开发人员指南

AWS 适用于 Unity 的移动 SDK

AWS 适用于 Unity 的移动 SDK: 开发人员指南

Table of Contents

适用于 Unity 的 AWS 移动 SDK 现已包含在 适用于 .NET 的 AWS SDK。本指南引用适用于 Unity 的 Mobile SDK 的存档版本。有关更多信息，请参阅 适用于 Unity 的 AWS 移动 SDK 是什么？。

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。

适用于 Unity 的 AWS 移动 SDK 是什么？

适用于 Unity 的 AWS 移动 SDK 现已包含在 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。

本指南已不再更新，其中引用适用于 Unity 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 Unity 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《适用于 .NET 的 AWS SDK 开发者指南》中的 Unity 支持的特殊注意事项。
- 为了便于参考，您可以在上找到适用于 Unity 的AWS 移动 SDK 的存档版本 GitHub。

存档的参考内容

存档的 Unity 移动 SDK 包含一组.NET 类，这些类允许使用 Unity 编写的游戏使用 AWS 服务。采用适用于 Unity 的 Mobile SDK 编写的应用程序可以在 iOS 或 Android 设备上运行。

支持的 AWS 服务包括：

- Amazon Cognito
- Amazon DynamoDB
- AWS Identity and Access Management (IAM)
- Amazon Kinesis Data Streams
- AWS Lambda
- Amazon Mobile Analytics
- Amazon Simple Email Service（Amazon SES）
- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Queue Service (Amazon SQS)
- Amazon Simple Storage Service（Amazon S3）

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，发送推送通知，及收集和分析使用率数据。

兼容性

适用于 Unity v3 的 Mobile SDK 与 Unity 4.6 版及更高版本兼容。

适用于 Unity 的 Mobile SDK 的最新版本增加了一些改进功能。要将此类功能添加到您的项目中，您可能需要更改代码。有关这些变更的更多信息，请参阅前端 Web 和AWS 移动博客上的 Unity 移动 SDK for Unity 的改进。

下载适用于 Unity 的 Mobile SDK

您也可以在此处以 .zip 文件的形式下载适用于 Unity 的 Mobile SDK。

适用于 Unity 的 Mobile SDK 中包含哪些内容？

有关适用于 Unity 的移动 SDK 中 NuGet 软件包、示例和其他文件的完整列表，请参阅适用于 .NET 的 AWS SDK GitHub。

设置适用于 Unity 的 AWS Mobile SDK

要开始使用适用于 Unity 的 AWS Mobile SDK，您可以设置该 SDK 并开始构建一个新项目，也可以将该 SDK 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。

先决条件

在可以使用适用于 Unity 的 AWS Mobile SDK 之前，您需要以下内容：

- 一个 AWS 账户
- Unity 4.x 或 5.x 版 (如果您希望编写在 64 位 iOS 上运行的应用程序，则需要 Unity 4.6.4p4 或 Unity 5.0.1p3)

满足先决条件后，您需要执行以下操作来开始使用该 SDK：

1. 下载适用于 Unity 的 AWS Mobile SDK。
2. 配置适用于 Unity 的 AWS Mobile SDK。
3. 使用 Amazon Cognito 获取 AWS 凭证。

第 1 步：下载适用于 Unity 的 AWS Mobile SDK

首先，下载适用于 Unity 的 AWS Mobile SDK。该 SDK 中的每个软件包是使用与软件包名称对应的 AWS 服务所必需的。例如， aws-unity-sdk-dynamodb-2.1.0.0.unitypackage 包用于调用 AWS DynamoDB 服务。您可以导入所有包，也可以仅导入要使用的包。

1. 打开 Unity 编辑器，新建一个空项目并使用默认设置。
2. 选择 Assets > Import Package > Custom Package。
3. 在 Import package 对话框中，导航到您要使用的 .unitypackage 文件并选择这些文件。
4. 在 Importing package 对话框中，确保选中所有项目，然后单击 Import。

第 2 步：配置适用于 Unity 的 AWS Mobile SDK

创建场景

使用适用于 Unity 的 AWS Mobile SDK 时，您可以首先在 mono 行为类的 `Start` 或 `Awake` 方法中包含以下代码行：

```
UnityInitializer.AttachToGameObject(this.gameObject);
```

通过从 File 菜单中选择 New Scene 创建场景。

适用于 Unity 的 AWS SDK 包含用于其支持的每个 AWS 服务的客户端类。这些客户端使用名为 awsconfig.xml 的文件进行配置。下面一部分介绍 awsconfig.xml 文件中最常用的设置。有关这些设置的更多信息，请参阅《Unity SDK API 参考》。

设置默认 AWS 服务区域

为所有服务客户端配置默认区域：

```
<aws region="us-west-2" />
```

这会为 Unity SDK 中的所有服务客户端设置默认区域。通过在创建服务客户端的实例时明确指定区域，可以覆盖此设置，如下所示：

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

设置日志记录信息

按如下所示指定日志记录设置：

```
<logging logTo="UnityLogger"
         logResponses="Always"
         logMetrics="true"
         logMetricsFormat="JSON" />
```

此设置用于在 Unity 中配置日志记录。当您将日志记录到 `UnityLogger` 时，该框架会在内部将输出发送到调试日志。如果要记录 HTTP 响应，请设置 LogResponses 标志，其值可以是 “始终”、“从

不” 或。 OnError您还可以使用 LogMetrics 属性记录 HTTP 请求的性能指标，日志格式可以使用 LogMetricsFormat属性指定，有效值为 JSON 或标准。

以下示例演示 awsconfig.xml 文件中最常用的设置。有关特定服务设置的更多信息，请参阅以下服务部分：

```
<?xml version="1.0" encoding="utf-8"?>
<aws region="us-west-2"
    <logging logTo="UnityLogger"
             logResponses="Always"
             logMetrics="true"
             logMetricsFormat="JSON" />
/>
```

使用 link.xml 文件

该 SDK 对特定于平台的组件使用反射。如果您使用的是 IL2 CPP脚本后端，strip bytecode则在 iOS上始终处于启用状态，因此您需要在程序集根目录中有一个包含以下条目的link.xml文件：

```
<linker>
<!-- if you are using AWSConfigs.HttpClient.UnityWebRequest option-->
<assembly fullname="UnityEngine">
    <type fullname="UnityEngine.Networking.UnityWebRequest" preserve="all" />
    <type fullname="UnityEngine.Networking.UploadHandlerRaw" preserve="all" />
    <type fullname="UnityEngine.Networking.UploadHandler" preserve="all" />
    <type fullname="UnityEngine.Networking.DownloadHandler" preserve="all" />
    <type fullname="UnityEngine.Networking.DownloadHandlerBuffer" preserve="all" />
</assembly>
<assembly fullname="mscorlib">
    <namespace fullname="System.Security.Cryptography" preserve="all"/>
</assembly>
<assembly fullname="System">
    <namespace fullname="System.Security.Cryptography" preserve="all"/>
</assembly>
<assembly fullname="AWSSDK.Core" preserve="all"/>
<assembly fullname="AWSSDK.CognitoIdentity" preserve="all"/>
<assembly fullname="AWSSDK.SecurityToken" preserve="all"/>
add more services that you need here...
</linker>
```

