Xamarin 開發人員指南

AWS Mobile SDK

AWS Mobile SDK: Xamarin 開發人員指南

Table of Contents

..... viii
適用於 .NET 和 Xamarin 的 AWS Mobile 開發套件是什麼？ 1
相關指南和主題 1
存檔的參考內容 1
適用於 .NET 的和 Xamarin 的 AWS Mobile 開發套件中包含哪些內含內容？ 1
相容性 2
如何取得適用於 .NET 的和 Xamarin 的 AWS Mobile 開發套件？ 2
關於 AWS 行動服務 3
設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 5
先決條件 5
步驟 1：取得 AWS 憑證 5
步驟 2：設定許可 6
步驟 3：建立新的 專案 7
Windows 7
OS X 7
步驟 4：安裝適用於 .NET 和 Xamarin 的 AWS Mobile SDK 8
Windows 8
Mac 9
步驟 5：配置適用於 .NET 和 Xamarin 的 AWS Mobile SDK 9
設定記錄 9
設置區域終端節點 10
配置 HTTP 代理服務器設置 10
糾正時鐘扭曲 10
後續步驟 11
適用於 .NET 和 Xamarin 的 AWS Mobile SDK 12
使用 Amazon S3 存儲和檢索文件 12
項目設定 12
初始化 S3TransferUtility用戶端 14
將檔案上傳到 Amazon S3 14
從 Amazon S3 下載檔案 15
通過認知同步同步用户數據 15
項目設定 12
初始化CognitoSync經理 16
同步使用者資料 16

使用 DynamoDB 存儲和檢索數據 17
項目設定 12
InitializeAmazonDynamoDB客户端 19
建立類別 19
儲存項目 20
擷取項目 20
更新項目 20
刪除項目 21
用 Amazon Mobile Analytics 追蹤應用程式用量資料 21
項目設定 12
InitializeMobileAnalytics經理 22
跟蹤工作階段事件 23
使用 SNS 接收推送通知 23
項目設定 12
建立 SNS 客户端 26
註冊應用程序以獲得遠程通知 26
從 SNS 控制台向終端節點發送消息 26
使用 SNS 接收推送通知 27
專案設定 12
建立 SNS 客户端 26
註冊應用程序以獲得遠程通知 26
從 SNS 控制台向終端節點發送消息 26
Amazon Cognito Identity 33
什麼是 Amazon Cognito 身分？ 33
使用公共提供程序對用户進行身份驗證 33
使用開發人員驗證的身分 33
Amazon Cognito Sync 34
什麼是 Amazon Cognito Sync？ 34
Amazon Mobile Analytics 35
重要概念 35
報告類型 35
項目設定 12
先決條件 5
配置 Mobile Analytics 設定 22
將 Mobile Analytics 與您的應用程式整合 37
在 Mobile Analytics 控制台中創建應用 21

建立MobileAnalytics管理員客户端 37
記錄獲利事件 37
記錄自訂事件 38
記錄工作階段 38
Amazon Simple Storage Service (S3) 40
什麼是 S3？ 40
重要概念 35
儲存貯體 40
物件 40
物件中繼資料 41
項目設定 12
先決條件 5
建立 S3 儲存貯體 41
設定 S3 的許可 13
（可選）配置 S3 請求的簽名版本 14
將 S3 與您的應用程式整合 43
使用 S3 Transfer File 43
初始化TransferUtility 43
（可選）配置TransferUtility 43
下載檔案 44
上傳檔案 44
使用服務級別 S3 API 44
初始化 Amazon S3 用户端 45
下載檔案 44
上傳檔案 44
刪除項目 21
刪除多個物件 46
列出儲存貯體 47
列出物件 48
獲取儲存貯體區域 48
獲取儲存貯體策略 49
Amazon DynamoDB 50
什麼是 Amazon DynamoDB？ 50
重要概念 35
資料表 50
項目和屬性 50

資料類型 50
主索引鍵 51
次要索引 51
查詢和掃描 51
項目設定 12
先決條件 5
建立 DynamoDB 資料表 17
設定 DynamoDB 的許可 18
將 DynamoDB 與您的應用程序集成 54
使用文件模型 54
建立 DynamoDB 用户端 55
CRUD 操作 55
使用物件持久性模型 57
概觀 58
支援的資料類型 58
建立 DynamoDB 用户端 55
CRUD 操作 55
查詢和掃描 51
使用 DynamoDB 服務級別 API 62
建立 DynamoDB 用户端 55
CRUD 操作 55
查詢及掃描 51
Amazon Simple Notification Service (SNS) 67
重要概念 35
主題 67
訂閱 67
發佈 67
項目設定 12
先決條件 5
整合 SNS 與您的應用程式 68
發送推送通知 (Xamarin 安卓) 68
專案設定 12
建立 SNS 客户端 26
註冊應用程序以獲得遠程通知 26
從 SNS 控制台向終端節點發送消息 26
發送推送通知 (Xamarin iOS) 73

