` 方法中。您还应将 `using Amazon` 和 `using Amazon.Util` 语句添加到 .cs 文件中。

按以下所示配置日志记录设置：

```
var loggingConfig = AWSConfigs.LoggingConfig;
```

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

当您登录时 SystemDiagnostics，框架会在内部将输出打印到 System.Console。如果您想记录 HTTP 响应，请设置 LogResponses 标记。值可以是 “始终”、“从不” 或 OnError。

您也可以使用 LogMetrics 属性，记录 HTTP 请求的性能指标。日志格式可通过 LogMetricsFormat 属性来指定。有效值包括 JSON 或 standard。

设置区域端点

为所有服务客户端配置默认区域，如下所示：

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

这会为软件开发工具包中的所有服务客户端设置默认区域。您可以在创建服务客户端的实例时明确指定区域，以覆盖此设置，如下所示：

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

配置 HTTP 代理设置

如果您的网络位于代理之后，则您可以配置 HTTP 请求的代理设置，如下所示。

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

校正时钟偏差

此属性通过确定正确的服务器时间并重新发出时间正确的请求，判断软件开发工具包是否应校正客户端时钟偏差。

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

如果服务调用导致异常且软件开发工具包已判断出本地时间和服务器时间之间存在差异，则可以设置此字段。

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

要详细了解时钟偏差，请参阅 AWS 博客上的时钟偏差校正。

后续步骤

设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK 之后，您可以：

- 开始使用。阅读开始使用适用于 .NET 和 Xamarin 的 AWS Mobile SDK，以快速了解有关如何在适用于 .NET 和 Xamarin 的 AWS Mobile SDK 中使用和配置服务的说明。
- 了解服务主题。了解适用于 .NET 和 Xamarin 的 AWS Mobile SDK 中的每项服务及其工作原理。
- 运行演示。查看我们的示例 Xamarin 应用程序，以便了解常用案例。要运行应用程序示例，请设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK（详见上文），然后按各个示例中 README 文件内的说明操作。
- 学习 APIs. 查看 | sdk-xamarin-ref |_。
- 提问：在 AWS 移动软件开发工具包论坛上发布问题或在上提出问题 GitHub。

开始使用适用于 .NET 和 Xamarin 的 AWS Mobile SDK

适用于 .NET 和 Xamarin 的 AWS Mobile SDK 提供从 Xamarin 应用程序调用 AWS 服务所需的库、示例和文档。

开始使用以下服务之前，您必须先完成设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK 的说明中的所有步骤。

这些入门主题将指导您完成以下操作：

主题

- 使用 Amazon S3 存储和检索文件
- 使用 Cognito Sync 同步用户数据
- 使用 DynamoDB 存储和检索数据
- 通过 Amazon Mobile Analytics 跟踪应用程序使用率数据
- 使用 SNS 接收推送通知 (Xamarin iOS)
- 使用 SNS 接收推送通知 (Xamarin Android)

有关其他 AWS 移动设备的信息 SDKs，请参阅 AWS 移动软件开发工具包。

使用 Amazon S3 存储和检索文件

Amazon Simple Storage Service (Amazon S3) 为移动开发人员提供安全、持久、高度可扩展的对象存储。Amazon S3 易于使用，包含一个简单的 Web 服务界面，可从 Web 上的任何位置存储和检索任意数量的数据。

以下教程介绍了如何将 S3（一个用于将 S3 TransferUtility 与您的应用程序结合使用的高级实用程序）集成。有关从 Xamarin 应用程序使用 S3 的更多信息，请参阅 Amazon Simple Storage Service (S3)。

项目设置

先决条件

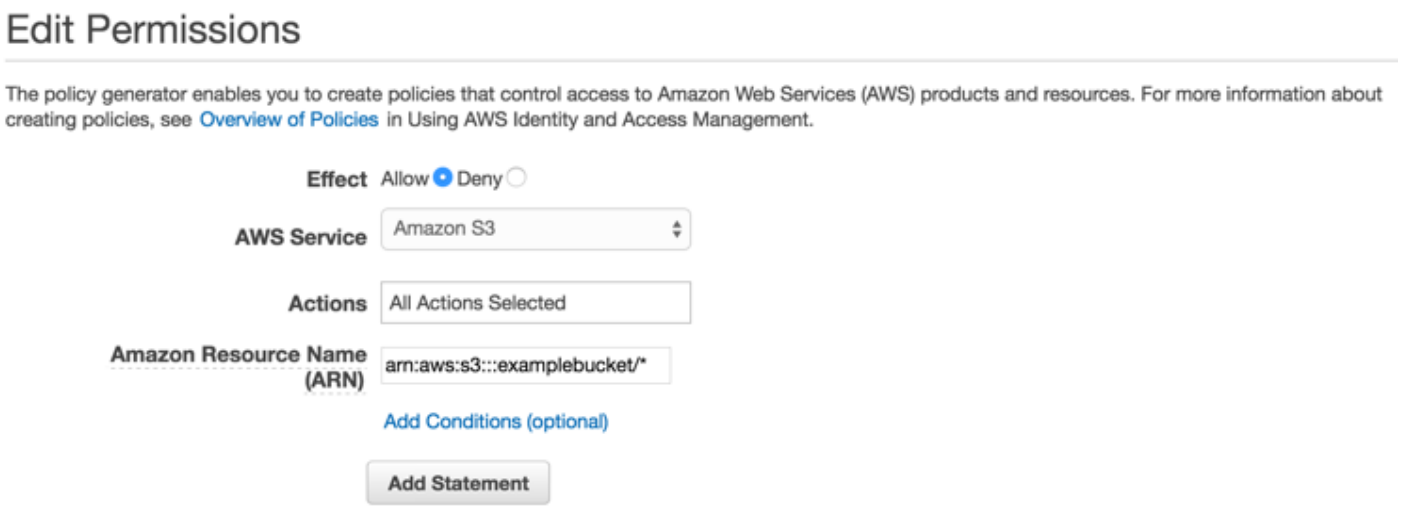
在开始本教程前，必须先完成有关设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK 的说明中的所有步骤。

本教程还假定您已创建 S3 存储桶。要创建 S3 存储桶，请访问 S3 AWS 管理控制台。

设置 S3 权限

默认 IAM 角色策略会授予您的应用程序访问 Amazon Mobile Analytics 和 Amazon Cognito Sync 的权限。为了让您的 Cognito 身份池能够访问 Amazon S3，您必须修改身份池的角色。

1. 转至 Identity and Access Management Console，然后单击左窗格中的 Roles。
2. 在搜索框中键入您的身份池名称。将列出两个角色：一个用于未经身份验证的用户，另一个用于经过身份验证的用户。
3. 单击用于未经过身份验证的用户的角色 (身份池名称后附加有“unauth”)。
4. 单击 Create Role Policy，选择 Policy Generator，然后单击 Select。
5. 在编辑权限页面上，输入下图所示的设置，用您自己的资源名称替换 Amazon 资源名称 (ARN)。S3 存储桶的 ARN 类似 `arn:aws:s3:::examplebucket/*`，由存储桶所在的区域和存储桶的名称构成。下面显示的设置将赋予您的身份池对指定存储桶执行所有操作的完全访问权限。



1. 单击 Add Statement 按钮，然后单击 Next Step。
2. 向导将向您显示生成的配置。单击应用策略。

有关授予访问 S3 的权限的更多信息，请参阅授予访问 Amazon S3 存储桶的权限。

将 Pac NuGet kage for S3 添加到您的项目中

按照设置适用于.NET 和 Xamarin 的 AWS 移动软件开发工具包中说明的第 4 步，将 NuGet S3 软件包添加到您的项目中。

(可选) 配置针对 S3 请求的签名版本

与 Amazon S3 的每一次交互都是经身份验证的或匿名的。AWS 使用签名版本 4 或签名版本 2 算法来对服务调用进行身份验证。

2014 年 1 月之后创建的所有新的 AWS 区域仅支持签名版本 4。但是，许多较旧的区域仍支持签名版本 4 和签名版本 2 请求。

如果您的存储桶位于不支持本页所列签名版本 2 请求的区域之一，则必须设置 AWSConfigs S3。UseSignatureVersion4 个属性变成 “true”，如下所示：

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

有关 AWS 签名版本的更多信息，请参阅对请求进行身份验证（AWS 签名版本 4）。

初始化 S3 TransferUtility 客户端

创建 S3 客户端，将其传递给您的 AWS 凭证对象，然后将 S3 客户端传递到 Transfer Utility，如下所示：

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

将文件上传到 Amazon S3

要将文件上传到 S3，请对 Transfer Utility 对象调用 `Upload`，并传递下列参数：

- `file` – 要上传的文件的名称 (字符串)
- `bucketName` – 用来存储文件的 S3 存储桶的字符串名称

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

上面的代码假设 “环境” 目录中有一个文件。 SpecialFolder。 ApplicationData。上传操作将自动对大文件使用 S3 的多分段上传功能以增强吞吐量。

从 Amazon S3 下载文件

要从 S3 下载文件，请对 Transfer Utility 对象调用 `Download`，并传递下列参数：

- `file` – 要下载的文件的名称 (字符串)
- `bucketName` – 要从中下载文件的 S3 存储桶的字符串名称
- `key` – 代表要下载的 S3 对象 (这里为一个文件) 名称的字符串

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

有关从 Xamarin 应用程序访问 Amazon S3 的更多信息，请参阅 Amazon Simple Storage Service (S3)。

使用 Cognito Sync 同步用户数据

Amazon Cognito Sync 让您可以轻松地在 AWS 云中保存移动用户数据（如应用程序首选项或游戏状态），无需编写任何后端代码或管理任何基础设施。您可以将数据本地保存在用户的设备上，这样，即使设备离线，应用程序也能工作。您也可以在用户的多个设备间同步数据，这样，无论他们使用什么设备，都可以得到一致的应用程序体验。

下面的教程将阐述如何将 Sync 与您的应用程序集成。

项目设置

先决条件

在开始本教程前，必须先完成有关设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK 的说明中的所有步骤。

授予访问您的 Cognito Sync 资源的权限

按照与您在设置期间所创建的未经身份验证和经过身份验证的角色相关联的默认策略，将授予您的应用程序访问 Cognito Sync 的权限。无需作进一步配置。

将 Cognito 同步的 Pack NuGet age 添加到你的项目中

按照设置适用于.NET 和 Xamarin 的 AWS 移动软件开发工具包中说明的第 4 步，将 Cog SyncManager NuGet n ito 软件包添加到您的项目中。