第 3 步：使用 Amazon Cognito 获取身份池 ID

要在移动应用程序中使用 AWS 服务，您必须使用 Amazon Cognito Identity 获取身份池 ID。使用 Amazon Cognito 获取身份池 ID 可让您的应用程序能够直接访问 AWS 服务，而不必在应用程序中嵌入您的私有凭证。这还允许您设置权限来控制用户有权访问的 AWS 服务。

要开始使用 Amazon Cognito，您必须创建一个身份池。身份池是用于存储特定于您的账户的用户身份数据的存储区。每个身份池都有可配置的 IAM 角色，您可以使用这些角色来指定应用程序的用户可以访问的 AWS 服务。通常情况下，开发人员对每个应用程序使用一个身份池。有关身份池的更多信息，请参阅 Amazon Cognito 开发人员指南。

为应用程序创建身份池：

1. 登录 Amazon Cognito 控制台，然后单击 Create new identity pool。
2. 输入身份池的名称，并选中“启用未经验证的身份的访问权限”复选框。单击 Create Pool (创建池) 创建身份池。
3. 单击允许可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 Cognito Sync 和 Mobile Analytics 提供身份池访问权限。

下一页将显示一段代码，该段代码用于创建凭证提供程序，以便您可以轻松地将 Cognito Identity 与 Unity 应用程序集成。您可以将凭证提供商对象传递给所使用的 AWS 客户端的构造函数。代码如下所示：

```
CognitoAWSCredentials credentials = new CognitoAWSCredentials (
    "IDENTITY_POOL_ID", // Identity Pool ID
    RegionEndpoint.USEast1 // Region
);
```

后续步骤

- 入门：阅读《适用于 Unity 的 AWS Mobile SDK 入门》，详细了解该 SDK 中包含的服务。
- 运行演示：查看演示常见使用案例的示例 Unity 应用程序。要运行示例应用，请按如上所述设置适用于 Unity 的 SDK ，然后按照各个示例的 README 文件中的说明进行操作。
- 阅读 API 参考：查看适用于 Unity 的 AWS Mobile SDK 的 API 参考。
- 提问：在 AWS Mobile SDK 论坛上发布问题，或者在 Github 上提出问题。

适用于 Unity 的 AWS Mobile SDK 入门

本页概述了适用于 Unity 的 AWS Mobile SDK 中的每种 AWS 服务，并说明了如何设置 Unity 示例。开始使用以下服务前，您必须完成安装适用于 Unity 的 AWS Mobile SDK 页面上说明的所有步骤。

Amazon Cognito Identity

对 AWS 的所有调用都需要 AWS 凭证。我们建议您使用 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。请按照安装适用于 Unity 的 AWS Mobile SDK 中的说明，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。通过新建与 IAM 角色相关联的身份池，可以配置 Cognito。IAM 角色用于指定应用程序可能会访问的资源/服务。

要开始使用 Cognito Identity，请参阅 Amazon Cognito 开发人员指南。

Amazon Cognito Sync

使用 Cognito Sync，您可以轻松将最终用户数据（如用户偏好或游戏状态）保存到 AWS Cloud，以便用户可以从任何设备访问这些数据。Cognito 也可以将此类数据保存在本地，使您的应用即使在 Internet 连接不可用时也能工作。当 Internet 连接变得可用时，您的应用会将其本地数据同步到云。

要开始使用 Cognito Sync，请参阅 Amazon Cognito 开发人员指南。

使用示 CognitoSyncManager例

在 “项目” 窗格中，导航到 Ass et s//examples/CognitoSync，然后在窗格的右侧选择要打开CognitoSync场景的场景。AWSSDK

要运行该示例，请单击编辑器屏幕顶部的播放按钮。当应用运行时，它会显示几个文本框和按钮，您可以在这些文本框中输入一些玩家信息。文本框下面是一系列按钮，这些按钮用于将玩家信息保存在本地、将本地玩家信息与 Cognito 云同步、从 Cognito 云刷新玩家信息，以及删除本地玩家信息。按每个按钮可执行相应的操作。该示例在游戏屏幕的顶部显示反馈。

要配置 CognitoSyncManager 示例，必须指定 Cognito 身份池 ID。要指定此值，请在 Unity 编辑器中，在 “层次结构” 面板SyncManager中选择，然后将其输入到 Insp ector 窗格的 IDENTIT Y_POOL_ID 文本框中。

> **Note**
>
> 该 CognitoSyncManager 示例包含的代码说明了如何使用 Facebook 身份提供商搜索 “USE_FACEBOOK_LOGIN” 宏。这要求使用适用于 Unity 的 Facebook SDK。有关更多信息，请参阅适用于 Unity 的 Facebook SDK。

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。

适用于 Unity 的 AWS SDK 提供了低级和高级库，这两种库均可与 DynamoDB 一起使用。高级库包括 DynamoDB Object Mapper，它允许您将客户端类映射到 DynamoDB 表，执行各种创建、读取、更新和删除 (CRUD) 操作，以及执行查询。使用 DynamoDB Object Mapper，您可以编写简单的可读代码，将对象存储在云中。

有关 DynamoDB 的更多信息，请参阅 DynamoDB 开发人员指南。

有关从 Unity 应用程序使用 Dynamo DB 的更多信息，请参阅 Amazon DynamoDB。

使用 DynamoDB 示例

在 “项目” 窗格中，导航到 “资产”/“//示例” AWSSDK/“DynamoDB”。本示例由以下场景组成：

- Dynamo DBExample -应用程序的初始场景
- LowLevelDynamoDbExample -使用低级 DynamoDBD API 的示例
- TableQueryAndScanExample -显示如何执行查询的示例
- HighLevelExample -使用高级 DynamoDB API 的示例

通过使用“Build Settings”对话框 (通过选择“File”>“Build Settings”打开)，将这些场景添加到生成中 (按它们出现的顺序)。此示例创建了四个表： ProductCatalog、论坛、话题、回复。