項目設定 12
建立 SNS 客户端 26
註冊應用程序以獲得遠程通知 26
從 SNS 控制台向終端節點發送消息 26
發送和接收短信通知 76
建立主題 77
使用 SMS 協議訂主題 77
發佈訊息 78
傳送訊息至 HTTP/HTTPS 端點 79
配置您的 HTTP/HTTPS 終端節點以接收 Amazon SNS 消息 79
訂 HTTP/HTTPS 端點至您的 Amazon SNS 主題 79
確認您的 訂閱 80
傳送訊息至 HTTP/HTTPS 端點 80
SNS 故障診斷 80
在 Amazon SNS 控制台中使用配送狀態 80
使用適用 .NET 和 Xamarin 的 AWS Mobile SDK 的最佳實務 81
AWS 服務文檔庫 81
Amazon Cognito 身分 33
Amazon Cognito Sync 3
Amazon Mobile Analytics 35
Simple Storage Service (Amazon S3) 82
Amazon DynamoDB 82
Amazon Side Notification Service (SNS) 82
其他有用鏈接 82
故障診斷 83
確保 IAM 角色具有所需的權限 83
使用 HTTP 代理調試器 84
文件歷史記錄 85

Xamarin 的AWS行動 SDK 現在已包含在AWS SDK for .NET. 本指南參考 Xamarin 行動 SDK 的封存版本。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。

適用於 .NET 和 Xamarin 的 AWS Mobile 開發套件是什麼？

Xamarin 的AWS行動 SDK 包含在AWS SDK for .NET. 如需詳細資訊，請參閱 AWS SDK for .NET 開發人員指南。

本指南不再更新 — 它會參照 Xamarin 行動 SDK 的封存版本。

相關指南和主題

- 對於前端和移動應用程序開發，我們建議使用 AWS Amplify.
- 如需使用 Xamarin 應用程式的AWS SDK for .NET特殊考量事項，請參閱AWS SDK for .NET開發人員指南中的 Xamarin 支援的特殊注意事項。
- 基於參考目的，您可以在上找到 XamarinAWS 行動 SDK 的封存版本 GitHub。

存檔的參考內容

適用於 .NET 的 AWS Mobile SDK for Mobile 開發套件的適用於 .NET 的 AWS Mobile SDK 內含程式庫、程式碼範本和文件，可協助開發人員針對下列項目建立連線行動應用程式：

- XamariOS
- 適用於 Android 的
- 視窗手機銀光
- Windows 8.1
- Windows 手機

使用適用於 .NET 和 Xamarin 的 AWS 行動開發套件撰寫的行動應用程式會呼叫原生平台 API，讓它們具有原生應用程式的外觀和感覺。開發套件中的 .NET 程式庫提供圍繞 AWS REST API 的 C# 包裝函式。

適用於 .NET 的和 Xamarin 的 AWS Mobile 開發套件中包含哪些內含內容？

目前支援 AWS 服務包括但不限於：

- Amazon Cognito
- Amazon Simple Storage Service (Amazon S3)

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

這些服務可讓您驗證使用者、儲存玩家和遊戲資料、將物件儲存在雲端、接收推播通知，以及收集和分析使用情況資料。

適用於 .NET 和 Xamarin 的 AWS 行動開發套件也可讓您使用適用於 .NET 的 AWS 開發套件所支援的大部分 AWS 服務。本開發人員指南將說明行動開發專用的 AWS 服務。如需適用於 .NET 的 AWS 開發套件的詳細資訊，請參閱：

- AWS SDK for .NET 開發套件
- 適用於 .NET 開發人員的 AWS 開發套件
- AWS SDK for .NET

相容性

適用於 .NET 和 Xamarin 的 AWS 行動開發套件是以可攜式類別程式庫 (PCL) 形式提供。PCL Support 被添加在夏馬安卓 4.10.1 和 Xamar.iOS 7.0.4。可移植庫項目內置到視覺工作室。

IDE

如需將 IDE 與 Xamarin SDK 的封存版本搭配使用的詳細資訊，請參閱設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK。

如何取得適用於 .NET 的和 Xamarin 的 AWS Mobile 開發套件？

若要取得適用於 .NET 和 Xamarin 的 AWS 行動開發套件，請參閱設定適用於 .NET 和 Xamarin 的 AWS 行動開發套件。適用於 .NET 和 Xamarin 的 AWS Mobile 開發套件會以 NuGet 套件的形式散發。您可以在適用於 .NET GitHub存放庫的 AWS 開發套件 NuGet或 AWS 開發套件中找到 AWS 服務套件的完整清單。

關於 AWS 行動服務

Amazon Cognito 身分

對 AWS 進行的所有呼叫都需要 AWS 登入資料。我們建議您使用 Amazon Cognito 身分為應用程式提供 AWS 登入資料，而不是將登入資料硬式編碼到您的應用程式中。遵循設定適用於 .NET 和 Xamarin 的 AWS 行動開發套件中的指示，透過 Amazon Cognito 取得 AWS 登入資料。

Cognito 還允許您使用公共登錄提供商（如亞馬遜，臉譜，推特和谷歌）以及支持 OpenID Connect 的提供商來身份驗證用戶。Cognito 也適用於未經驗證的使用者。Cognito 提供具有有限存取權限的臨時登入資料，您可以透過 I dentity and Access Management (IAM) 角色指定。Cognito 可透過建立身分集區來設定 Cognito，可設定 Cognito。IAM 角色會指定您的應用程式可存取的資源/服務。

若要開始使 Cognito amarin 身分，請參閱設定 AWS Mobile 開發套件。

若要進一步了解 Cognito 身分識別，請參閱 Amazon Cognito 可身分。

Amazon Cognito Sync

Cognito Sync 是 AWS 服務和用戶端程式庫，可讓您跨裝置同步應用程式相關的使用者資料。您可以使用 Cognito Sync API，跨裝置和登入供應商 (亞馬遜、Facebook、Google 和您自己的自訂身分供應商) 同步使用者設定檔資料。