初始化 CognitoSyncManager

将您的已初始化的 Amazon Cognito 凭证提供商传递给 `CognitoSyncManager` 构造函数：

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

同步用户数据

同步未经身份验证的用户数据：

1. 创建一个数据集。
2. 将用户数据添加至该数据集。
3. 将数据集与云同步。

创建数据集

创建 `Dataset` 的实例。D openOrCreate ataset 方法用于创建新的数据集或打开设备上本地存储的数据集的现有实例：

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

将用户数据添加至数据集

用户数据以 key/value 成对的形式添加：

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

Cognito 数据集的功能与字典一样，包含可以通过键访问的值：

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

同步数据集

要同步数据集，请调用其同步方法：

```
dataset.SynchronizeAsync();

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

所有写入数据集的数据都存储在本地，直到数据集同步。本部分中的代码假定您使用的是未经身份验证的 Cognito 身份，所以，当用户数据与云同步时，将按设备存储数据。设备具有与其相关联的设备 ID。当用户数据同步到云时，用户数据将与该设备的 ID 关联。

有关 Cognito Sync 的更多信息，请参阅 Amazon Cognito Sync。

使用 DynamoDB 存储和检索数据

Amazon DynamoDB 是一项快速、高度可扩展、高度可用且经济实惠的非关系数据库服务。DynamoDB 消除了传统上对数据存储可扩展性的限制，同时保留了低延迟性和可预测的性能。

以下教程介绍了如何将 DynamoDB 对象持久化模型 (用于将对象存储在 DynamoDB 中) 与您的应用集成。

项目设置

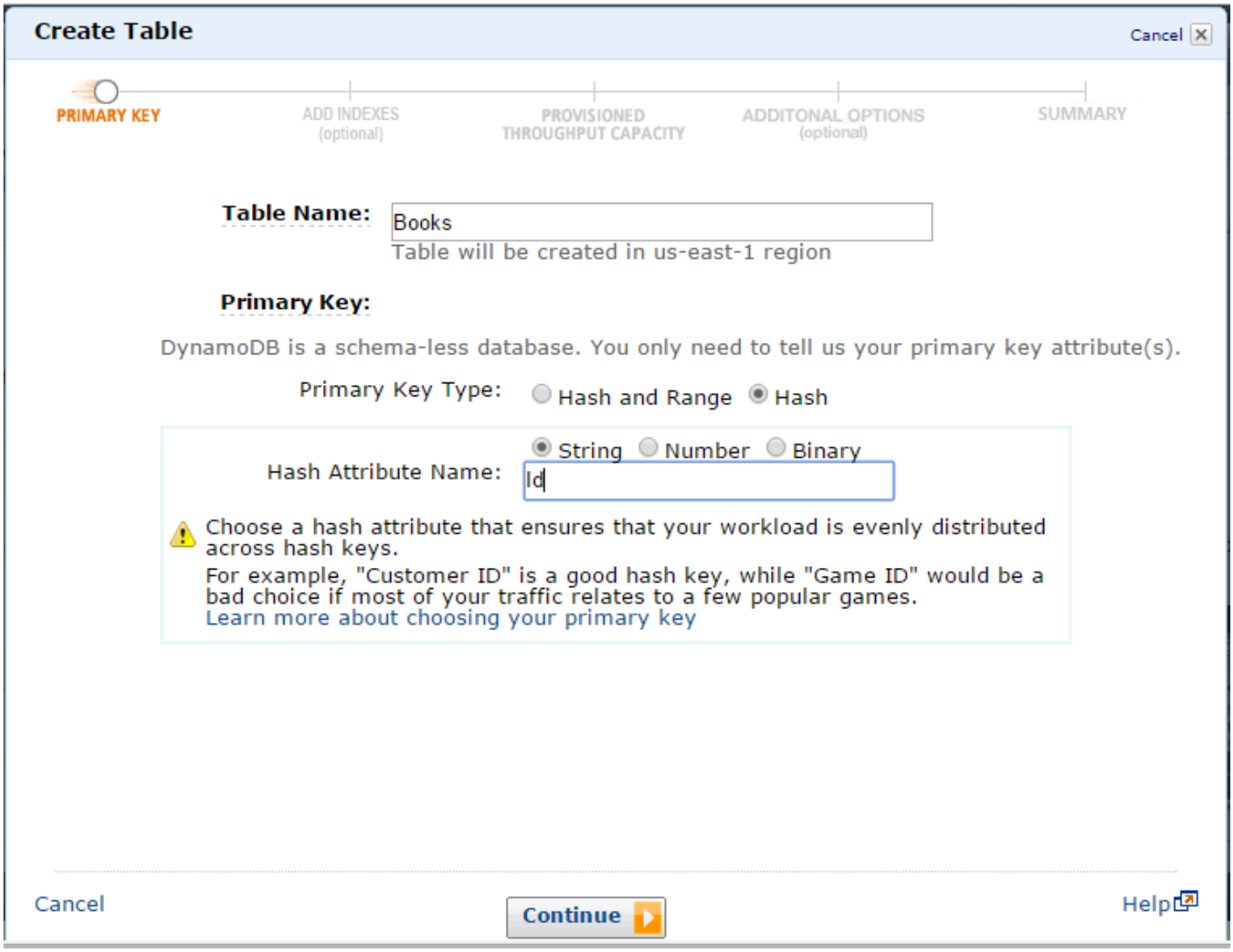
先决条件

在开始本教程前，必须先完成有关设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK 的说明中的所有步骤。

创建 DynamoDB 表

在对 DynamoDB 数据库执行数据读写操作之前，您必须先创建一个表。创建表时，您必须指定主键。主键由哈希特性 (attribute) 和可选的范围特性组成。有关如何使用主特性和范围特性的更多信息，请参阅处理表。

1. 转到 DynamoDB 控制台，然后单击 Create Table (创建表)。此时将打开“Create Table”向导。
2. 按如下所示指定表名称、主键类型（哈希）和哈希特性名称 (“Id”)，然后单击 Continue (继续)：

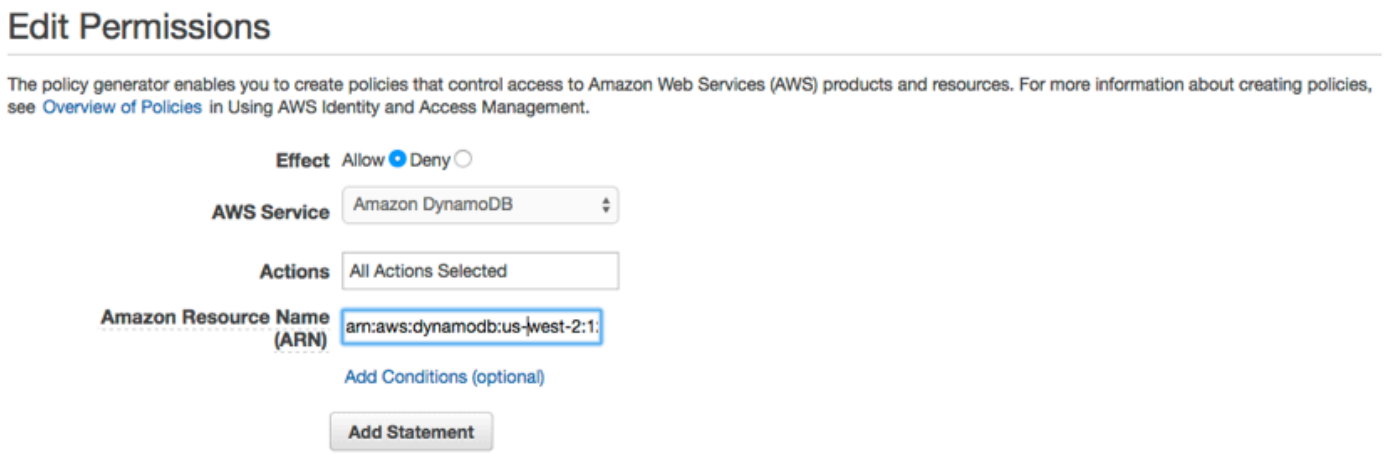


3. 将下一个屏幕中的编辑字段留空，然后单击 Continue (继续)。
4. 接受 Read Capacity Units (读取容量单元) 和 Write Capacity Units (写入容量单元) 的默认值，然后单击 Continue (继续)。
5. 在下一个屏幕上，在 Send notification to: (发送通知到:) 文本框中输入您的电子邮件地址，然后单击 Continue (继续)。此时将打开“Review”屏幕。
6. 单击创建。创建表可能需要几分钟。

设置 DynamoDB 的权限

为了使身份池能够访问 Amazon DynamoDB，您必须修改身份池的角色。

1. 导航到 Identity and Access Management 控制台，然后在左侧窗格中单击 Roles (角色)。搜索身份池名称 – 将列出两个角色，一个用于未经身份验证的用户，另一个用于经过身份验证的用户。
2. 单击用于未经过身份验证的用户（身份池名称后附加有“unauth”字样）的角色，然后单击 Create Role Policy (创建角色策略)。
3. 选择 Policy Generator (策略生成器)，然后单击 Select (选择)。
4. 在 Edit Permissions (编辑权限) 页上，按下图所示输入设置。DynamoDB 表的 Amazon 资源名称 (ARN) 类似于 `arn:aws:dynamodb:us-west-2:123456789012:table/Books`，由表所在的区域、所有者的 AWS 账号和表的名称构成，格式为 `table/Books`。有关指定的更多信息 ARNs，请参阅 DynamoDB 的亚马逊资源名称。



5. 单击 Add Statement (添加语句)，然后单击 Next Step (下一步)。向导将向您显示生成的配置。
6. 单击应用策略。

将 DynamoDB NuGet 软件包添加到您的项目中

按照设置适用于.NET 和 Xamarin 的 AWS 移动软件开发工具包中说明的第 4 步，将 Dyn amoDB NuGet 软件包添加到您的项目中。

初始化 AmazonDynamo DBClient

将初始化的 Amazon Cognito 凭证提供程序和您的区域传递给AmazonDynamoDB构造函数，然后将客户端传递给 Dynamo：DBContext

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

创建类

要向表中写入行，请定义一个类以保存行数据。该类还应包含用于保存行特性 (attribute) 数据的属性 (property)，并将映射到在控制台中创建的 DynamoDB 表。以下类声明说明了这种类：