要运行该示例，请单击编辑器屏幕顶部的播放按钮。当应用运行时，它会显示多个按钮：

- 低级表操作 – 演示如何创建、列出、更新、描述和删除表。
- 中级查询和扫描操作 – 演示如何执行查询。
- 高级 Object Mapper – 演示如何创建、更新和删除对象。

Mobile Analytics

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。适用于 Unity 的 AWS SDK 可与 Amazon Mobile Analytics 服务集成。有关 Mobile Analytics 的信息，请参阅 Mobile Analytics 用户指南。有关从 Unity 应用程序使用 Mobile Analytics 的更多信息，请参阅 Amazon Mobile Analytics。

配置 Mobile Analytics

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

```
<mobileAnalytics sessionTimeout = "5"
                 maxDBSize = "5242880"
                 dbWarningThreshold = "0.9"
                 maxRequestSize = "102400"
                 allowUseDataNetwork = "false"/>
```

- sessionTimeout – 从应用程序进入后台到可以终止会话之间的时间间隔。
- 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。
- dbWarningThreshold -这是数据库的大小限制，一旦达到该限制，就会生成警告日志。
- 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）。
- allowUseDataNetwork-一个布尔值，用于指定会话事件是否在数据网络上发送。

使用 Mobile Analytics 示例

在 “项目” 窗格中，导航到 “资产//示例 AWSSDK/Mobile Analytics”，然后在窗格的右侧选择 Amazon Mobile Analytics 示例场景以打开场景。要使用该示例，您需要使用 Amazon Mobile Analytics 控制台添加应用程序。有关使用 Mobile Analytics 控制台的更多信息，请参阅 Amazon Mobile Analytics 用户指南。

运行前，请执行下列步骤配置示例：

1. 选择 AmazonMobileAnalyticsSample 游戏对象。
2. 在“App Id (应用程序 ID)”字段中指定应用程序 ID（在 Amazon Mobile Analytics 控制台中创建）。
3. 在“Cognito Identity Pool Id (Cognito 身份池 ID)”字段中指定 Cognito 身份池 ID（使用 Amazon Cognito 控制台创建）。
4. 确保经过身份验证和未经过身份验证的角色都有权访问 Mobile Analytics 服务。有关对 IAM 角色应用策略的更多信息，请参阅管理角色。

请注意，当运行示例应用程序时，事件可能无法立即传送到后端服务。后台线程会将事件缓存到本地，然后定期 (默认值为 60 秒) 将它们分批发送到 Amazon Mobile Analytics 后端，确保您的游戏性能不会受到不利影响。由于 Amazon Mobile Analytics 会对数据进行复杂的处理，在初次提交后长达 60 分钟的时间内，AWS 管理控制台中可能看不到已提交的事件和对应的报告。

有关 Amazon Mobile Analytics 提供的报告的更多信息，请参阅报告和移动指标。

Amazon S3

Amazon Simple Storage Service (Amazon S3) 为开发人员和 IT 团队提供安全、持久、高度可扩展的对象存储。您可以从 Unity 使用 S3 来存储、列出和检索图像、视频、音乐以及游戏使用的其他数据。

有关 S3 的更多信息，请参阅 Amazon S3 和 S3 入门。

有关从 Unity 应用程序使用 S3 的更多信息，请参阅 Amazon Simple Storage Service (S3)。

配置 S3 默认签名

按如下说明配置 S3 默认签名：

```
<s3 useSignatureVersion4="true" />
```

用于指定是否应对 S3 请求使用签名版本 4。

使用 S3 示例

在 “项目” 窗格中，导航到 Ass et s AWSSDK//examples/S3，然后在窗格的右侧选择 S3 Example 场景以打开场景。该示例演示如何列出存储桶、列出存储桶中的对象、将对象发布到存储桶，以及从存储桶下载对象。运行前，请执行下列步骤配置示例：

1. 在 Hierarchy 窗格中选择 S3 游戏对象。
2. 在 Inspector 窗格中输入 S 3 BucketName 和的值SampleFileName。S3 BucketName 是示例使用的存储桶的名称，S3 SampleFileName 是示例将上传到指定 S3 存储桶中的文件的名称。
3. 确保经过身份验证和未经过身份验证的角色都有权访问账户中的 S3 存储桶。有关对 IAM 角色应用策略的更多信息，请参阅管理角色。

要运行该示例，请单击编辑器屏幕顶部的播放按钮。当应用运行时，它会显示多个按钮：

- 获取对象 – 获取 AWS 账户中所有存储桶中全部对象的列表。
- 获取存储桶 – 获取 AWS 账户中所有存储桶的列表。
- 发布对象 – 将对象上传到指定的 S3 存储桶。
- 删除对象 – 从指定的 S3 存储桶删除所有对象。

该示例在游戏屏幕的顶部显示反馈。

Amazon Simple Notification Service

Amazon Simple Notification Service 是一项快速、灵活、完全托管的推送通知服务，可以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 Amazon Simple Notification Service，请参阅 Amazon Simple Notification Service。

AWS Lambda

AWS Lambda 是一个计算服务，它运行您的代码来响应请求或事件，并且自动为您管理计算资源，使构建快速响应新信息的应用程序变得容易。AWS Lambda 函数可以直接从移动设备、物联网和 Web 应用调用，并且可以同步发回响应，因此无需预配置或管理基础设施，就可方便地为移动应用程序创建可扩展、安全且高度可用的后端。有关更多信息，请参阅 AWS Lambda。

Amazon Cognito Identity

什么是 Amazon Cognito Identity？

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。Cognito 还支持已经过开发人员验证的身份，借助该身份，您可以注册用户并通过自己的后端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。

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 Amazon Cognito Identity 区域可用性。

使用公共提供商对用户进行身份验证

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

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

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 跨设备同步用户数据。要在应用程序中使用 Cognito Sync，必须在项目中包括适用于 Unity 的 AWS Mobile SDK 。

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

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。有关 Mobile Analytics 的信息，请参阅 AWS Mobile Analytics。

集成 Amazon Mobile Analytics

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。

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

在 Mobile Analytics 控制台中创建应用程序时，您需要指定 Cognito 身份池 ID。要创建新的 Cognito 身份池并生成一个 ID，请参阅 Cognito Identity 开发人员指南。

将 Mobile Analytics 集成到应用程序

要从 Unity 访问 Mobile Analytics，您将需要以下 using 语句：

```
using Amazon.MobileAnalytics.MobileAnalyticsManager;
using Amazon.CognitoIdentity;
```

最佳做法是使用 Amazon Cognito 向您的应用程序提供临时的 AWS 凭证。这些凭证使得应用程序能够访问您的 AWS 资源。要创建凭证提供程序，请按照 Amazon Cognito Identity 上的说明操作。