若要開始使用 Cognito 同步，請參閱使用 Cognito 同步來同步使用者資料。

如需有關 Cognito 同步的詳細資訊，請參閱 Amazon Cognito 同步。

Mobile Analytics

Amazon Mobile Analytics 可讓您收集、視覺化並瞭解行動應用程式的應用程式使用情況。報告可用於作用中使用者、工作階段、保留率、應用程式內收益和自訂事件的指標，並且可依平台和日期範圍進行篩選。Amazon Mobile Analytics 專為隨著您的業務擴展而建置，可以從數百萬個端點收集和處理數十億個事件。

若要開始使用 Mobile Analytics，請參閱使用 Amazon Mobile Analytics 追蹤應用程式使用情況資料。

如需有關 Mobile Analytics 的詳細資訊，請參閱 Amazon Mobile Analytics。

Dynamo DB

Amazon DynamoDB 是一種快速、可輕鬆擴展、高度可用、經濟實惠、非關聯式資料庫服務。DynamoDB 消除了傳統的資料儲存可擴展性限制，同時保持低延遲和可預測的效能。

若要開始使用發電機資料庫，請參閱使用 DynamoDB 儲存和擷取資料。

如需有關迪納摩資料庫的詳細資訊，請參閱 Amazon DynamoDB DB。

Amazon Simple Notification Service

Amazon Simple Notification Service (SNS) 是快速、彈性、全受管推播通知服務，可讓您傳送個別訊息或散發訊息給大量收件人。Amazon 簡單通知服務可讓您輕鬆且經濟實惠地傳送推播通知給行動裝置使用者、電子郵件收件者，甚至將訊息傳送到其他分散式服務。

若要開始使用適用於 Xamarin iOS 的 SNS，請參閱使用 SNS 接收推送通知 (iOS)。

若要開始使用 SNS 的 Xamarin 安卓系統，請參閱使用 SNS 接收推送通知 (安卓系統)。

如需 SNS 的詳細資訊，請參閱 Amazon Simple Notification Service (SNS)。

設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK

您可以設置適用於 .NET 和 Xamarin 的 AWS 移動開發工具包，並開始構建新項目，也可以將開發工具包與現有項目集成。您也可以複製和執行樣本來瞭解 SDK 的工作原理。請按照以下步驟設定並開始使用適用於 .NET 和 Xamarin 的 AWS Mobile SDK。

先決條件

在使用適用於 .NET 和 Xamarin 的 AWS Mobile SDK in 之前，您必須執行以下操作：

- 建立一個 AWS 帳户。
- 安裝Xamarin。

在您完成事前準備後：

1. 通過使用 Amazon Cognito 獲取 AWS 證書。
2. 為您將在應用程序中使用的每個 AWS 服務設置所需的權限。
3. 在 IDE 中建立新專案。
4. 安裝適用 .NET 和 Xamarin 的 AWS Mobile SDK
5. 配置適用於 .NET 和 Xamarin 的 AWS Mobile SDK

步驟 1：取得 AWS 憑證

要在應用程序中調用 AWS，您必須首先獲取 AWS 證書。您可以使用 Amazon Cognito（AWS 服務）來完成此操作，該服務允許您的應用程序訪問軟件開發工具包中的服務，而無需在應用程序中嵌入私有 AWS 證書。

要開始使用 Amazon Cognito，您需要創建一個身份池。身分集區是特定於您的帳户的信息存儲區，由如下所示的唯一身份集區 ID 進行標識。 ：

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

1. 登錄到Amazon Cognito 主控台，選擇管理聯合身分，然後選擇建立新的身分集區。
2. 輸入身分集區的名稱，然後選中該複選框以允許對未驗證身份的訪問。選擇建立集區以建立身分集區。

3. 選擇Allow以建立與身分集區相關聯的兩個默認角色，一個用於未經驗證的使用者，另一個用於已驗證的使用者。這兩個預設角色可讓您的身分集區存取 Amazon Cognito Sync (Amazon Cognito) 和 Amazon Mobile Analytics

通常，每個應用程序只能使用一個標識池。

在您建立身分集區後，您 AWS 以通過創建CognitoAWSCredentials對象（傳遞您的身份池 ID），然後將其傳遞給 AWS 客户端的構造函數，如下所示。 ：

```
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
);
```

步驟 2：設定許可

您需要設定要在應用程式中使用的每個 AWS 服務設定權限。首先，您需要瞭解 AWS 如何查看您的應用程序用户。

當有人使用您的應用程序並調用 AWS 時，AWS 會為該用户分配身份。您在步驟 1 中創建的身份池是 AWS 存儲這些身份的位置。有兩種類型的身分：已驗證和未驗證。已驗證的身份屬於由公有登入供應商 (例如 Facebook、Amazon、Google) 驗證的使用者。未驗證的身分屬於訪客用户。

每個身份都與AWS Identity and Access Management角色。在步驟 1 中，您已建立兩個 IAM 角色，一個用於已驗證的使用者，另一個用於未驗證的使用者。每個 IAM 角色都附加了一個或多個策略，用於指定分配給該角色的身份可以訪問哪些 AWS 服務。例如，以下示例政策允許 Amazon S3 存取儲存儲體。 ：

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

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

要設置要在應用程序中使用的 AWS 服務的權限，請修改附加到角色的策略。

1. 前往IAM 控制台，然後選擇角色。在搜尋方塊中輸入您的身分集區名稱。選擇您要設定的 IAM 角色。如果您的應用程序同時允許經過身份驗證的用户和未經身份驗證的用户，則需要為這兩個角色授予權限。
2. 按一下連接政策，選擇您希望下一步的政策，然後按一下連接政策。您創建的 IAM 角色的默認策略提供了對 Amazon Cognito 同步和 Mobile Analytics 的訪問權限。