```
[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; }
}
```

保存项目

要保存项目，请先创建一个对象：

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

然后保存。

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

要更新行，请修改 DDTableRow 类的实例，并按如上所示调用 `AWSDynamoObjectMapper.save()`。

检索项目

使用主键检索项目：

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

更新项目

如何更新项目:

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

删除项目

如何删除项目:

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

有关从 Xamarin 应用程序访问 DynamoDB 的更多信息，请参阅 Amazon DynamoDB。

通过 Amazon Mobile Analytics 跟踪应用程序使用率数据

借助 Amazon Mobile Analytics，您可以衡量应用程序的使用和收入情况。通过跟踪新老用户、应用程序收入、用户保留率及自定义应用程序内行为事件等关键趋势，您可以做出数据驱动型决策，以提高应用程序的吸引力和盈利能力。

下面的教程将阐述如何将 Mobile Analytics 与您的应用程序集成。

项目设置

先决条件

在开始本教程前，必须先完成有关设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK 的说明中的所有步骤。

在 Mobile Analytics 控制台中创建应用程序

转到 Amazon Mobile Analytics 控制台并创建应用程序。请记下 appId 值，因为您稍后会用到它。在 Mobile Analytics 控制台中创建应用程序时，您需要指定身份池 ID。有关创建身份池的说明，请参阅设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK。

要详细了解如何使用控制台，请参阅 Amazon Mobile Analytics 用户指南。

设置 Mobile Analytics 的权限

按照与您在设置期间所创建的角色相关联的默认策略，系统将授予您的应用程序访问 Mobile Analytics 的权限。无需作进一步配置。

将 Mobile Analytics Package 添加到 NuGet 您的项目中

按照设置适用于.NET 和 Xamarin 的 AWS 移动软件开发工具包中说明的第 4 步，将 Mobile Analytics 软件包添加到您的 NuGet 项目中。

配置 Mobile Analytics 设置

Mobile Analytics 会定义一些可在 awsconfig.xml 文件中配置的设置：

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

- AllowUseDataNetwork -一个布尔值，用于指定会话事件是否通过数据网络发送。
- DBWarning阈值-这是对数据库大小的限制，一旦达到该限制，就会生成警告日志。
- Max DBSize -这是 SQLIte 数据库的大小。如果数据库大小达到此上限值，任何后续事件都将被丢弃。
- MaxRequestSize -这是应在 HTTP 请求中传输到移动分析服务的请求的最大大小（以字节为单位）。
- SessionTimeout -这是应用程序进入后台之后以及可以终止会话的时间间隔。

上方显示的设置均为每个配置项目的默认值。

初始化 MobileAnalyticsManager

要初始化您的 MobileAnalyticsManager，请调用 GetOrCreateInstance 您`MobileAnalyticsManager`的 AWS 证书、您的区域、您的 Mobile Analytics 应用程序 ID 和可选的配置对象：

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

跟踪会话事件

Xamarin Android

覆盖活动的 `OnPause()` 和 `OnResume()` 方法以记录会话事件。

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

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

您应用程序中的每个活动都需要实施此操作。

Xamarin iOS

在你的 AppDelegate .cs 中：

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

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

有关 Mobile Analytics 的更多信息，请参阅 Amazon Mobile Analytics。

使用 SNS 接收推送通知 (Xamarin iOS)

本文档介绍了如何使用 Amazon Simple Notification Service (SNS) 和适用于 .NET 和 Xamarin 的 AWS Mobile SDK 将推送通知发送到 Xamarin iOS 应用程序。

项目设置

先决条件

在开始本教程前，必须先完成有关设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK 的说明中的所有步骤。

设置 SNS 权限

按照设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK 中第 2 步的说明操作，将下述策略附加到您应用程序的角色中。这样可为您的应用程序提供访问 SNS 的适当权限：

1. 转到 IAM 控制台，然后选择您要配置的 IAM 角色。
2. 单击 “附加策略”，选择 Amazon SNSFull 访问策略，然后单击 “附加策略”。

> **Warning**
>
> 不建议在生产环境中SNSFull使用 Amazon Access。我们在此处使用它是为了让您快速启动并运行。有关为 IAM 角色指定权限的更多信息，请参阅 IAM 角色权限概述。

获得 Apple iOS 开发人员计划成员资格

您需要在物理设备上运行您的应用程序以接收推送通知。要在设备上运行您的应用程序，您必须拥有 Apple iOS 开发人员计划成员资格。一旦您拥有了成员资格，就可以使用 Xcode 生成签名身份。有关更多信息，请参阅 Apple 的 App Distribution Quick Start 文档。

创建 iOS 证书

首先，您需要创建一个 iOS 证书。然后，您需要创建为推送通知而配置的预置配置文件。为此，请执行以下操作：

1. 转至 Apple Developer Member Center，单击 Certificates, Identifiers & Profiles。

2. 单击 iOS Apps 下的 Identifiers，单击 Web 页面右上角的加号按钮以添加一个新的 iOS 应用程序 ID，然后输入应用程序 ID 描述。
3. 向下滚动到 Add ID Suffix 部分，选择 Explicit App ID，然后输入您的服务包标识符。
4. 向下滚动到 App Services 部分，并选择 Push Notifications。
5. 单击继续。
6. 单击 Submit (提交)。
7. 单击完成。
8. 选择您刚刚创建的应用程序 ID，然后单击 Edit。
9. 向下滚动到 Push Notifications 部分。单击 Development SSL Certificate 下的 Create Certificate。
10. 按照说明创建证书签名请求 (CSR)、上传请求、下载将用于与 Apple Notification Service (APNS) 通信的 SSL 证书。
11. 返回到证书、身份和配置文件页面。单击 Provisioning Profiles 下的 All。
12. 单击右上角的 + 按钮以添加新的预置配置文件。
13. 选择 iOS App Development，然后单击 Continue。
14. 选择您的应用程序 ID，然后单击 Continue。
15. 选择您的开发人员证书，然后单击 Continue。
16. 选择您的设备，然后单击 Continue。
17. 输入配置文件名称，然后单击 Generate。
18. 下载预置文件后双击以安装预置配置文件。

有关预置为推送通知而配置的配置文件的更多信息，请参阅 Apple 的配置推送通知文档。

使用证书在 SNS 控制台中创建平台 ARN

1. 运行 KeyChain 访问应用程序，选择屏幕左下角的我的证书，然后右键单击您为连接到 APNS 而生成的 SSL 证书并选择导出。系统将提示您指定文件的名称和密码以保护证书。证书将保存在 P12 文件中。
2. 转至 SNS Console，单击屏幕左侧的 Applications (应用程序)。
3. 单击 Create platform application，以创建新的 SNS 平台应用程序。
4. 输入 Application Name。
5. 对于 Push notification platform，选择 Apple Development。
6. 单击 Choose File，选择导出 SSL 证书时创建的 P12 文件。

7. 输入在导出 SSL 证书时指定的密码，然后单击 Load Credentials From File。
8. 单击 Create platform application。
9. 选择您刚创建的平台应用程序，然后复制应用程序 ARN。在后续步骤中，您将需要此信息。

将 Pack NuGet age for SNS 添加到您的项目中

按照设置适用于.NET 和 Xamarin 的 AWS 移动软件开发工具包中说明的第 4 步，将亚马逊简单通知NuGet服务包添加到您的项目中。

创建 SNS 客户端

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

为您的应用程序注册远程通知

要注册应用程序，请调用 RegisterForRemoteNotifications 您的 UIApplication对象，如下所示。将以下代码放在 AppDelegate .cs 中，在出现以下提示的地方插入您的平台应用程序 ARN：

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

将消息从 SNS 控制台发送到端点

1. 转到 SNS 控制台 >“Applications (应用程序)”。
2. 依次选择您的平台应用程序和端点，然后单击 Publish to endpoint。
3. 在文本框中键入文本消息，然后单击 Publish message 发布消息。

使用 SNS 接收推送通知 (Xamarin Android)

本教程介绍了如何使用 Amazon Simple Notification Service (SNS) 和适用于 .NET 和 Xamarin 的 AWS Mobile SDK 将推送通知发送到 Xamarin Android 应用程序。

项目设置

先决条件

在开始本教程前，必须先完成有关设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK 的说明中的所有步骤。

设置 SNS 权限

按照设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK 中第 2 步的说明操作，将下述策略附加到您应用程序的角色中。这样可为您的应用程序提供访问 SNS 的适当权限：

1. 转到 IAM 控制台，然后选择您要配置的 IAM 角色。
2. 单击 “附加策略”，选择 Amazon SNSFull 访问策略，然后单击 “附加策略”。

> **Warning**
>
> 不建议在生产环境中SNSFull使用 Amazon Access。我们在此处使用它是为了让您快速启动并运行。有关为 IAM 角色指定权限的更多信息，请参阅 IAM 角色权限概述。

在 Google Cloud 上启用推送通知

首先，添加一个新的 Google API 项目：

1. 转到 Google Developers Console。
2. 单击 Create Project。
3. 在 New Project 框中，输入项目名称，记下项目 ID (稍后您会用到它)，然后单击 Create。

接下来，为您的项目启用 Google Cloud Messaging (GCM) 服务：

1. 在 Google Developers Console 中，系统应该已经选择了您的新项目。如果没有，请在页面顶部的下拉列表中选择。
2. 从页面左侧的侧栏中选择 APIs & auth。
3. 在搜索框中，键入“Google Cloud Messaging for Android”，然后单击Google Cloud Messaging for Android链接。
4. 单击 Enable API。

最后，获取 API 密钥：

1. 在 Google 开发者控制台中，选择并验证 APIs > 凭据。
2. 在 Public API access 下，单击 Create new key。
3. 在 Create a new key 对话框中，单击 Server key。
4. 在出现的对话框中，单击 Create，然后复制显示的 API 密钥。稍后，您将使用此 API 密钥来执行身份验证。

使用项目 ID 在 SNS 控制台中创建平台 ARN

1. 转到 SNS 控制台。
2. 单击屏幕左侧的 Applications。
3. 单击 Create platform application，以创建新的 SNS 平台应用程序。
4. 输入 Application Name。
5. 对于 Push notification platform，选择 Google Cloud Messaging (GCM)。
6. 将 API 密钥粘贴到标记为 API key 的文本框中。
7. 单击 Create platform application。

8. 选择您刚创建的平台应用程序，然后复制应用程序 ARN。

将 Pack NuGet age for SNS 添加到您的项目中

按照设置适用于.NET 和 Xamarin 的 AWS 移动软件开发工具包中说明的第 4 步，将亚马逊简单通知NuGet服务包添加到您的项目中。

创建 SNS 客户端

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

为您的应用程序注册远程通知

要在 Android 上注册远程通知，您需要创建一个可以接收 Google Cloud 消息 BroadcastReceiver 的。请按照下方的提示，更改程序包名称：