使用以下信息 MobileAnalyticsManager 实例化实例：

- cognitoIdentityPoolID-你的应用程序的 Cognito 身份池的 ID
- CognitoRegion-您的 Cognito 身份池所在的区域，例如 “。RegionEndpoint USEast1”
- 区域-Mobile Analytics 服务的区域，例如 “RegionEndpoint。 USEast1”

- appId – 当您添加应用程序时，Mobile Analytics 控制台生成的值

使用初始 MobileAnalyticsClientContextConfig 化**MobileAnalyticsManager**实例，如以下代码片段所示：

```
// Initialize the MobileAnalyticsManager
void Start()
{
    // ...
    analyticsManager = MobileAnalyticsManager.GetOrCreateInstance(
        new CognitoAWSCredentials(<cognitoIdentityPoolId>, <cognitoRegion>),
        <region>,
        <appId>);
    // ...
}
```

> **Note**
>
> 应用程序 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。

appId 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 Unity 的 SDK 提供了 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，假设您的应用程序是一个游戏，并且您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

当应用程序失去焦点时，您可以暂停会话。在 OnApplicationFocus 中，检查应用程序是否暂停。如果暂停，则调用 PauseSession，否则调用 ResumeSession，如以下代码段所示：

```
void OnApplicationFocus(bool focus)
{
    if(focus)
    {
        analyticsManager.ResumeSession();
    }
    else
    {
        analyticsManager.PauseSession();
```

```
    }
}
```

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。可在 awsconfig.xml 文件中配置此设置。有关更多信息，请参阅《适用于 Unity 的 AWS Mobile SDK 入门》的“配置 Mobile Analytics”部分。

Amazon Simple Storage Service (S3)

Amazon Simple Storage Service (Amazon S3) 为开发人员和 IT 团队提供安全、持久、高度可扩展的对象存储。Unity 开发人员可以利用 S3 来动态加载其游戏所使用的资产。这使得从一开始就能更快地从应用程序商店下载游戏。

有关 S3 的更多信息，请参阅 Amazon S3。

有关 AWS S3 区域可用性的信息，请参阅 AWS 服务区域可用性。

> **Note**
>
> 本文档中的一些示例假设使用名为的文本框变量 ResultText 来显示跟踪输出。

创建和配置 S3 存储桶

Amazon S3 可将您的资源存储到 Amazon S3 存储桶 (存在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。

创建 S3 存储桶

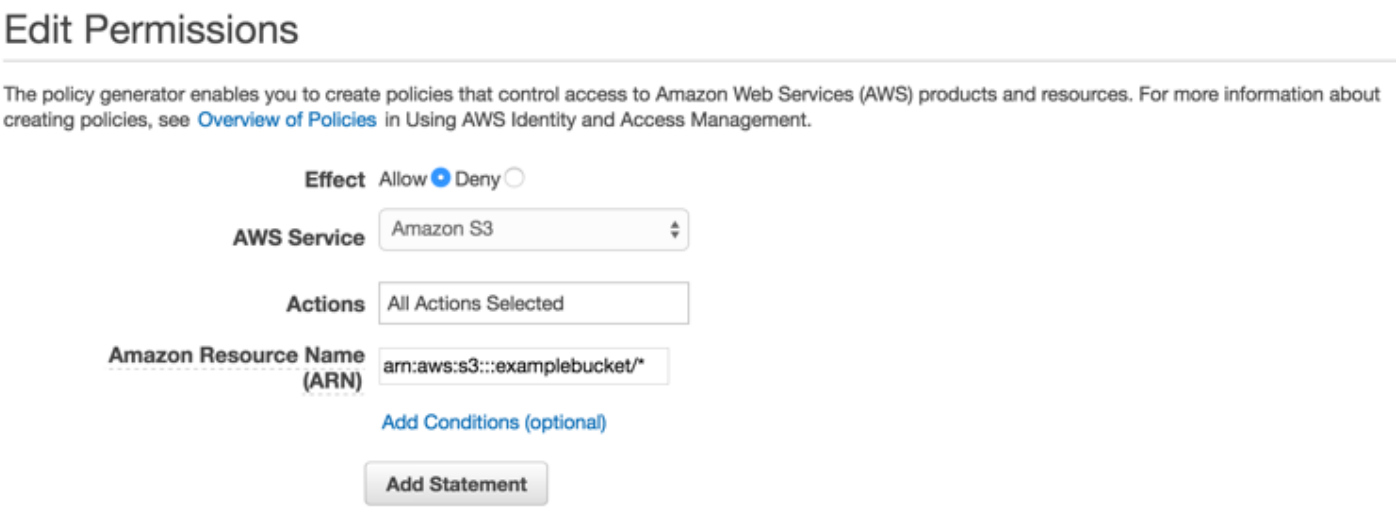
1. 登录 Amazon S3 控制台，然后单击 Create Bucket。
2. 输入存储桶的名称，选择一个区域，然后单击 Create。

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。

从控制台上传文件

将测试文件上传到您的存储桶：

1. 在 S3 控制台上的存储桶视图中，单击 Upload。
2. 单击 Add Files，然后选择一个要上传的测试文件。在本教程中，我们假设您上传一个名为 `myImage.jpg` 的图像。
3. 选择测试图像后，单击 Start Upload。

(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 属性变为 “真”。

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）。

创建 Amazon S3 客户端

要使用 Amazon S3，我们首先需要创建一个 Amazons3Client 实例，该实例引用你之前创建的 Cognit AWSCredentials o 实例：

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

`AmazonS3Client` 类是高级 S3 API 的入口点。

列出存储桶

要列出 AWS 账户中的存储桶，请调用 `AmazonS3Client.ListBucketsAsync` 方法，如以下示例代码中所示：

```
// ResultText is a label used for displaying status information
ResultText.text = "Fetching all the Buckets";
 Client.ListBucketsAsync(new ListBucketsRequest(), (responseObject) =>
 {
     ResultText.text += "\n";
     if (responseObject.Exception == null)
     {
         ResultText.text += "Got Response \nPrinting now \n";
         responseObject.Response.Buckets.ForEach((s3b) =>
         {
             ResultText.text += string.Format("bucket = {0}, created date = {1} \n",
             s3b.BucketName, s3b.CreationDate);
         });
     }
     else
     {
         ResultText.text += "Got Exception \n";
     }
 });
```