有關創建政策或從現有政策列表中進行選擇的詳細資訊，請參IAM 政策。

步驟 3：建立新的 專案

Windows

您可以使用 Visual Studio 開發您的應用程式。

OS X

您可以使用 Visual Studio 開發您的應用程序。使用 Xamarin 的 iOS 開發需要訪問 Mac 才能運行您的應用程序。如需詳細資訊，請參閱「」在窗口上安裝。

> **Note**
>
> 跨平台商用 IDE騎士從JetBrains在 Windows 和 Mac 平台上都包括 Xamarin 支持。

步驟 4：安裝適用於 .NET 和 Xamarin 的 AWS Mobile SDK

Windows

選項 1：使用套件管理員主控台進行安裝

適用於 .NET 和 Xamarin 的 AWS 行動開發套件由一組 .NET 集團組成。要安裝適用於 .NET 和 Xamarin 的 AWS 移動開發工具包，請在軟件包管理器控制台中為每個軟件包運行安裝包命令。例如，要安裝 Cognito 身份，請運行以下命令。 ：

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

所有項目都需要 AWS 核心運行時和 Amazon Cognito 身份包。以下是每個服務的軟件包名稱的完整列表。

服務	套件名稱
AWS 核心運行時間	AWSSDK.core
Amazon Cognito Sync	AWSSDK.CognitoSync
Amazon Cognito 身分	AWSSDK.CognitoIdentity
Amazon DynamoDB	AWSSDK.動態oDBv2
Amazon Mobile Analytics	AWSSDK.MobileAnalytics
Simple Storage Service (Amazon S3)	AWSSDK.S3
Amazon SNS	AWSSDK。SimpleNotificationService (服務)

要包含售前發佈包，請在`-Pre`命令行參數，同時安裝軟件包，如下所示。 ：

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

您可以在以下網址找到 AWS 服務包的完整列表：AWS 開發套件NuGet或適用於 .NET 的 AWS 開發套件GitHub儲存庫。

選項 2：通過使用 IDE 進行安裝

Visual Studio

1. 用鼠右鍵按一下專案，然後按一下Manage (管理)NuGet套件。
2. 搜尋您要新增至項目的套件名稱。要包括預租用NuGet軟件包，選擇包含預租用。您可以在以下網址找到 AWS 服務包的完整列表：AWS 開發套件NuGet。
3. 選擇套件，然後選擇 Install (安裝)。

Mac

Visual Studio

1. 用鼠右鍵按一下套件文件夾，然後選擇新增套件。
2. 搜尋您要新增至項目的套件名稱。要包括預租用NuGet軟件包，選擇顯示預發佈軟件包。您可以在以下網址找到 AWS 服務包的完整列表：AWS 開發套件NuGet。
3. 選取您希望下載的套件旁的核取方塊，然後選擇新增套件。

> **Important**
>
> 如果您使用便攜式類庫開發，您還必須將AWSSDK.coreNuGet包添加到從可移植類庫派生的所有項目。

步驟 5：配置適用於 .NET 和 Xamarin 的 AWS Mobile SDK

設定記錄

您可以通過使用`Amazon.AWSConfigs`類別和`Amazon.Util.LoggingConfig`類別。您可以在`AWSSdk.Core`組件，可以通過 Visual Studio 中的 Nuget 套件管理員獲取。您可以將日誌記錄設置代碼放在`OnCreate`方法`MainActivity.cs`文件或`AppDelegate.cs`文件中的 iOS 應用程序。您也應該新增`using Amazon`和`using Amazon.Util`語句添加到 .cs 文件中。

按如下方式配置日誌記錄設置。 ：

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

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

當您登錄到SystemDiagnostics，框架內部將輸出打印到 System.Console。如果要記錄 HTTP 響應，請將LogResponses旗標。這些值可以是「始終」、「從不」或OnError。

您還可以通過使用LogMetrics屬性。日誌格式可以通過使用LogMetricsFormat屬性。有效值是 JSON 或標準值。

設置區域終端節點

按以下方式為所有服務客户端設定默認區域。 ：

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

這將為 SDK 中的所有服務客户端設置默認區域。您可以通過在創建服務客户端實例時顯式指定區域來覆蓋此設置，如下所示。 ：

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

配置 HTTP 代理服務器設置

如果您的網絡位於代理後面，則可以按如下方式為 HTTP 請求配置代理設置。

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

糾正時鐘扭曲

此屬性確定 SDK 是否應通過確定正確的服務器時間並以正確的時間重新發出請求來糾正客户端時鐘偏差。

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

如果服務調用導致異常，並且 SDK 已確定本地時間和服務器時間之間存在差異，則設置此字段。

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

若要進一步了解時鐘扭曲，請參時鐘偏斜校正在 AWS 博客上。

後續步驟

現在您已經設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK in，您可以：

- 開始使用。閱讀適用於 .NET 和 Xamarin 的 AWS Mobile SDK以取得有關如何使用和配置適用於 .NET 和 Xamarin 的 AWS Mobile SDK in 的快速入門說明。
- 瀏覽服務主題。瞭解每項服務及其在適用於 .NET 和 Xamarin 的 AWS Mobile SDK in 的工作原理。
- 執行示範。查看我們的範例 Xamarin 應用程式，展示常見的使用案例。要運行示例應用程序，請按照前述的方式設置適用於 .NET 和 Xamarin 的 AWS 移動開發工具包，然後按照單個示例的自述文件中包含的說明進行操作。
- 瞭解 API。檢視|sdk-xamarin-ref|_。
- 提問問題：將問題發佈到AWS Mobile SDK或者打開問題GitHub。