```
[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);
      }
}
```

以下是接收来自的推送通知 BroadcastReceiver 并在设备的通知栏上显示通知的服务：

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

将消息从 SNS 控制台发送到端点

1. 转到 SNS 控制台 >“Applications (应用程序)”。
2. 依次选择您的平台应用程序和端点，然后单击 Publish to endpoint。
3. 在文本框中键入文本消息，然后单击 Publish message 发布消息。

Amazon Cognito Identity

什么是 Amazon Cognito Identity？

借助 Amazon Cognito Identity，您能够为用户创建唯一的身份，并通过身份提供商对其进行身份验证。有了身份，您便可以获取具有有限权限的临时 AWS 凭证，以使用 Amazon Cognito Sync 同步数据，或者直接访问其他 AWS 服务。Amazon Cognito Identity 支持公共身份提供商 (Amazon、Facebook 和 Google)，以及未经身份验证的身份。此外，它还支持已经过开发人员验证的身份，这让您能够通过自己的后端身份验证流程注册用户并对用户进行身份验证。

有关 Cognito Identity 的更多信息，请参阅 Amazon Cognito 开发人员指南。

有关 Cognito 身份验证区域可用性的信息，请参阅 AWS 服务区域可用性。

使用公共提供商对用户进行身份验证

使用 Amazon Cognito Identity，您可以为用户创建唯一的身份并对其进行身份验证，以实现对 AWS 资源（如 Amazon S3 或 Amazon DynamoDB）的安全访问。Amazon Cognito Identity 支持公共身份提供商 (Amazon、Facebook、Twitter/Digits、Google 或兼容 OpenID Connect 的任何提供商)，以及未经身份验证的身份。

有关使用公共身份提供商 (如 Amazon、Facebook、Twitter/Digits 或 Google) 对用户进行身份验证的信息，请参阅《Amazon Cognito 开发人员指南》中的外部提供商。

使用已经过开发人员验证的身份

除了通过 Facebook、Google 和 Amazon 进行 Web 联合身份验证之外，Amazon Cognito 还支持已经过开发人员验证的身份。借助已经过开发人员验证的身份，您可以注册用户并通过自己的现有身份验证流程对用户进行身份验证，同时仍然使用 Amazon Cognito Sync 同步用户数据和访问 AWS 资源。使用已经过开发人员验证的身份涉及最终用户设备、身份验证后端和 Amazon Cognito 之间的交互。

有关已经过开发人员验证的身份的更多信息，请参阅《Amazon Cognito 开发人员指南》中的已经过开发人员验证的身份。

Amazon Cognito Sync

什么是 Amazon Cognito Sync？

Cognito Sync 是一种 AWS 服务和客户端库，可允许跨设备同步用户数据（例如，游戏得分、用户首选项、游戏状态）。您可以使用 Cognito Sync API 跨设备同步用户数据。要在应用程序中使用 Cognito Sync，您必须在项目中添加 |。

有关如何在应用程序中集成 Amazon Cognito Sync 的说明，请参阅 Amazon Cognito Sync 开发人员指南。

Amazon Mobile Analytics

Amazon Mobile Analytics 是一个用于大规模收集、可视化、理解和提取应用程序使用率数据的服务。Mobile Analytics 可轻松捕获标准设备数据和自定义事件，并代表您自动计算报告。除了下面列出的聚合报告，您还可以设置自动将数据导出到 Redshift 和 S3 以作进一步分析。

使用 Amazon Mobile Analytics，您可以跟踪客户行为、聚合指标、生成数据可视化以及确定有意义的模式。

重要概念

报告类型

Mobile Analytics 在 Mobile Analytics 控制台中提供以下现成的报告：

- 每日活跃用户 (DAU)，每月活跃用户 (MAU) 和新用户
- 粘性系数 (DAU 除以 MAU)
- 会话计数和单个每日活跃用户的平均会话数
- 单个每日活跃用户的平均收入 (ARPDAU) 和单个每日付费活跃用户的平均收入 (ARPPDAU)
- 第 1、第 3 和第 7 天的保留率和第 1、第 2 和第 3 周的保留率
- 自定义事件

这些报告在控制台中通过六个报告选项卡提供：

- 概述 — 在 simple-to-review仪表板中跟踪九份预先选择的报告，以快速了解参与度：MAU、DAU、新用户、每日会话、粘性因子、1 天留存率、ARPDAU、每日付费用户、ARPPDAU。
- 活跃用户 – 跟踪每天和每月有多少用户使用您的应用程序，监视粘性系数，以衡量参与度、吸引力和货币化指标。
- 会话 – 跟踪在给定的某一天您的应用程序被使用的频率以及一天内每个用户打开您应用程序的频率。
- 保留率 – 跟踪每天和每周客户回来使用您的应用程序的比率。
- 收入 – 跟踪应用程序内收入趋势以确定货币化进展有待改进的方面。
- 自定义事件 – 跟踪特定于您的应用程序的自定义用户操作。

要了解更多有关 Mobile Analytics 报告和在 Mobile Analytics 控制台中操作的信息，请参阅《Mobile Analytics 开发人员指南》中的 Mobile Analytics 控制台报告概览。

项目设置

先决条件

要在您的应用程序中使用 Mobile Analytics，需要将软件开发工具包 (SDK) 添加到您的项目中。为此，请按照设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK 中的说明操作。

配置 Mobile Analytics 设置

Mobile Analytics 会定义一些可在 awsconfig.xml 文件中配置的设置：

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

- SessionTimeout-如果应用程序在后台停留的时间长于该时间， SessionTimeout 则 Mobile Analytics 客户端将终止当前会话，并在应用程序返回前台时创建新的会话。建议使用从 5 到 10 的值。默认值是 5。
- Max DBSize-用于本地存储事件的数据库的最大大小（以字节为单位）。如果数据库大小超过此值，后续事件将被忽略。建议使用从 1 MB 到 10 MB 的值。默认值是 5242880 (5 MB)。
- DBWarningThreashold-警告阈值。有效值范围为 0 到 1。如果值超出阈值，会生成警告日志。默认值为 0.9。
- MaxRequestSize-向 Mobile Analytics 服务发出的 HTTP 请求的最大大小。该值以字节为单位，范围为 1-512 KB。默认值为 102400 (100 KB)。不要使用大于 512 KB 的值，否则可能导致服务拒绝 HTTP 请求。
- AllowUseDataNetwork-表示是否允许通过蜂窝数据网络进行服务呼叫的值。使用此选项要小心，因为有可能增加客户的数据使用量。

上方显示的设置均为每个配置项目的默认值。

将 Mobile Analytics 与您的应用程序集成

以下各节将阐述如何将 Mobile Analytics 与您的应用程序集成。

在 Mobile Analytics 控制台中创建应用程序

转到 Amazon Mobile Analytics 控制台并创建应用程序。请记下 appId 值，因为您稍后会用到它。在 Mobile Analytics 控制台中创建应用程序时，您需要指定身份池 ID。有关创建身份池的说明，请参阅设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK。

要了解更多有关在 Mobile Analytics 控制台中操作的信息，请参阅《Mobile Analytics 开发人员指南》中的 Mobile Analytics 控制台报告概览。

创建 MobileAnalyticsManager 客户端

要初始化您的 MobileAnalyticsManager，请调用 GetOrCreateInstance 您MobileAnalyticsManager的 AWS 证书、您的区域、您的 Mobile Analytics 应用程序 ID 和可选的配置对象：

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

APP_ID 是在执行应用程序创建向导期间生成的。这两个值都必须与 Mobile Analytics 控制台中相应的值匹配。APP_ID 用于在 Mobile Analytics 控制台中将您的数据分组。在 Mobile Analytics 控制台创建应用程序后，要查找应用程序 ID，请导航到 Mobile Analytics 控制台，然后单击屏幕右上角的齿轮图标。这将显示应用程序管理页面，其中列出了所有注册的应用程序及其应用程序 IDs。

记录货币化事件

适用于 .NET 和 Xamarin 的 AWS Mobile SDK 可提供 MonetizationEvent 类，该类可用来生成货币化事件，以跟踪在移动应用程序内进行的购买。以下代码段演示如何创建货币化事件：

```
// 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);
```

记录自定义事件

Mobile Analytics 允许您定义自定义事件。自定义事件完全由您自己定义；它们可帮助您跟踪特定于您的应用程序或游戏的用户操作。有关自定义事件的更多信息，请参阅自定义事件。

在本示例中，假设我们的应用程序是一个游戏，并且我们希望在用户完成一关时记录一个事件。通过创建新`AmazonMobileAnalyticsEvent`实例来创建 “LevelComplete” 事件：