列出对象

要列出一个存储桶中的所有对象，请调用 AmazonS3Client.ListObjectsAsync 方法，如以下示例代码中所示：

```
// ResultText is a label used for displaying status information
ResultText.text = "Fetching all the Objects from " + S3BucketName;

var request = new ListObjectsRequest()
{
    BucketName = S3BucketName
};

Client.ListObjectsAsync(request, (responseObject) =>
{
    ResultText.text += "\n";
    if (responseObject.Exception == null)
    {
        ResultText.text += "Got Response \nPrinting now \n";
        responseObject.Response.S3Objects.ForEach((o) =>
        {
            ResultText.text += string.Format("{0}\n", o.Key);
        });
    }
    else
    {
        ResultText.text += "Got Exception \n";
    }
});
```

下载对象

要下载对象，请创建一个 GetObjectRequest，指定存储桶名称和密钥，然后将该对象传递给客户端。GetObjectAsync:

```
private void GetObject()
{
    ResultText.text = string.Format("fetching {0} from bucket {1}",
    SampleFileName, S3BucketName);
    Client.GetObjectAsync(S3BucketName, SampleFileName, (responseObj) =>
    {
```

```
        string data = null;
        var response = responseObj.Response;
        if (response.ResponseStream != null)
        {
           using (StreamReader reader = new StreamReader(response.ResponseStream))
           {
              data = reader.ReadToEnd();
           }

           ResultText.text += "\n";
           ResultText.text += data;
        }
    });
}
```

GetObjectAsync 取一个实例 GetObjectRequest、一个回调和一个 AsyncOptions 实例。回调的类型必须为：`AmazonServiceCallback<GetObjectRequest, GetObjectResponse>`。该 AsyncOptions 实例是可选的。如果指定，它将决定是否将在主线程上运行回调。

上传对象

要上传对象，请将您的对象写入流，创建一个新的对象 PostObjectRequest 并指定密钥、存储桶名称和流数据。

适用于 Unity 的 AWS SDK 使用不支持 HTTP PUT 操作的 WWW HTTP 客户端。要想将对象上传到您的 S3 存储桶，需要使用 S3 的浏览器 Post，如下所示。

```
public void PostObject(string fileName)
{
    ResultText.text = "Retrieving the file";

    var stream = new FileStream(Application.persistentDataPath +
    Path.DirectorySeparatorChar + fileName,
    FileMode.Open, FileAccess.Read, FileShare.Read);

    ResultText.text += "\nCreating request object";
    var request = new PostObjectRequest()
    {
        Bucket = S3BucketName,
        Key = fileName,
        InputStream = stream,
        CannedACL = S3CannedACL.Private
```

```
    };

    ResultText.text += "\nMaking HTTP post call";

    Client.PostObjectAsync(request, (responseObj) =>
    {
        if (responseObj.Exception == null)
        {
            ResultText.text += string.Format("\nobject {0} posted to bucket {1}",
            responseObj.Request.Key, responseObj.Request.Bucket);
        }
        else
        {
            ResultText.text += "\nException while posting the result object";
            ResultText.text += string.Format("\n receieved error {0}",
            responseObj.Response.HttpStatusCode.ToString());
        }
    });
}
```

Amazon DynamoDB

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。有关 DynamoDB 的信息，请参阅 Amazon DynamoDB。

适用于 Unity 的 AWS Mobile SDK 为您使用 DynamoDB 提供了高级库。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。

> **Note**
>
> 本文档中的一些示例假设使用名为的文本框变量 ResultText 来显示跟踪输出。

集成 Amazon DynamoDB

要在 Unity 应用程序中使用 DynamoDB，需要将 Unity SDK 添加到您的项目中。如果您尚未添加，则下载适用于 Unity 的 SDK，并按照设置适用于 Unity 的 AWS Mobile SDK 中的说明操作。我们建议您使用 Amazon Cognito Identity 为您的应用程序提供临时的 AWS 凭证。这些凭证允许您的应用程序访问 AWS 服务和资源。

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

此策略应当应用到分配给 Cognito 身份池的角色，但您需要用您的 DynamoDB 表的正确 ARN 来替换 **`Resource`** 值。Cognito 自动为您的新身份池创建一个角色，您可以在 IAM 控制台中将策略应用于此角色。

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，请参阅使用 IAM 控制对 DynamoDB 资源的访问。

创建 DynamoDB 表

现在，我们已设置了权限和凭证，接下来，我们为应用程序创建一个 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 `Bookstore` 作为表名。
3. 选择 Hash 作为主键类型。
4. 选择数字并输入哈希属性名称的 `id`。单击继续。
5. 再次单击 Continue 以跳过添加索引步骤。
6. 将读取容量设置为 `10`，将写入容量设置为 `5`。单击继续。
7. 输入通知电子邮件，然后单击 Continue 来创建吞吐量警报。
8. 单击创建。DynamoDB 将创建您的数据库。

创建 DynamoDB 客户端

我们需要一个客户端来让应用程序与 DynamoDB 表交互。可以创建一个默认的 DynamodDB 客户端，如下所示：

```
var credentials = new CognitoAWSCredentials(IDENTITY_POOL_ID, RegionEndpoint.USEast1);
AmazonDynamoDBClient client = new AmazonDynamoDBClient(credentials);
DynamoDBContext Context = new DynamoDBContext(client);
```

该 AmazonDynamoDBClient 类是 DynamoDB API 的入口点。该类提供实例方法，用于创建、描述、更新和删除表以及其他操作。上下文在客户端之上又增加了一个抽象层，使您能够使用对象持久化模型等其他功能。

描述表

要获取 DynamoDB 表的描述，可以使用以下代码：

```
resultText.text +=("\n*** Retrieving table information ***\n");
       var request = new DescribeTableRequest
       {
           TableName = @"ProductCatalog"
       };
       Client.DescribeTableAsync(request, (result) =>
       {
               if (result.Exception != null)
               {
                       resultText.text += result.Exception.Message;
                       Debug.Log(result.Exception);
                       return;
               }
               var response = result.Response;
               TableDescription description = response.Table;
               resultText.text += ("Name: " + description.TableName + "\n");
               resultText.text += ("# of items: " + description.ItemCount + "\n");
               resultText.text += ("Provision Throughput (reads/sec): " +
                   description.ProvisionedThroughput.ReadCapacityUnits + "\n");
               resultText.text += ("Provision Throughput (reads/sec): " +
                   description.ProvisionedThroughput.WriteCapacityUnits + "\n");

       }, null);
   }
```

在此示例中，我们创建了一个客户端和一个 DescribeTableRequest 对象，将表的名称分配给该**`TableName`**属性，然后将请求对象传递给该 AmazonDynamoDBClient 对象上的 DescribeTableAsync 方法。 DescribeTableAsync 还需要一个委托，该委托将在异步操作完成时被调用。