適用於 .NET 和 Xamarin 的 AWS Mobile SDK

適用於 .NET 和 Xamarin 的 AWS 移動開發工具包提供了從 Xamarin 應用程序調用 AWS 服務所需的庫、示例和文檔。

您必須完成設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK，然後再開始使用以下服務。

這些入門主題將引導您完成以下內容：

主題

- 使用 Amazon S3 存儲和檢索文件
- 通過認知同步同步用户數據
- 使用 DynamoDB 存儲和檢索數據
- 用 Amazon Mobile Analytics 追蹤應用程式用量資料
- 使用 SNS 接收推送通知
- 使用 SNS 接收推送通知

如需其他 AWS Mobile SDKAWS Mobile SDK。

使用 Amazon S3 存儲和檢索文件

Amazon Simple Storage Service (Amazon S3) 為移動開發人員提供安全、耐用、高可擴展性的物件儲存空間。Amazon S3 易於使用，提供了簡單好用的 Web 服務界面，可從 Web 中的任意位置，來儲存和檢索任意數量的資料。

下面的教學課程介紹如何集成 S3TransferUtility，這是一個用於將 S3 用於應用程序的高級實用程序。如需使用 Xamarin 應用程式 S3 的詳細資訊，請參Amazon Simple Storage Service (S3)。

項目設定

先決條件

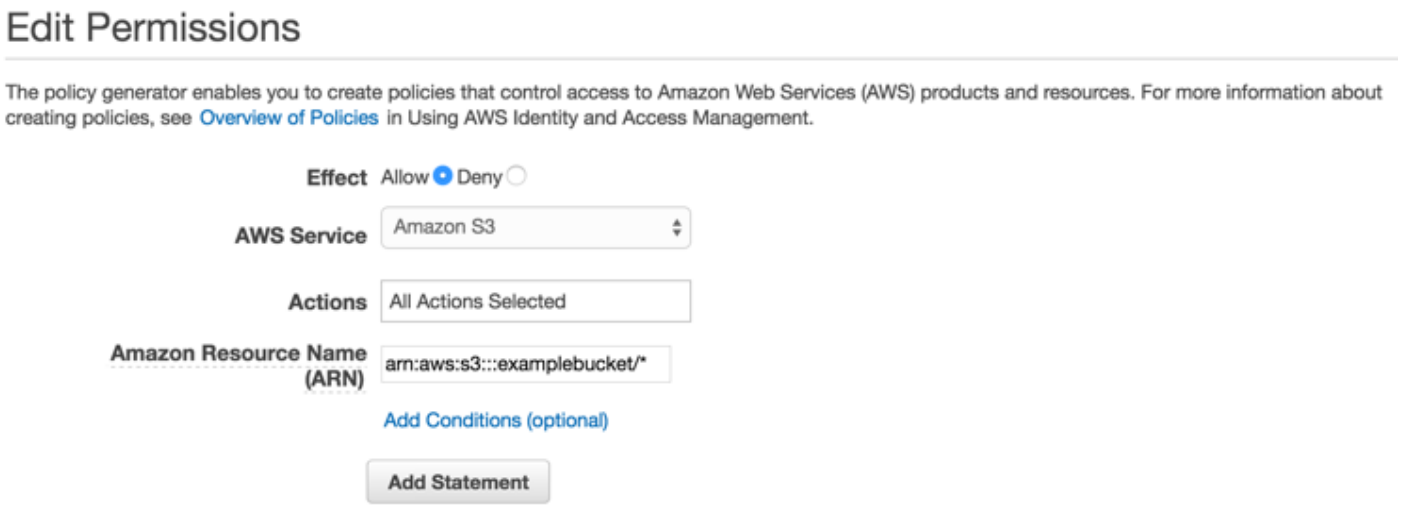
您必須完成設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK，然後再開始本教程。

本教學課程還假設您已建立 S3 儲存貯體。若要建立 S3 儲存貯體，請訪問S3 AWS 主控台。

設定 S3 的許可

默認 IAM 角色政策授予您的應用程式存取 Amazon Mobile Analytics 和 Amazon Cognito Sync 的權限。要使您的 Cognito 身份池能夠訪問 Amazon S3，您必須修改身份池的角色。

1. 前往Identity and Access Management，然後按一下角色在左側窗格中。
2. 在搜尋方塊中輸入您的身分集區名稱。將會列出兩個角色：一個用於未經驗證的使用者，另一個用於通過驗證的使用者。
3. 按一下未通過身份驗證的用戶的角色（它將在您的身份池名稱後附加無授權）。
4. 按一下建立角色政策，選擇政策產生器，然後按一下選擇。
5. 在編輯許可頁面上，輸入下圖中顯示的設定，將 Amazon Resource Name (ARN) 更換為您自己的資源名稱 (ARN)。S3 儲存貯體的 ARN 看起來像`arn:aws:s3:::examplebucket/*`，由儲存貯體所在的區域和儲存貯體名稱組成。下面顯示的設置將為您的身份池提供完全訪問指定存儲桶的所有操作。