```
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);
```

记录会话

Xamarin iOS

当应用程序失去焦点时，您可以暂停会话。对于 iOS 应用程序，在 AppDelegate .cs 文件中，覆盖`DidEnterBackground`和`WillEnterForeground`并调

用MobileAnalyticsManager.PauseSesion和MobileAnalyticsManager.ResumeSession，如以下代码段所示：

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

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

Xamarin Android

对于 Android 应用程序，请MobileAnalyticsManager.ResumeSession在 OnPause () 方法和 OnResume () 方法中调MobileAnalyticsManager.PauseSesion用，如以下代码片段所示：

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

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

默认情况下，如果用户切换焦点，应用程序失焦少于 5 秒，然后再切换回应用程序，则会话将恢复。如果用户切换焦点，使应用程序失焦 5 秒或更长时间，将创建新的会话。通过将“SESSION_DELTA”属性设置为创建新会话前等待的秒数，可在 aws_mobile_analytics.json 配置文件中配置此设置。

Amazon Simple Storage Service (S3)

什么是 S3？

Amazon Simple Storage Service (Amazon S3) 为开发人员提供安全、持久、高度可扩展的对象存储。Amazon S3 易于使用，包含一个简单的 Web 服务界面，可从 Web 上的任何位置存储和检索任意数量的数据。使用 Amazon S3，您只需为实际使用的存储付费。没有最低费用，也没有设置成本。

Amazon S3 为各种使用情形提供经济高效的对象存储，包括云应用程序、内容分发、备份和归档、灾难恢复和大数据分析。

有关 AWS S3 区域可用性的信息，请参阅 AWS 服务区域可用性。

重要概念

存储桶

存储于 Amazon S3 中的每个数据元都存储在存储段中。如同通过目录对文件系统中的文件进行分组一样，您也可以使用存储桶对相关对象进行分组。存储桶具有诸如访问权限和版本控制状态之类的属性，并且您可以指定希望它们所在的区域。

要了解更多有关 S3 存储桶的信息，请参阅《S3 开发人员指南》中的使用存储桶。

对象

对象是您在 Amazon S3 中存储的数据。每个对象都位于您在特定 AWS 区域中创建的存储桶中。

在某一区域存储的对象将一直留在该区域，除非您特意将其传输到另一区域。例如，在欧洲地区（爱尔兰）区域存储的对象将一直保留在欧洲。在某个 Amazon S3 区域中存储的对象会以物理形式保留在该区域。Amazon S3 不会保留这些对象的副本或将其移动到其他任何区域。但是，只要您具有必要的权限，就可以从任何地方访问这些对象。

对象可以是任何类型的文件：图像、备份数据和电影等。一个对象的大小可以高达 5 TB。一个存储桶中可以有无限量的对象。

您必须先拥有存储段写入权限，才能将数据元上传到 Amazon S3 中。有关设置存储桶权限的更多信息，请参阅《S3 开发人员指南》中的编辑存储桶权限。

要了解更多有关 S3 对象的信息，请参阅《S3 开发人员指南》中的使用对象。

对象元数据

Amazon S3 中的每个对象都有一组代表其元数据的键值对。元数据有两种类型：

- 系统元数据 – 有时通过 Amazon S3 处理，例如 Content-Type 和 Content-Length。
- 用户元数据 – 从不通过 Amazon S3 处理。用户元数据会与对象存储在一起，并会随其返回。用户元数据最大可以为 2 KB，键及其值均必须符合 US-ASCII 标准。

要了解更多有关 S3 对象元数据的信息，请参阅编辑对象元数据。

项目设置

先决条件

要在您的应用程序中使用 Amazon S3，需要将 SDK 添加到您的项目中。为此，请按照设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK 中的说明操作。

创建 S3 存储桶

Amazon S3 可将您的应用程序资源存储到 Amazon S3 存储桶（位于某个特定区域的云存储容器）。每个 Amazon S3 存储桶必须具有一个全局唯一名称。您可以使用 Amazon S3 控制台创建存储桶。

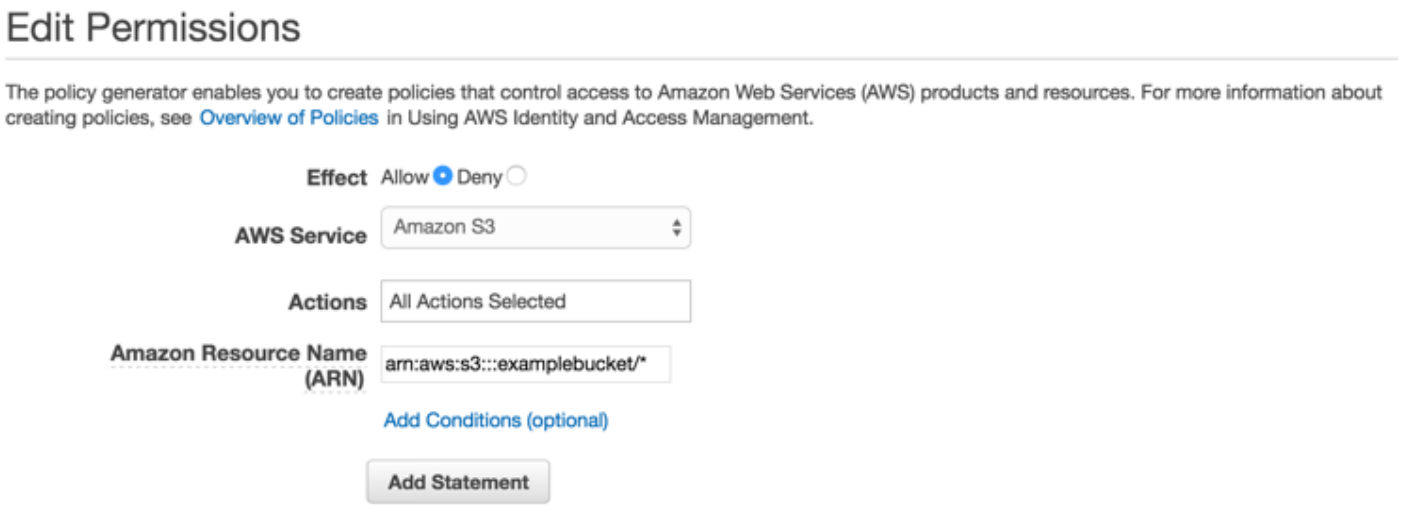
1. 登录 Amazon S3 控制台，然后单击 Create Bucket。
2. 输入存储桶的名称，选择一个区域，然后单击 Create。

设置 S3 权限

默认 IAM 角色策略会授予您的应用程序访问 Amazon Mobile Analytics 和 Amazon Cognito Sync 的权限。为了让您的 Cognito 身份池能够访问 Amazon S3，您必须修改身份池的角色。

1. 转至 Identity and Access Management Console，然后单击左窗格中的 Roles。
2. 在搜索框中键入您的身份池名称。将列出两个角色：一个用于未经身份验证的用户，另一个用于经过身份验证的用户。
3. 单击用于未经过身份验证的用户的角色 (身份池名称后附加有“unauth”)。

4. 单击 Create Role Policy，选择 Policy Generator，然后单击 Select。
5. 在编辑权限页面上，输入下图所示的设置，用您自己的资源名称替换 Amazon 资源名称 (ARN)。S3 存储桶的 ARN 类似 `arn:aws:s3:::examplebucket/*`，由存储桶所在的区域和存储桶的名称构成。下面显示的设置将赋予您的身份池对指定存储桶执行所有操作的完全访问权限。



1. 单击 Add Statement 按钮，然后单击 Next Step。
2. 向导将向您显示生成的配置。单击应用策略。

有关授予访问 S3 的权限的更多信息，请参阅授予访问 Amazon S3 存储桶的权限。

(可选) 配置针对 S3 请求的签名版本

与 Amazon S3 的每一次交互都是经身份验证的或匿名的。AWS 使用签名版本 4 或签名版本 2 算法来对服务调用进行身份验证。

2014 年 1 月之后创建的所有新的 AWS 区域仅支持签名版本 4。但是，许多较旧的区域仍支持签名版本 4 和签名版本 2 请求。

如果您的存储桶位于不支持本页所列签名版本 2 请求的区域之一，则必须设置 AWSConfigs S3。UseSignatureVersion4 个属性变成 “true”，如下所示：

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

有关 AWS 签名版本的更多信息，请参阅对请求进行身份验证（AWS 签名版本 4）。

将 S3 与您的应用程序集成

您可以通过两种方式在您的 Xamarin 中与 S3 进行交互。以下主题更深入地探讨了这两种方法：

使用 S3 Transfer Utility

利用 S3 Transfer Utility，可以更轻松地从 Xamarin 应用程序向 S3 上传文件以及从中下载文件。

初始化 TransferUtility

创建 S3 客户端，将其传递给您的 AWS 凭证对象，然后将 S3 客户端传递到 Transfer Utility，如下所示：

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

（可选）配置 TransferUtility

有三个可选属性可以配置：

- ConcurrentServiceRequests-确定文件将使用多少活动线程或并发异步 Web 请求 upload/download 的数量。默认值是 10。
- MinSizeBeforePartUpload-获取或设置上传段的最小分段大小（以字节为单位）。默认值为 16 MB。降低最小分段大小会导致多分段上传被拆分为大量更小的分段。此值设置得过低会对传输速度产生不利影响，导致每个分段出现额外的延迟和网络通信。
- NumberOfUploadThreads-获取或设置正在执行的线程数。该属性决定将用于上传文件的活动线程的数量。默认值为 10 个线程。

要配置 S3 TransferUtility 客户端，请创建一个配置对象，设置您的属性，然后将该对象传递给您的 TransferUtility 构造函数，如下所示：

```
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);
```

下载文件

要从 S3 下载文件，请对 Transfer Utility 对象调用 Download，并传递下列参数：

- file – 要下载的文件的名称 (字符串)
- bucketName – 要从中下载文件的 S3 存储桶的字符串名称
- key – 代表要下载的 S3 对象 (这里为一个文件) 名称的字符串

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

上传文件

要将文件上传到 S3，请对 Transfer Utility 对象调用 Upload，并传递下列参数：

- file – 要上传的文件的名称 (字符串)
- bucketName – 用来存储文件的 S3 存储桶的字符串名称

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

上面的代码假设 “环境” 目录中有一个文件。 SpecialFolder。 ApplicationData。上传操作将自动对大文件使用 S3 的多分段上传功能以增强吞吐量。

使用服务级别 S3 APIs

除了使用 S3 之外 TransferUtility，您还可以使用低级 S3 与 S3 进行交互。 APIs

初始化 Amazon S3 客户端

要使用 Amazon S3，我们首先需要创建一个 Amazons3Client 实例，该实例引用您之前创建的 Cognito AWSCredentials 实例和您所在的地区：

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

下载文件

从 S3 下载文件：

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

上传文件

将文件上传到 S3:

```
// 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);
```

删除项目

删除 S3 中的项目：

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

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

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

删除多个项目

使用单个 HTTP 请求从存储桶中删除多个对象：

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

您最多可以指定 1000 个键。

列出存储桶

返回已经过身份验证的请求发件人所拥有的所有存储桶列表：

```
// 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);
```

```
}
```

列出对象

您可以返回 S3 存储桶中存储的部分或全部（最多 1000 个）对象。为此，您必须拥有对存储桶的读取权限。

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

获取存储桶所在的区域

获取存储桶所在的区域：

```
// 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);
```

获取存储桶的策略

获取存储桶的策略：

```
// 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

什么是 Amazon DynamoDB？

Amazon DynamoDB 是一个快速、高度可扩展的非关系型数据库服务。DynamoDB 消除了传统上对数据存储可扩展性的限制，同时保留了低延迟性和可预测的性能。

重要概念

DynamoDB 数据模型概念包括表、项目和属性。

表

在 Amazon DynamoDB 中，数据库是表的集合。表是项目的集合，而每个项目是属性的集合。

在关系数据库中，表具有预定义的架构，例如表名、主键、其列名称列表及其数据类型。存储在表中的所有记录必须具有相同的列集。相反，DynamoDB 只需要表具有一个主键，而不需要您预先定义所有属性名称和数据类型。

要了解使用表的更多信息，请参阅使用 DynamoDB 表。

项目和属性

DynamoDB 表中的各个项目可以有任意数量的属性，但项目大小的上限为 400 KB。项目大小是其属性名称和值的长度总和（二进制和 UTF-8 长度）。

项目中的每个属性都是名称值对。属性可以是单值或多值。例如，书籍项目可以具有标题和作者属性。每本书都有一个书名，但可以有多个作者。多值属性是一个集合；不允许使用重复的值。

例如，假设要在 DynamoDB 中存储产品目录。您可以创建一个以 Id 属性作为其主键的表。ProductCatalog主键唯一地标识每一项，因此，表中不会有两个产品具有相同 ID。

要了解使用项目的更多信息，请参阅使用 DynamoDB 项目。

数据类型

Amazon DynamoDB 支持以下数据类型：

- 标量类型 – 数字、字符串、二进制、布尔和 Null。
- 多值类型 – 字符串集、数字集和二进制集。

- 文档类型 – 列表和映射。

有关标量数据类型、多值数据类型和文档数据类型的更多信息，请参阅 DynamoDB 数据类型。

主键

创建表时，除表名称外，您还必须指定表的主键。主键唯一标识表中的每个项目，因此，任意两个项目的主键都不相同。DynamoDB 支持以下两类主键：

- 哈希主键 – 这类主键由一个属性（哈希属性）构成。DynamoDB 基于此主键属性构建无序的哈希索引。表中的每个项目由其哈希键值进行唯一标识。
- 哈希和范围主键 – 这类主键由两个属性构成。第一个属性是哈希属性，第二个属性是范围属性。DynamoDB 基于哈希主键属性构建无序的哈希索引，基于范围主键属性构建有序的范围索引。表中的每一项由其哈希和范围键值的组合进行唯一标识。两个项目可能具有相同的哈希键值，但是这两个项目必须具有不同的范围键值。

二级索引

当您创建具有哈希和范围键的表时，可以选择在该表上定义一个或多个二级索引。利用二级索引，除了可对主键进行查询外，还可使用替代键查询表中的数据。

DynamoDB 支持两种类型的二级索引：本地二级索引和全局二级索引。

- 本地二级索引：具有与表相同的哈希键、不同范围键的索引。
- 全局二级索引：具有与表不同的哈希键和范围键的索引。

您最多可以为每个表定义 5 个全局二级索引和 5 个本地二级索引。有关更多信息，请参阅《DynamoDB 开发人员指南》中的在 DynamoDB 中使用二级索引改善数据访问。

查询和扫描

除了使用主键访问项目外，Amazon DynamoDB 还提供了 APIs 两个用于搜索数据的功能：查询和扫描。建议您阅读《DynamoDB 开发人员指南》中的查询和扫描指南，以熟悉一些最佳实践。