> **Note**
>
> 异步操作完成时调 AmazonDynamoDBClient 用的 take 委托上的所有异步方法。

保存对象

要将对象保存到 DynamoDB，请使用 SaveAsync<T> AmazonDynamoDBClient 该对象的方法，其中 T 是您要保存的对象的类型。

我们将我们的数据库称为“书店”，围绕这个主题，我们将实施一个数据模型，用于记录图书相关的属性。以下是定义我们的数据模型的类。

```
[DynamoDBTable("ProductCatalog")]
    public class Book
    {
        [DynamoDBHashKey]   // Hash key.
        public int Id { get; set; }
        [DynamoDBProperty]
        public string Title { get; set; }
        [DynamoDBProperty]
        public string ISBN { get; set; }
        [DynamoDBProperty("Authors")]    // Multi-valued (set type) attribute.
        public List<string> BookAuthors { get; set; }
    }
```

当然，就真正的书店应用程序而言，一些事项（如作者和价格等）需要附加字段。Book 类采用 [DynamoDBTable] 属性装饰，该属性定义了要写入 Book 类型的数据库表对象。Book 类的每个实例的密钥都是使用 [Dynamo DBHash Key] 属性标识的。属性用 [DynamoDBProperty] 属性标识，这些属性指定数据库表中要写入该属性的列。借助现成模型，我们可以编写一些方法来创建、检索、更新和删除 Book 对象。

创建书籍

```
private void PerformCreateOperation()
{
    Book myBook = new Book
    {
        Id = bookID,
        Title = "object persistence-AWS SDK for.NET SDK-Book 1001",
        ISBN = "111-1111111001",
        BookAuthors = new List<string> { "Author 1", "Author 2" },
    };

    // Save the book.
```

```
    Context.SaveAsync(myBook,(result)=>{
        if(result.Exception == null)
            resultText.text += @"book saved";
    });
}
```

检索书籍

```
private void RetrieveBook()
{
    this.displayMessage += "\n*** Load book**\n";
    Context.LoadAsync<Book>(bookID,
                            (AmazonDynamoResult<Book> result) =>
    {
        if (result.Exception != null)
        {

            this.displayMessage += ("LoadAsync error" +result.Exception.Message);
            Debug.LogException(result.Exception);
            return;
        }
        _retrievedBook = result.Response;
        this.displayMessage += ("Retrieved Book: " +
                                "\nId=" + _retrievedBook.Id +
                                "\nTitle=" + _retrievedBook.Title +
                                "\nISBN=" + _retrievedBook.ISBN);
        string authors = "";
        foreach(string author in _retrievedBook.BookAuthors)
            authors += author + ",";
        this.displayMessage += "\nBookAuthor= "+ authors;
        this.displayMessage += ("\nDimensions= "+ _retrievedBook.Dimensions.Length + "
 X " +
                                _retrievedBook.Dimensions.Height + " X " +
                                _retrievedBook.Dimensions.Thickness);

    }, null);
}
```

更新书籍

```
private void PerformUpdateOperation()
```

```
{
    // Retrieve the book.
    Book bookRetrieved = null;
    Context.LoadAsync<Book>(bookID,(result)=>
    {
        if(result.Exception == null )
        {
            bookRetrieved = result.Result as Book;
            // Update few properties.
            bookRetrieved.ISBN = "222-2222221001";
            // Replace existing authors list with this
            bookRetrieved.BookAuthors = new List<string> { "Author 1", "Author x" };
            Context.SaveAsync<Book>(bookRetrieved,(res)=>
            {
                if(res.Exception == null)
                     resultText.text += ("\nBook updated");
            });
        }
   });
}
```

删除书籍

```
private void PerformDeleteOperation()
{
    // Delete the book.
    Context.DeleteAsync<Book>(bookID,(res)=>
    {
        if(res.Exception == null)
        {
            Context.LoadAsync<Book>(bookID,(result)=>
            {
                 Book deletedBook = result.Result;
                 if(deletedBook==null)
                     resultText.text += ("\nBook is deleted");
            });
        }
   });
}
```

Amazon Simple Notification Service

使用 Amazon Simple Notification Service (SNS) 和 Unity SDK，您可以编写接收移动推送通知的 iOS 和 Android 应用程序。有关 SNS 的信息，请参阅 Amazon Simple Notification Service。

本主题将引导您配置适用于 Unity 的 AWS 开发工具包示例应用程序 SNSExample .unity，使其通过亚马逊 SNS 接收移动推送通知。

你可以使用 SNSExample .unity 示例创建 iOS 和安卓应用程序。iOS 和 Android 的配置步骤不同，请阅读下面对应于您的目标平台的部分。

先决条件

需要满足以下先决条件。

设置 SNS 权限

当您创建一个 Cognito 身份池时，会生成两个 IAM 角色：

- Cognito/_ <Identity-Pool-Name>Auth_ DefaultRole -经过身份验证的用户的默认 IAM 角色
- Cognito/_ <Identity-Pool-Name>Unauth_ DefaultRole -未经身份验证的用户的默认 IAM 角色

必须为这些角色添加访问 Amazon SNS 服务的权限。要实现此目的，应按照以下步骤进行：

1. 浏览到 IAM 控制台并选择要配置的 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”。

> **Note**
>
> 不建议在生产环境中使用 Amazon A SNSFull ccess，我们在这里使用它来帮助您快速启动并运行。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。

iOS 先决条件

- Apple iOS 开发人员计划成员资格

- 生成签名身份
- 创建为推送通知而配置的预置配置文件

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。接下来，您需要一个为推送通知而配置的预置配置文件。有关更多信息，请参阅 Apple 的 Configuring Push Notifications 文档。

Android 先决条件

- 安装 Android SDK
- 安装 JDK
- android-support-v4.jar
- google-play-services.jar

配置针对 iOS 的 Unity 示例应用程序

打开 Unity 编辑器并创建一个新项目。通过选择资产 /导入包/ 自定义包并选择 aws-unity-sdk-sns-2.0.0.1.unitypackage 来导入适用于 Unity 的 AWS 开发工具包包。确保选中 Importing Package 对话框中的所有项目，然后单击 Import。