1. 按一下新增陳述式按鈕，然後單擊後續步驟。
2. 嚮導將顯示您產生的配置。按一下應用政策。

如需授予 S3 存取權的詳細資訊，請參授予 Amazon S3 儲存貯體的存取權。

AddNuGetS3 的軟件包到您的項目

請按照設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK添加 S3NuGet套件添加到您的項目。

（可選）配置 S3 請求的簽名版本

與 Amazon S3 的每次互動，可以經過驗證身份或是匿名進行。AWS 使用簽名版本 4 或簽名版本 2 算法對服務的調用進行身份驗證。

2014 年 1 月之後創建的所有新 AWS 區域僅支持簽名版本 4。但是，許多較舊的區域仍然支援 Signature 第 4 版和 Signature 第 2 版請求。

如果您的存儲桶位於不支持簽名版本 2 請求的區域之一，如此頁面，您必須設定 AWSConfigsS3.UseSignature版本 4 屬性設置為「true」，如下所示：

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

如需 AWS Signature 版本的詳細資訊，請參驗證請求（AWS Signature 第 4 版）。

初始化 S3TransferUtility用戶端

創建一個 S3 客户端，將其傳遞您的 AWS 證書對象，然後將 S3 客户端傳遞給傳輸實用程序，如下所示：

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

將檔案上傳到 Amazon S3

若要將檔案上傳到 S3，請調用Upload，傳遞以下參數：

- file-您要上傳的檔案字符串名稱
- bucketName-存放檔案的 S3 儲存貯體的字符串名稱

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

上面的代碼假設目錄環境中有一個檔案。SpecialFolder。ApplicationData。上傳功能會自動對大文件使用 S3 的多段上傳功能，以提高吞吐量。

從 Amazon S3 下載檔案

若要從 S3 下載檔案，請調用Download，傳遞以下參數：

- file-您要下載的檔案的字符串名稱
- bucketName-您要從中下載檔案的 S3 儲存貯體的字符串名稱
- key-一個字符串，表示要下載的 S3 對象（在本例中是一個文件）的名稱

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

有關從 Xamarin 應用程序訪問 Amazon S3 的詳細信息，請參閱Amazon Simple Storage Service (S3)。

通過認知同步同步用户數據

Amazon Cognito Sync 可更輕鬆地在 AWS 雲端儲存手機使用者資料，例如應用程式偏好設定或遊戲狀態，而不必編寫後端程式碼或管理任何基礎設施。您可以在使用者的本機裝置上儲存資料。如此一來，即使裝置處於離線狀態，應用程式仍然可以運作。您還可以在使用者的多台裝置間同步資料，無論他們使用哪一台裝置，皆能維持一致的應用程式體驗。

以下教學課程介紹如何將同步與您的應用程式集成。

項目設定

先決條件

您必須完成設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK，然後再開始本教程。

授予對您的 Cognito 同步資源的訪問權限

與您在安裝過程中創建的未經身份驗證和身份驗證角色相關聯的默認策略授予應用程序對 Cognito Sync 的訪問權限。無需進一步設定。

AddNuGet用於 Cognito 同步到您的項目的軟件包

請按照中的說明步驟 4設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK添加 CognitoSyncManager NuGet套件添加至您的項目。