查询

查询操作仅使用主键属性值查找表或二级索引中的项目。您必须提供哈希键属性名称和要搜索的确切值。您可以选择提供范围键属性名称和值，并使用比较运算符来优化搜索结果。

有关示例查询，请参阅：

- 使用文档模型
- 使用对象持久化模型
- 使用 DynamoDB 服务级别 APIs

有关查询的更多信息，请参阅《DynamoDB 开发人员指南》中的查询。

Scan

扫描操作读取表或二级索引中的每个项目。默认情况下，扫描操作返回表或索引中每个项目的全部数据属性。您可以使用 ProjectionExpression 参数，以便 Scan 仅返回部分属性，而不是全部属性。

有关示例扫描，请参阅：

- 使用文档模型
- 使用对象持久化模型
- 使用 DynamoDB 服务级别 APIs

有关扫描的更多信息，请参阅《DynamoDB 开发人员指南》中的扫描。

项目设置

先决条件

要在您的应用程序中使用 DynamoDB，需要将开发工具包添加到您的项目中。为此，请按照设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK 中的说明操作。

创建 DynamoDB 表

要创建表，请转至 DynamoDB 控制台并执行以下步骤：

1. 单击创建表。
2. 输入表的名称。
3. 选择 Hash 作为主键类型。
4. 选择一个类型并输入哈希属性名称的值。单击继续。

5. 在添加索引页面上，如果您计划使用全局二级索引，请将索引类型设置为“全局二级索引”，然后在索引哈希键下，输入二级索引的值。这将使您能够同时使用主索引和二级索引进行查询和扫描。单击将索引添加到表，然后单击继续。要跳过使用全局二级索引，请单击 Continue。
6. 将读取和写入容量设置为所需水平。有关配置容量的更多信息，请参阅 Amazon DynamoDB 中预配置的吞吐量。单击继续。
7. 在下一个屏幕中，输入通知电子邮件以创建吞吐量警报（如果需要）。单击继续。
8. 在摘要页面上，单击创建。DynamoDB 将创建您的数据库。

设置 DynamoDB 的权限

要在应用程序中使用 DynamoDB，必须设置正确的权限。以下 IAM 策略允许用户删除、获取、放置、查询、扫描和更新特定的用 ARN 标识的 DynamoDB 表中的项目：

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

您可以在 IAM 控制台中修改策略。应根据您的应用程序的需要添加或删除允许的操作。

要了解有关 IAM 策略的更多信息，请参阅使用 IAM。

要了解更多有关 DynamoDB 特定策略的信息，请参阅《DynamoDB 开发人员指南》中的使用 IAM 控制对 DynamoDB 资源的访问。

将 DynamoDB 与您的应用程序集成

适用于 .NET 和 Xamarin 的 AWS Mobile SDK 可为您使用 DynamoDB 提供高级库。您也可以直接针对低级 DynamoDB API 发出请求，但在大多数情况下，建议您使用高级库。AmazonDynamoDBClient 是高级库中特别有用的部分。使用此类，您可以执行创建、读取、更新和删除 (CRUD) 操作和执行查询。

适用于.NET 和 Xamarin 的 AWS 移动软件开发工具包允许您 APIs 使用适用于.NET 的 AWS 开发工具包拨打电话，以便与 DynamoDB 配合使用。所有这些 APIs 都可以在 AWSSDK .dll 中找到。有关下载适用于 .NET 的 AWS SDK 的信息，请参阅适用于 .NET 的 AWS SDK。

您可以通过三种方式在您的 Xamarin 应用程序中与 DynamoDB 交互：

- 文档模型：该 API 提供有关低级 DyanmoDB API 的封装类，以进一步简化编程任务。表和文档是主要的封装类。您可以使用文档模式执行数据操作，例如创建、检索、更新和删除项目。该 API 已在 Amazon.dynamoDB 中提供。 DocumentModel 命名空间。
- 对象持久化模型：对象持久化 API 允许您将客户端类映射到 DynamoDB 表。然后，每个对象实例映射到对应表中的项目。此 API 中的 Dynamo DBContext 类为您提供了将客户端对象保存到表中、将项目作为对象检索以及执行查询和扫描的方法。您可以使用对象持久化模型执行数据操作，例如创建、检索、更新和删除项目。您必须先使用服务客户端 API 创建表，然后才能使用对象持久化模型将类映射到表。该 API 已在 Amazon.dynamoDB 中提供。 DataModel 命名空间。
- 服务客户端 API：这是与 DynamoDB API 密切相关的协议级 API。您可以针对所有表和项目操作（例如创建、更新、删除表和项目）使用该低级 API。您还可以查询和扫描自己的表。该 API 在 Amazon.DynamoDB 命名空间中提供。

以下主题更深入地探讨了这三个模型：

使用文档模型

文档模型围绕低级别 .NET API 提供封装类。表和文档是主要的封装类。您可以使用文档模型创建、检索、更新和删除项目。要创建、更新和删除表，您必须使用低级别 API。有关如何使用低级 API 的说明，请参阅使用 DynamoDB 服务级别。 APIs低级 API 已在 Amazon.dynamoDB 中提供。 DocumentModel 命名空间。

要了解有关文档模型的更多信息，请参阅 .NET 文档模型。

创建 DynamoDB 客户端

创建 DynamoDB 客户端：

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

CRUD 操作

保存项目

创建项目：

```
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";
```

将项目保存到 DynamoDB 表中：

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

检索项目

检索项目：

```
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);
}
```

更新项目

如何更新项目:

```
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);
}
```

有条件更新项目：

```
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);
}
```

删除项目

如何删除项目:

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

查询和扫描

查询和检索作者为“Mark Twain”的所有图书：

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

下面的扫描示例代码返回我们的表中的所有图书：

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

使用对象持久化模型

借助适用于 .NET 和 Xamarin 的 AWS Mobile SDK 提供的对象持久化模型，您能够将客户端类映射到 DynamoDB 表。然后，每个对象实例都可以映射到对应表中的项目。为了将客户端对象保存到表中，对象持久化模型提供了 Dynamo DBContext 类，这是 DynamoDB 的入口点。这个类提供到 DynamoDB 的连接，让您能够访问表、执行各种 CRUD 操作，以及执行查询。

对象持久化模型不提供用于创建、更新或删除表的 API。它只提供数据操作。要创建、更新和删除表，您必须使用低级别 API。有关如何使用低级 API 的说明，请参阅使用 DynamoDB 服务级别。 APIs

概述

对象持久化模型提供了一组属性，用于将客户端类映射到表和 properties/fields 表属性。对象持久化模型支持类属性和表属性之间的明确映射和默认映射。

- 显式映射：要将属性映射到主键，必须使用 Dynamo K DBHash ey 和 Dynamo K DBRange ey 对象持久化模型属性。此外，对于非主键属性，如果类中的属性名称与要将其映射到的相应表属性不相同，则必须通过显式添加 Dynamo DBProperty 属性来定义映射。
- 默认映射 – 默认情况下，对象持久化模型会将类属性映射到表中同名的特性。

您无需映射每一个类属性，您可以通过添加 Dynamo DBIgnore 属性来识别这些属性。保存和检索对象的实例会忽略用此特性标记的任何属性。

支持的数据类型

对象持久化模型支持一系列 .NET 基元数据类型、集合数据类型和其他任意数据类型。该模型支持以下基元数据类型。

- 布尔
- 字节
- char
- DateTime
- decimal、double、float
- Int16、Int32、Int64
- SByte
- 字符串
- UInt16, UInt32, UInt64

对象持久化模型也支持 .NET 集合类型，但有以下限制：

- 集合类型必须实现 ICollection 接口。
- 集合类型必须由支持的基元类型组成。例如， ICollection<string> ， ICollection<bool>。

- 集合类型必须提供无参数构造函数。

有关对象持久化模型的更多信息，请参阅 .NET 对象持久化模型。

创建 DynamoDB 客户端

创建 DynamoDB 客户端：

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

CRUD 操作

保存对象

创建对象：

```
[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
};
```

将对象保存到 DynamoDB 表中：

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

检索对象

检索对象：

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

更新对象

更新对象：

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

删除对象

删除对象：

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

查询和扫描

查询和检索作者为“Charles Dickens”的所有图书：

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

下面的扫描示例代码返回我们的表中的所有图书：

```
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());
```

```
  });
}
```

使用 DynamoDB 服务级别 APIs

Dynamo 服务级别 APIs 允许您创建、更新和删除表。您还可以使用该 API 对表中的项目执行典型的创建、读取、更新和删除 (CRUD) 操作。

创建 DynamoDB 客户端

创建 DynamoDB 客户端：

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

CRUD 操作

保存项目

将项目保存到 DynamoDB 表中：

```
// 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);
```

检索项目

检索项目：

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

更新项目

如何更新项目:

```
// 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);
```

删除项目

如何删除项目:

```
// 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);
```

查询和扫描

查询和检索作者为“Mark Twain”的所有图书：

```
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);
    });

  }
}
```