Unity 配置

执行以下步骤来配置 Unity 项目：

1. 在 “项目” 窗格中，导航到 “资源/AWSSDK/示例”，然后打开 SNSExample 场景。
2. 在 “层次结构” 窗格中，选择 SNSExample。
3. 在 Inspector 窗格中，指定您的 Cognito 身份池 ID。
4. 请注意，有一个标签为 iOS Platform Application ARN 的文本框，稍后您将生成该信息。
5. 选择 File/Build Settings，在 Build Settings 对话框中，单击 Scenes in Build 列表框下的 Add Current 按钮将当前场景添加到该生成中。
6. 在 Platform (平台) 下，选择 iOS，单击 Player Settings... (播放器设置...) 按钮，在 Unity 编辑器的 Inspector Pane (检查器窗格) 中，单击 iPhone 图标并向下滚动到 Identification (身份证明) 部分，并指定一个 Bundle Identifier (服务包标识符)。

iOS 配置

执行以下步骤配置示例来配置 iOS 特定设置：

1. 在 Web 浏览器中，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“Continue”按钮。
6. 单击 Submit 按钮。
7. 单击 Done 按钮。
8. 选择您刚刚创建的应用程序 ID，然后单击 Edit 按钮。
9. 向下滚动到 Push Notifications 部分。
10. 单击 Development SSL Certificate 下的 Create Certificate 按钮。
11. 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。
12. 回到 Certificates, Identifiers & Profiles Web 页面，单击 Provisioning Profiles 下的 All。
13. 单击右上角的 + 按钮以添加新的预置配置文件。
14. 选择 iOS App Development，然后单击 Continue 按钮。
15. 选择您的应用程序 ID，然后单击 Continue 按钮。
16. 选择您的开发人员证书，然后单击 Continue 按钮。
17. 选择您的设备，然后单击 Continue 按钮。
18. 输入配置文件名称，然后单击 Generate 按钮。
19. 下载预置文件后双击以安装预置配置文件。

添加新预置配置文件后，您可能需要在 Xcode 中刷新“预置配置文件”。在 Xcode 中：

1. 选择 Xcode/Preferences 菜单项。
2. 选择 Accounts 选项卡，选择您的 Apple ID，单击 View Details。

3. 单击对话框左下角的刷新按钮，以刷新您的预置配置文件并确保显示新配置文件。

SNS 配置

1. 运行 KeyChain 访问应用程序，选择屏幕左下角的我的证书，右键单击您为连接到 APNS 而生成的 SSL 证书，然后选择导出，系统将提示您指定文件名称和保护证书的密码。证书将保存在 P12 文件中。
2. 在 Web 浏览器中，转至 SNS Console，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 Apple Push Notification Service Sandbox (APNS_SANDBOX)。
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。
10. 在 Unity 编辑器中返回您的项目，SNSExample在 “层次结构” 窗格的 “Ins pector” 窗格中进行选择，然后将平台应用程序 ARN 粘贴到标有 iO S 平台应用程序 AR N 的文本框中。
11. 选择 File/Build Settings，单击 Build 按钮，这将创建一个 Xcode 项目。

使用 Xcode

1. 打开 Xcode 项目，在“Project Navigator”中选择项目。
2. 验证服务包标识符是否设置正确。
3. 确保在 Team 中指定了 Apple 开发人员账户 – 这是使您的预置配置文件生效所必需的。
4. 生成项目并在设备上运行。
5. 单击 Register for Notification，单击 OK 以允许通知，应用程序将显示您的设备令牌。

在 SNS Console 中，单击 Applications，选择您的平台应用程序，单击 Create Platform Endpoint，然后输入应用程序显示的设备令牌。

此时，您的应用程序 APNS 和 NSN 完全配置完毕。您可以选择您的平台应用程序，选择终端节点，然后单击 Publish to endpoint 以便将推送通知发送到您的设备。

Unity 示例 (iOS)

该示例创建了一个 Cognito AWSCredentials 实例来生成允许应用程序调用 AWS 服务的临时有限范围证书。它还会创建用于与 SNS 通信的 AmazonSimpleNotificationServiceClient 实例。该应用程序显示两个按钮，其标签分别为 Register for Notification 和 Unregister。

点击 Register for Notifications (注册通知) 按钮时将调用 `RegisterDevice()` 方法。`RegisterDevice()` 调用 `UnityEngine.iOS.NotificationServices.RegisterForNotifications`，以指定使用哪些通知类型（警报、声音或徽章）。它还对 APNS 进行异步调用以获取设备令牌。因为没有定义回调，所以 `CheckForDeviceToken` 被重复调用 (多达 10 次) 以检查设备令牌。

当检索到令牌时，调用 `AmazonSimpleNotificationServiceClient.CreatePlatformEndpointAsync()` 来为 SNS 平台应用程序创建终端节点。

此示例现在配置为接收推送通知。您可以浏览到 SNS Console，单击页面左侧的 Applications，选择平台应用程序，选择终端节点，然后单击 Publish to endpoint。选择要使用的终端节点，然后单击 Publish to Endpoint。在文本框中键入文本消息，然后单击 Publish message 发布消息。

配置针对 Android 的 Unity 示例应用程序

打开 Unity 编辑器并创建一个新项目。通过选择资产 /导入包/ 自定义包并选择 aws-unity-sdk-sns-2.0.0.1.unitypackage 来导入适用于 Unity 的 AWS 开发工具包包。确保选中 Importing Package 对话框中的所有项目，然后单击 Import。

Unity 配置

执行以下步骤来配置 Unity 项目：

1. 在 “项目” 窗格中，导航到 “资源/AWSSDK/示例”，然后打开 SNSExample 场景。
2. 在 “层次结构” 窗格中，选择 SNSExample。
3. 在 Inspector 窗格中，指定您的 Cognito 身份池 ID。
4. 请注意，有标签为 Android Platform Application ARN 和 Google Console Project ID 的文本框，稍后您将生成这些信息。
5. 选择 File/Build Settings，在 Build Settings 对话框中，单击 Scenes in Build 列表框下的 Add Current 按钮将当前场景添加到该生成中。

6. 在 Platform (平台) 下，选择 Android，单击 Player Settings... (播放器设置...) 按钮，在 Unity 编辑器的 Inspector Pane (检查器窗格) 中，单击 Android 图标并向下滚动到 Identification (身份证明) 部分，并指定一个 Bundle Identifier (服务包标识符)。
7. 将 android-support-v 4.jar 和 google-play-services .jar 复制到 “项目” 窗格的 Ass et s/ P lugins/ Android 目录中。

有关在哪里可以找到 android-support-v 4.jar 的更多信息，请参阅 Android Support 库设置。有关如何查找 google-play-services .jar 的更多信息，请参阅谷歌 APIs 安卓版安装程序。

Android 配置

首先，添加一个新的 Google API 项目：

1. 在 Web 浏览器中，转至 Google Developers Console，单击 Create Project。
2. 在 New Project 框中，输入项目名称，记下项目编号 (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，您的新项目应当已经被选中，如果没有，则在页面顶部的下拉菜单中选中。
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”，单击下面的 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。