初始化CognitoSync經理

將您已初始化的 Amazon Cognito 登入資料供應商傳遞到 `CognitoSyncManager` 建構函數：

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

同步使用者資料

要同步未經身份驗證的用户數據：

1. 建立資料集。
2. 將使用者資料添加至資料集。
3. 將數據集與雲同步。

建立資料集

建立 `Dataset` 的執行個體。所以此openOrCreate資料集方法可用於建立新的資料集或開啟裝置上本機儲存的資料集的現有實例：

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

將用户數據添加到數據集

使用者資料以索引鍵/值對的形式添加：

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

Cognito 資料集的功能有如字典，可依索引鍵來存取值：

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

同步資料集

若要同步資料集，請調用其同步方法：

```
dataset.SynchronizeAsync();

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

寫入數據集的所有數據都將在本地存儲，直到數據集同步為止。本節中的代碼假定您使用的是未經身份驗證的 Cognito 身份，因此當用户數據與雲同步時，它將按設備存儲。設備具有與其關聯的設備 ID。當用户數據同步到雲時，它將與該設備 ID 相關聯。

有關 Cognito 同步的詳細信息，請參閱Amazon Cognito Sync。

使用 DynamoDB 存儲和檢索數據

Amazon DynamoDB 是一種快速、可輕鬆擴展、高度可用、經濟實惠、非關聯式資料庫服務。DynamoDB 移除資料儲存體的傳統擴展性限制，而仍維持低延遲及可預期的效能。

下面的教學課程介紹如何將 DynamoDB 對象持久化模型與您的應用集成，該模型會在 DynamoDB 中存儲對象。

項目設定

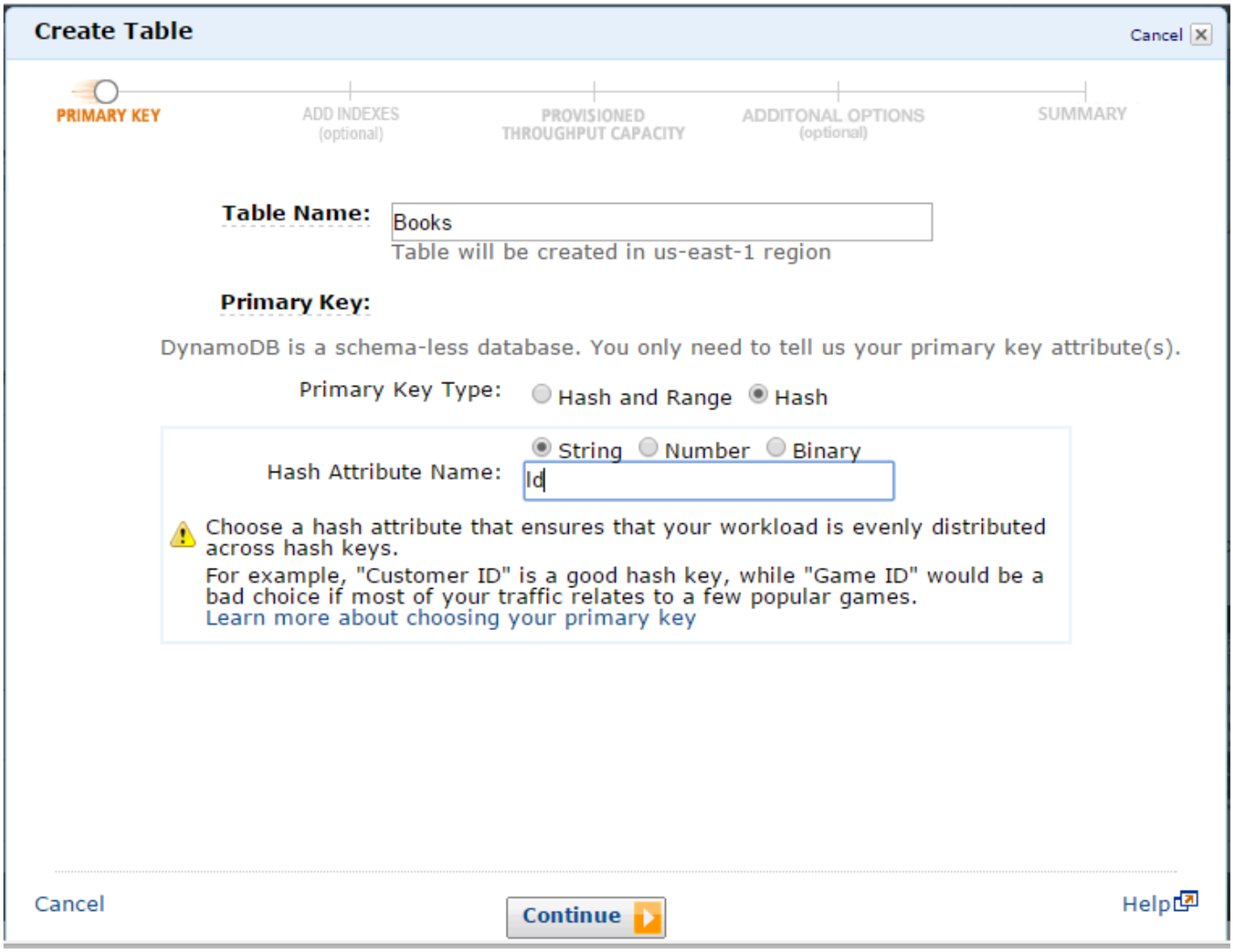
先決條件

您必須完成設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK，然後再開始本教程。

建立 DynamoDB 資料表

您必須先建立資料表，然後才可以將數據寫入 DynamoDB 數據庫。建立資料表時，您必須指定主索引鍵。主索引鍵包含散列屬性和一個可選的範圍屬性。如需如何使用主要屬性和範圍屬性的詳細資訊，請參。處理表格。

1. 前往DynamoDB 主控台，然後按一下建立資料表。建立資料表精靈隨即出現。
2. 指定表名、主鍵類型（哈希）和哈希屬性名稱（「Id」），如下所示，然後單擊Continue：



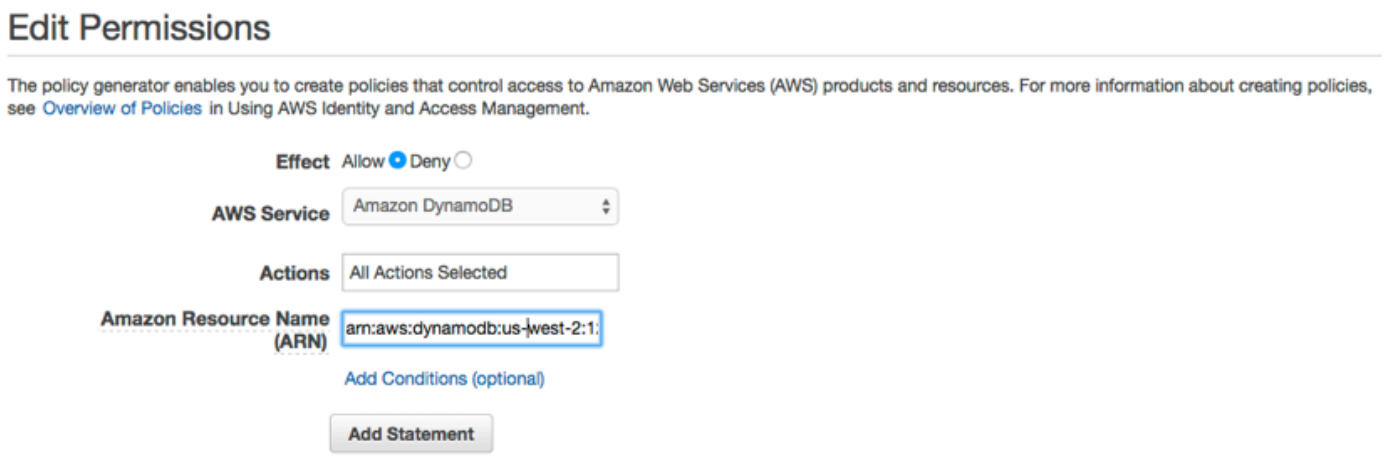
3. 將下一個屏幕中的編輯字段留空，然後單擊Continue。
4. 接受個讀取容量單位和寫入容量單位，然後按一下Continue。
5. 在下一個屏幕上，在Send notification to:文字方塊，然後按一下Continue。審核畫面隨即出現。
6. 按一下 Create (建立)。建立資料表需要幾分鐘的時間來完成。

設定 DynamoDB 的許可

要使您的身份池能夠訪問 Amazon DynamoDB，您必須修改身份池的角色。

1. 導覽至 。Identity and Access Management，然後按一下角色在左側窗格中。搜索您的身分集區名稱-將會列出兩個角色，一個用於未驗證的使用者，一個用於已驗證的使用者。
2. 按一下未經驗證使用者的角色 (它會在您的身分集區名稱後附加「unauth」)，然後按一下建立角色政策。

3. 選擇政策產生器，然後按一下選擇。
4. 在編輯許可頁面上，輸入下圖顯示的設定。DynamoDB 資源名稱 (ARN) 類似`arn:aws:dynamodb:us-west-2:123456789012:table/Books`，由表所在的區域、所有者的 AWS 賬戶號碼以及表格的名稱組成，格式為`table/Books`。如需如何指定 ARN 的詳細資訊，請參。DynamoDB 的 Amazon Resource Name。



5. 按一下新增陳述式，然後按一下後續步驟。嚮導會顯示產生的組態。
6. 按一下應用政策。

AddNuGet軟件包 DynamoDB 到您的項目中

請按照中的說明步驟 4設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK添加 DynamoDB 庫NuGet套件至您的項目。

InitializeAmazonDynamoDB客户端

將您已初始化的 Amazon Cognito 憑證提供商及您的區域傳遞至`AmazonDynamoDB`構造函數，然後將用户端傳遞給 DynamoDBContext：

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

建立類別

要將行寫入表，請定義一個類來保存您的行數據。類還應包含保存該行的屬性數據的屬性，並將映射到在控制台中創建的 DynamoDB 表。以下類聲明說明瞭這樣的類：