下面的扫描示例代码返回我们的表中的所有图书：

```
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)

使用 SNS 以及适用于 .NET 和 Xamarin 的 AWS Mobile SDK，您可以编写能够接收移动推送通知的应用程序。有关 SNS 的信息，请参阅 Amazon Simple Notification Service。

重要概念

亚马逊 SNS 允许不同设备上的应用程序和最终用户通过移动推送通知（苹果、谷歌和 Kindle Fire 设备）、HTTP/HTTPS, Email/Email-JSON、短信或亚马逊简单队列服务 (SQS) Simple Queue Service 队列或 AWS Lambda 函数接收通知。利用 SNS，可以向订阅了某个主题的大量收件人发送单个或多个消息。

主题

主题是允许收件人动态订阅同一通知的相同副本的“接入点”。一个主题可以支持传送到多个端点类型。例如，可以将 iOS、Android 和 SMS 收件人组成一组。

订阅

要接收发布至主题的消息，您必须订阅一个端点到该主题。端点是可以从 Amazon SNS 接收通知消息的移动应用程序、Web 服务器、电子邮件地址或 Amazon SQS 队列。为端点订阅主题且确认订阅后，此端点会接收向该主题发布的所有消息。

发布

当您发布到一个主题时，SNS 会将正确格式化的消息副本传送给该主题的每个订阅用户。对于移动推送通知，您可以直接发布到端点，或者为端点订阅一个主题。

项目设置

先决条件

要在您的应用程序中使用 SNS，需要将开发工具包添加到您的项目中。为此，请按照设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK 中的说明操作。

设置 SNS 权限

有关设置 SNS 权限的信息，请参阅管理对您的 Amazon SNS 主题的访问。

将 Pack NuGet age for SNS 添加到您的项目中

按照设置适用于.NET 和 Xamarin 的 AWS 移动软件开发工具包中说明的第 4 步，将亚马逊简单通知 NuGet 服务包添加到您的项目中。

将 SNS 与您的应用程序集成

有几种方式可以在您的 Xamarin 中与 SNS 进行交互：

发送推送通知 (Xamarin Android)

本文档介绍了如何使用 Amazon Simple Notification Service (SNS) 和适用于 .NET 和 Xamarin 的 AWS Mobile SDK 将推送通知发送到 Xamarin Android 应用程序。

项目设置

先决条件

在开始本教程前，必须先完成有关设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK 的说明中的所有步骤。

设置 SNS 权限

按照设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK 中第 2 步的说明操作，将下述策略附加到您应用程序的角色中。这样可为您的应用程序提供访问 SNS 的适当权限：

1. 转到 IAM 控制台，然后选择您要配置的 IAM 角色。
2. 单击 “附加策略”，选择 Amazon SNSFull 访问策略，然后单击 “附加策略”。

> **Warning**
>
> 不建议在生产环境中SNSFull使用 Amazon Access。我们在此处使用它是为了让您快速启动并运行。有关为 IAM 角色指定权限的更多信息，请参阅 IAM 角色权限概述。

在 Google Cloud 上启用推送通知

首先，添加一个新的 Google API 项目：

1. 转到 Google Developers Console。
2. 单击 Create Project。
3. 在 New Project 框中，输入项目名称，记下项目 ID (稍后您会用到它)，然后单击 Create。

接下来，为您的项目启用 Google Cloud Messaging (GCM) 服务：

1. 在 Google Developers Console 中，系统应该已经选择了您的新项目。如果没有，请在页面顶部的下拉列表中选择。
2. 从页面左侧的侧栏中选择 APIs & auth。
3. 在搜索框中，键入“Google Cloud Messaging for Android”，然后单击Google Cloud Messaging for Android链接。
4. 单击 Enable API。

最后，获取 API 密钥：

1. 在 Google 开发者控制台中，选择并验证 APIs > 凭据。
2. 在 Public API access 下，单击 Create new key。
3. 在 Create a new key 对话框中，单击 Server key。
4. 在出现的对话框中，单击 Create，然后复制显示的 API 密钥。稍后，您将使用此 API 密钥来执行身份验证。

使用项目 ID 在 SNS 控制台中创建平台 ARN

1. 转到 SNS 控制台。
2. 单击屏幕左侧的 Applications。
3. 单击 Create platform application，以创建新的 SNS 平台应用程序。
4. 输入 Application Name。
5. 对于 Push notification platform，选择 Google Cloud Messaging (GCM)。
6. 将 API 密钥粘贴到标记为 API key 的文本框中。
7. 单击 Create platform application。
8. 选择您刚创建的平台应用程序，然后复制应用程序 ARN。

将 Pack NuGet age for SNS 添加到您的项目中

按照设置适用于.NET 和 Xamarin 的 AWS 移动软件开发工具包中说明的第 4 步，将亚马逊简单通知 NuGet 服务包添加到您的项目中。

创建 SNS 客户端

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

为您的应用程序注册远程通知

要在 Android 上注册远程通知，您需要创建一个可以接收 Google Cloud 消息 BroadcastReceiver 的。请按照下方的提示，更改程序包名称：