SNS 配置

1. 在 Web 浏览器中，转至 SNS Console，单击屏幕左侧的 Applications。
2. 单击 Create platform application，以创建新的 SNS 平台应用程序。
3. 输入 Application Name。
4. 对于 Push notification platform，选择 Google Cloud Messaging (GCM)。
5. 将 API 密钥粘贴到标记为 API key 的文本框中。
6. 单击 Create platform application。
7. 选择您刚创建的平台应用程序，然后复制应用程序 ARN。
8. 在 Unity 编辑器中返回您的项目，SNSExample在 “层次结构” 窗格的 “检查器” 窗格中进行选择，然后将平台应用程序 ARN 粘贴到标有 Android 平台应用程序 ARN 的文本框中，将您的项目编号粘贴到标有 Google 控制台项目 ID 的文本框中。
9. 将您的 Android 设备连接到计算机，选择 File/Build Settings，然后单击 Build and Run。

Unity 示例 (Android)

该示例创建了一个 Cognito AWSCredentials 实例来生成允许应用程序调用 AWS 服务的临时有限范围证书。它还会创建用于与 SNS 通信的 AmazonSimpleNotificationServiceClient 实例。

该应用程序显示两个按钮，其标签分别为 Register for Notification 和 Unregister。点击 Register for Notifications (注册通知) 按钮时将调用 `RegisterDevice()` 方法。`RegisterDevice()` 调用 `GCM.Register`，这将向 GCM 注册应用程序。GCM 是在示例代码范围内定义的一个类。它执行异步调用来向 GCM 注册应用程序。

当调用回调时，调用 `AmazonSimpleNotificationServiceClient.CreatePlatformEndpointAsync` 来创建平台终端节点以接收 SNS 消息。

此示例现在配置为接收推送通知。您可以浏览到 SNS Console，单击页面左侧的 Applications，选择平台应用程序，选择终端节点，然后单击 Publish to endpoint。选择要使用的终端节点，然后单击 Publish to Endpoint。在文本框中键入文本消息，然后单击 Publish message 发布消息。

AWS Lambda

AWS Lambda 是一个计算服务，它运行您的代码来响应请求或事件，并且自动为您管理计算资源，使构建快速响应新信息的应用程序变得容易。AWS Lambda 函数可以直接从移动设备、物联网和 Web 应用调用，并且可以同步发回响应，因此无需预配置或管理基础设施，就可方便地为移动应用程序创建可扩展、安全且高度可用的后端。

AWS Lambda 可以执行您的 Lambda 函数来响应以下任务之一：

- 事件，例如离散更新（例如，Amazon S3 中的对象创建事件或 CloudWatch 警报），或流式更新（例如，网站点击流或来自联网设备的输出）。
- 来自您的自定义应用程序的 JSON 输入或 HTTPS 命令。

只有在需要时 AWS Lambda 才执行您的代码，并且能自动扩展，从每天几个请求扩展到每秒数千个请求。借助这些功能，您可以使用 Lambda 来轻松地为 AWS 服务（如 Amazon S3 和 Amazon DynamoDB）构建触发程序，处理存储在 Amazon Kinesis 中的流数据，或创建您自己的按 AWS 规模、性能和安全性运行的后端。

要了解有关 AWS Lambda 工作原理的更多信息，请参阅 AWS Lambda：工作原理。

Permissions

有两种权限与 Lambda 函数有关：

- 执行权限 – 您的 Lambda 函数访问您账户中的其他 AWS 资源所需的权限。您通过创建 IAM 角色（称作执行角色）来授予此类权限。
- 调用权限 – 事件源与您的 Lambda 函数通信所需的权限。根据调用模型 (推模型或拉模型)，您可以使用执行角色或资源策略 (与 Lambda 函数关联的访问策略) 来授予这些权限。

项目设置

设置 AWS Lambda 权限

1. 打开 AWS IAM 控制台。
2. 将这种自定义策略附加到您的角色，让您的应用程序能够调用 AWS Lambda。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:*"
      ],
      "Resource": "*"
    }
  ]
}
```

创建新的执行角色

该角色适用于您将在下一步中创建的 Lambda 函数，另外它还决定该函数可以访问哪些 AWS 资源。

1. 打开 AWS IAM 控制台。
2. 单击 Roles。
3. 单击 Create New Roles。
4. 按照屏幕上的说明选择您的 Lambda 函数需要访问的服务和相应策略。例如，如果您想让您的 Lambda 函数创建 S3 存储桶，则您的策略需要 S3 写入权。
5. 单击创建角色。

在 AWS Lambda 中创建函数

1. 打开 AWS Lambda 控制台。
2. 单击 Create a Lambda function。
3. 单击 Skip 以跳过创建蓝图。
4. 在下一屏幕上配置您自己的函数。输入函数名称、描述并选择您的运行时。基于您选择的运行时，按照屏幕上的说明操作。通过将新创建的执行角色分配给您的函数来指定执行权限。
5. 完成后，单击 Next。
6. 单击 Create Function。

创建 Lambda 客户端

```
var credentials = new CognitoAWSCredentials(IDENTITY_POOL_ID, RegionEndpoint.USEast1);
var Client = new AmazonLambdaClient(credentials, RegionEndpoint.USEast1);
```

创建请求对象

创建请求对象来指定调用类型和函数名称：

```
var request = new InvokeRequest()
{
    FunctionName = "hello-world",
    Payload = "{\"key1\" : \"Hello World!\"}",
    InvocationType = InvocationType.RequestResponse
};
```

调用 Lambda 函数

调用 invoke，传递请求对象：

```
Client.InvokeAsync(request, (result) =>
{
    if (result.Exception == null)
    {
        Debug.Log(Encoding.ASCII.GetString(result.Response.Payload.ToArray()));
    }
    else
    {
        Debug.LogError(result.Exception);
    }
});
```

故障排除

由于适用于 Unity 的 AWS SDK 所用的 Unity.WWW 类存在限制，因此，在调用 AWS 服务时，如果发生问题，系统不会返回详细的错误消息。本主题将介绍一些排除此类故障的思路。

确保 IAM 角色具有所需权限

在调用 AWS 服务时，您的应用程序会使用来自 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，用于指定分配给此角色的用户可以访问哪些 AWS 资源。默认情况下，系统会创建两个角色：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

如果应用程序要调用的 AWS 服务具有 HTTP 或 HTTPS 端点，则您可以使用 HTTP/HTTPS 代理调试程序来查看请求和响应，以便更深入地了解所发生的活动。我们提供有多种 HTTP 代理调试程序，例如：

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

> **Important**
>
> 在运行 Charles Web 调试代理时，Cognito 凭证提供程序会出现导致其无法正常运行的已知问题。

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（位于中 AWSUnitySDK/AWSCore/Resources），然后将需要调试的 AWS 服务的 HTTP 标签设置为 true