```
[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);
      }
}
```

以下是接收来自的推送通知 BroadcastReceiver 并在设备的通知栏上显示通知的服务：

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

将消息从 SNS 控制台发送到端点

1. 转到 SNS 控制台 >“Applications (应用程序)”。
2. 依次选择您的平台应用程序和端点，然后单击 Publish to endpoint。
3. 在文本框中键入文本消息，然后单击 Publish message 发布消息。

发送推送通知 (Xamarin iOS)

本文档介绍了如何使用 Amazon Simple Notification Service (SNS) 以及适用于 .NET 和 Xamarin 的 AWS Mobile SDK 将推送通知发送到 Xamarin iOS 应用程序。

项目设置

先决条件

在开始本教程前，必须先完成有关设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK 的说明中的所有步骤。

设置 SNS 权限

按照设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK 中第 2 步的说明操作，将下述策略附加到您应用程序的角色中。这样可为您的应用程序提供访问 SNS 的适当权限：

1. 转到 IAM 控制台，然后选择您要配置的 IAM 角色。
2. 单击 “附加策略”，选择 Amazon SNSFull 访问策略，然后单击 “附加策略”。

> **Warning**
>
> 不建议在生产环境中SNSFull使用 Amazon Access。我们在此处使用它是为了让您快速启动并运行。有关为 IAM 角色指定权限的更多信息，请参阅 IAM 角色权限概述。

获得 Apple iOS 开发人员计划成员资格

您需要在物理设备上运行您的应用程序以接收推送通知。要在设备上运行您的应用程序，您必须拥有 Apple iOS 开发人员计划成员资格。一旦您拥有了成员资格，就可以使用 Xcode 生成签名身份。有关更多信息，请参阅 Apple 的 App Distribution Quick Start 文档。

创建 iOS 证书

首先，您需要创建一个 iOS 证书。然后，您需要创建为推送通知而配置的预置配置文件。为此，请执行以下操作：

1. 转至 Apple Developer Member Center，单击 Certificates, Identifiers & Profiles。
2. 单击 iOS Apps 下的 Identifiers，单击 Web 页面右上角的加号按钮以添加一个新的 iOS 应用程序 ID，然后输入应用程序 ID 描述。
3. 向下滚动到 Add ID Suffix 部分，选择 Explicit App ID，然后输入您的服务包标识符。
4. 向下滚动到 App Services 部分，并选择 Push Notifications。
5. 单击继续。
6. 单击 Submit (提交)。
7. 单击完成。
8. 选择您刚刚创建的应用程序 ID，然后单击 Edit。
9. 向下滚动到 Push Notifications 部分。单击 Development SSL Certificate 下的 Create Certificate。
10. 按照说明创建证书签名请求 (CSR)、上传请求、下载将用于与 Apple Notification Service (APNS) 通信的 SSL 证书。
11. 返回到证书、身份和配置文件页面。单击 Provisioning Profiles 下的 All。
12. 单击右上角的 + 按钮以添加新的预置配置文件。
13. 选择 iOS App Development，然后单击 Continue。
14. 选择您的应用程序 ID，然后单击 Continue。
15. 选择您的开发人员证书，然后单击 Continue。
16. 选择您的设备，然后单击 Continue。
17. 输入配置文件名称，然后单击 Generate。
18. 下载预置文件后双击以安装预置配置文件。

有关预配置为推送通知而配置的配置文件的更多信息，请参阅 Apple 的配置推送通知文档。

使用证书在 SNS 控制台中创建平台 ARN

1. 运行 KeyChain 访问应用程序，选择屏幕左下角的我的证书，然后右键单击您为连接到 APNS 而生成的 SSL 证书并选择导出。系统将提示您指定文件的名称和密码以保护证书。证书将保存在 P12 文件中。
2. 转至 SNS Console，单击屏幕左侧的 Applications (应用程序)。
3. 单击 Create platform application，以创建新的 SNS 平台应用程序。
4. 输入 Application Name。
5. 对于 Push notification platform，选择 Apple Development。
6. 单击 Choose File，选择导出 SSL 证书时创建的 P12 文件。
7. 输入在导出 SSL 证书时指定的密码，然后单击 Load Credentials From File。
8. 单击 Create platform application。
9. 选择您刚创建的平台应用程序，然后复制应用程序 ARN。在后续步骤中，您将需要此信息。

将 Pack NuGet age for SNS 添加到您的项目中

按照设置适用于.NET 和 Xamarin 的 AWS 移动软件开发工具包中说明的第 4 步，将亚马逊简单通知NuGet服务包添加到您的项目中。

创建 SNS 客户端

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

为您的应用程序注册远程通知

要注册应用程序，请调用 RegisterForRemoteNotifications 您的 UIApplication对象，如下所示。将以下代码放在 AppDelegate .cs 中，在出现以下提示的地方插入您的平台应用程序 ARN：

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

将消息从 SNS 控制台发送到端点

1. 转到 SNS 控制台 >“Applications (应用程序)”。
2. 依次选择您的平台应用程序和端点，然后单击 Publish to endpoint。
3. 在文本框中键入文本消息，然后单击 Publish message 发布消息。

发送和接收 SMS 通知

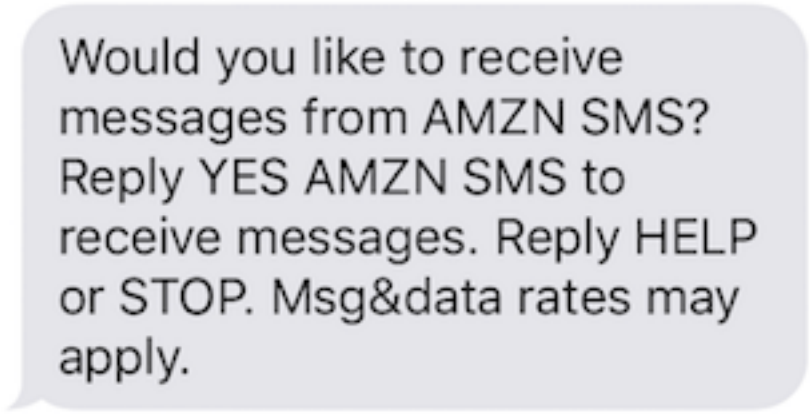
您可以使用 Amazon Simple Notification Service (Amazon SNS) 向支持短信服务 (SMS) 的移动电话和智能手机发送和从中接收 SMS 通知。

> **Note**
>
> 在美国地区，手机号码现已支持接收 SMS 通知。只能从在美国东部（弗吉尼亚州北部）区域创建的主题发送 SMS 消息，但可以从任何其他区域向在美国东部（弗吉尼亚州北部）区域创建的主题发布消息。

创建主题

要创建主题，请执行以下操作:

1. 在 Amazon SNS 控制台中，单击创建新主题。随即出现“Create new topic”对话框。
2. 在 Topic name 框中，键入主题名称。
3. 在“显示名称”框中，键入显示名称。主题必须有一个配置的显示名称因为显示名称的前十（10）字符要用作是文本消息前缀的初始部分。您输入的显示名称将出现在 SNS 发送给用户的确认消息中（下面的显示名称为“AMZN SMS”)。



1. 单击创建主题。新主题将显示在 Topics 页面中。
2. 选择新主题，然后单击主题 ARN。此时将会显示 Topic Details（主题详细信息）页。
3. 复制主题 ARN，因为下一步您订阅主题时需要用到它。

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

使用 SMS 协议订阅主题

创建 SNS 客户端，传递您的凭证对象和身份池区域：

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

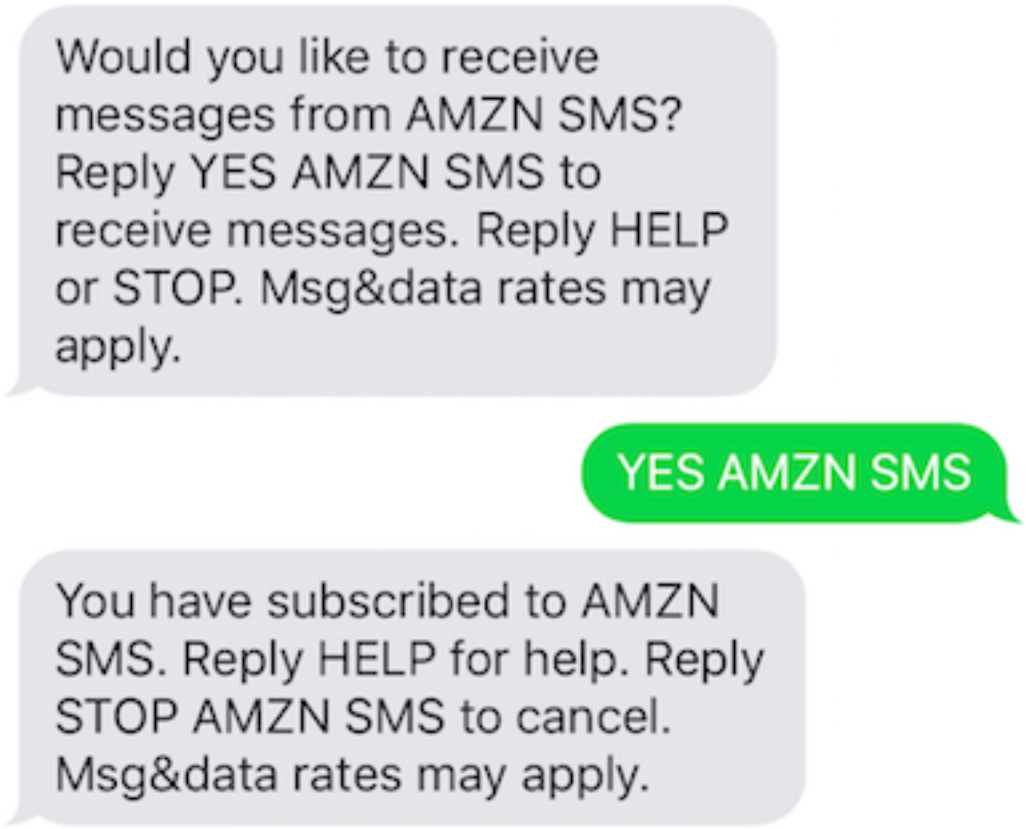
要订阅主题，请调用 SubscribeAsync 并向它传递您要订阅的主题的 ARN、协议（“sms”）和电话号码：

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

您将在订阅响应对象中收到订阅 ARN。您的订阅 ARN 如下面这样：

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

当设备订阅主题时，SNS 会发送一条确认消息给设备，用户必须确认他们是否要接收通知，如下所示：



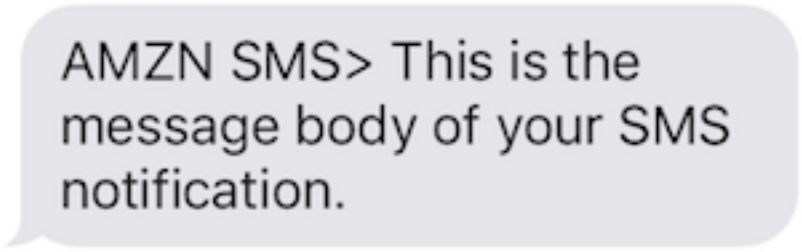
用户订阅主题后，当您将 SMS 消息发布到该主题时，他们会收到这些消息。

发布消息

向一个主题发布消息:

1. 登录 AWS 管理控制台，打开 Amazon SNS 控制台。
2. 在左侧导航窗格中，单击 Topics，然后选择您要向其发布消息的主题。
3. 单击 Publish to topic。
4. 在“Subject”框中，键入一个主题。

5. 在“消息”框中，键入消息。Amazon SNS 会将您在“消息”框中输入的文本发送给 SMS 订阅用户，除非您在“主题”框中也输入了文本。因为 Amazon SNS 会在您发送的所有 SMS 消息前都加上一个显示名称前缀，所以显示名称前缀与消息的有效负荷加起来不能超过总共 140 个 ASCII 字符或 70 个 Unicode 字符。Amazon SNS 会截断超出这一限制的消息。
6. 单击发布消息。Amazon SNS 将显示一个确认对话框。SMS 消息显示在您的支持 SMS 的设备上，如下所示。



向 HTTP/HTTPS 终端发送消息

您可以使用 Amazon SNS 向一个或多个 HTTP 或 HTTPS 端点发送通知消息。流程如下：

1. 配置您的终端节点以接收 Amazon SNS 消息。
2. 为 HTTP/HTTPS 终端节点订阅主题。
3. 确认订阅。
4. 将通知发布到该主题。Amazon SNS 然后发送一个 HTTP POST 请求，向已订阅终端节点传递通知内容。

将您的 HTTP/HTTPS 终端节点配置为接收 Amazon SNS 消息

按照向终端节点发送 Amazon SNS 消息的步骤 1 中的说明配置您的 HTTP/HTTPS 终端节点。

将您的 HTTP/HTTPS 终端节点订阅您的 Amazon SNS 主题

创建 SNS 客户端，传递您的凭证对象和身份池区域：

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

要通过主题向 HTTP 或 HTTPS 终端节点发送消息，必须为终端节点订阅 Amazon SNS 主题。您可以使用终端节点的 URL 指定终端节点：

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

确认订阅

完成为终端节点订阅后，Amazon SNS 会向该终端节点发送一条订阅确认消息。终端节点上的代码必须检索订阅确认消息中的 SubscribeURL值，并访问 SubscribeURL 自身指定的位置，或使其可供您使用，这样您就可以手动访问 SubscribeURL (例如，使用 Web 浏览器)。

在订阅得到确认前，Amazon SNS 不会向终端节点发送消息。当您访问 SubscribeURL时，该响应将包括 XML 文档，该文档包含指定订阅 ARN 的元素 SubscriptionArn。

向 HTTP/HTTPS 终端节点发送消息

您可以通过发布到主题的方式，向主题的订阅发送消息。调用 PublishAsync 并向其传递主题 ARN 和您的消息。

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

SNS 故障排除

使用 Amazon SNS 控制台中的传达状态

Amazon SNS 控制台包含一个传达状态功能，允许您收集消息传送到移动推送通知平台 (Apple (APNS)、Google (GCM)、Amazon (ADM)、Windows (WNS 和 MPNS)、Baidu) 是成功还是失败的反馈信息。

它还提供其他重要信息，如在 Amazon SNS 中的驻留时间等。这些信息是在亚马逊 CloudWatch 日志组中捕获的，当通过亚马逊 SNS 控制台或亚马逊 SNS 启用此功能时，该组由 Amazon SNS 自动创建。 APIs

有关使用送达状态功能的说明，请参阅 AWS Mobile 博客中的使用 Amazon SNS 的送达状态功能。

使用适用于 .NET 和 Xamarin 的 AWS Mobile SDK 的最佳实践

在使用适用于 .NET 和 Xamarin 的 AWS Mobile SDK 时，了解几个基本原则和最佳实践会对您大有帮助。

- 使用 Amazon Cognito 来获取 AWS 凭证，而不是将凭证硬编码到应用程序中。如果将凭证硬编码到应用程序中，结果可能使凭证暴露于公众，使得他人能够利用您的凭证调用 AWS。有关如何使用 Amazon Cognito 获得 AWS 凭证的说明，请参阅设置适用于 .NET 和 Xamarin 的 AWS Mobile SDK。
- 有关使用 S3 的最佳实践，请参阅 AWS 博客上的这篇文章。
- 有关使用 DynamoDB 的最佳实践，请参阅《DynamoDB 开发人员指南》中的 DynamoDB 最佳实践。

我们一直在努力帮助客户取得成功，并欢迎提供反馈，因此请随时在 AWS 论坛上发帖或在上面提出问题 GitHub。

AWS 服务文档库

适用于 .NET 和 Xamarin 的 AWS Mobile SDK 中的每项服务都配有单独的开发人员指南和服务 API 参考，它们提供了或许对您有帮助的额外信息。

Amazon Cognito Identity

- Cognito 开发人员指南
- Cognito Identity Service API 参考

Amazon Cognito Sync

- Cognito 开发人员指南
- Cognito Sync Service API 参考

Amazon Mobile Analytics

- Mobile Analytics 开发人员指南
- Mobile Analytics Service API 参考

Amazon S3

- S3 开发人员指南
- S3 入门指南
- S3 Service API 参考

Amazon DynamoDB

- DynamoDB 开发人员指南
- DynamoDB 入门指南
- DynamoDB Service API 参考

Amazon Simply Notification Service (SNS)

- SNS 开发人员指南
- SNS Service API 参考

其他有用链接

- AWS 术语表
- 关于 AWS 凭证

故障排除

本主题提供了几种帮助您排查在使用适用于 .NET 和 Xamarin 的 AWS Mobile SDK 时可能遇到的问题的思路。

确保 IAM 角色具有所需权限

在调用 AWS 服务时，您的应用程序应使用来自 Cognito 身份池的身份。池中的每个身份都与一个 IAM (Identity and Access Management) 角色相关联。

一个角色具有一个或多个与之关联的策略文件，用来指定分配给该角色的用户可以访问哪些 AWS 资源。默认情况下，为每个身份池创建两个角色：一个用于经过身份验证的用户，另一个用于未经身份验证的用户。

您需要修改现有策略文件，或将新策略文件与应用程序所需的权限相关联。如果您的应用程序支持经过身份验证和未经身份验证的用户，则您必须为这两个角色授予相应权限，使其能够访问您的应用程序所需的 AWS 资源。

以下策略文件展示了如何授予对 S3 存储桶的访问权限：

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

以下策略文件展示了如何授予对 DynamoDB 数据库的访问权限：

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

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

有关如何指定策略的更多信息，请参阅 IAM 策略。

使用 HTTP 代理调试程序

如果您的应用程序正在调用的 AWS 服务具有 HTTP 或 HTTPS 终端节点，则可以使用 HTTP/HTTPS 代理调试器查看请求和响应，以更深入地了解正在发生的事情。我们提供有多种 HTTP 代理调试程序，例如：

- Charles – 适用于 Windows 和 OSX 的 Web 调试代理
- Fiddler – 适用于 Windows 的 Web 调试代理

Charles 和 Fiddler 都需要一些配置才能查看 SSL 加密的流量，请阅读此类工具的相关文档，以进一步了解相应信息。如果您使用的 Web 调试代理无法配置为显示加密流量，请打开 aws_endpoints_json 文件，将需要调试的 AWS 服务的 HTTP 标签设置为 true。

文档历史记录

下表描述了自适用于 .NET 和 Xamarin 的 AWS Mobile SDK上一次发布以来对文档所做的重要更改。

- API 版本：2015-08-27
- 文档上次更新时间：2021-02-23

更改	API 版本	描述	发行日期
已存档	2015-08-27	适用于 Xamarin 的 AWS 移动 SDK 包含在。 适用于 .NET 的 AWS SDK本指南参考了适用于 Xamarin 的 Mobile SDK 的存档版本。	2021-02-23
GA 版本	2015-08-27	GA 版本	2015-08-27
Beta 版本	2015-07-28	Beta 版本	rn2015-07-28