```
[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; }
}
```

儲存項目

要保存項目，請首先創建一個對象：

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

然後保存它：

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

要更新行，請修改DDTableRow類別並調用AWSDynamoObjectMapper.save()如上所示。

擷取項目

使用主鍵檢索項目：

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

更新項目

若要更新項目：

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

刪除項目

刪除項目：

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

有關從 Xamarin 應用程序訪問 DynamoDB 的詳細信息，請參閱Amazon DynamoDB。

用 Amazon Mobile Analytics 追蹤應用程式用量資料

Amazon Mobile Analytics 讓您可以測量應用程式用量和應用程式收入。通過跟蹤新用户與返回用户、應用收入、用户保留率和自定義應用內行為事件等關鍵趨勢，您可以做出數據驅動的決策，以提高應用的參與度和盈利。

以下教學課程介紹如何集成 Mobile Analytics 與您的應用程式。

項目設定

先決條件

您必須完成設定適用 .NET 和 Xamarin 的 AWS Mobile SDK，然後再開始本教程。

在 Mobile Analytics 控制台中創建應用

前往Amazon Mobile Analytics並建立應用程式。請注意`appId`值，因為您稍後會用到。在 Mobile Analytics 控制台中創建應用時，您需要指定身份池 ID。若要進一步了解建立身分集區，請參設定適用 .NET 和 Xamarin 的 AWS Mobile SDK。

若要進一步了解在主控台中工作，請參Amazon Mobile Analytics 用户指南。

設置 Mobile Analytics 的權限

與您在安裝過程中創建的角色相關聯的默認策略授予您的應用程序對 Mobile Analytics 的訪問權限。無需進一步設定。

AddNuGet針對您的項目的 Mobile Analytics 軟件包

請按照中的說明步驟 4設定適用 .NET 和 Xamarin 的 AWS Mobile SDK添加 Mobile AnalyticsNuGet軟件包添加到您的項目。

配定 Mobile Analytics 設定

Mobile Analytics 定義了一些可以在 awsconfig.xml 文件中配置的設置：

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

- AllowUseDataNetwork-一個布爾值，它指定會話事件是否在數據網絡上發送。
- DBWarningThreshold-這是對數據庫大小的限制，一旦達到，將生成警告日誌。
- MaxDBSize-這是 SQLite 數據庫的大小。當數據庫達到最大大小時，將刪除任何其他事件。
- MaxRequest大小-這是應在 HTTP 請求中傳輸到移動分析服務的請求的最大大小（以字節為單位）。
- SessionTimeout-這是應用程序進入後台和會話可以終止的時間間隔。

上面顯示的設置是每個配置項目的默認值。

InitializeMobileAnalytics經理

初始化MobileAnalytics經理，呼叫GetOrCreateInstance您的`MobileAnalyticsManager`，傳遞 AWS 證書、您的區域、Mobile Analytics 應用程序 ID 和可選配置對象：

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

跟蹤工作階段事件

Xamarin Android

覆蓋活動的OnPause()和OnResume()方法來記錄會話事件。

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

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

這需要為應用程序中的每個活動實現。

Xamarin iOS

在您的AppDelegate.cs：

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

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

有關 Mobile Analytics 的詳細信息，請參閱Amazon Mobile Analytics。

使用 SNS 接收推送通知

本文檔介紹如何使用 Amazon Simple Notification Service (SNS) 和適用於 .NET 和 Xamarin 的 AWS 移動軟件開發工具包向 Xamarin iOS 應用程序發送推送通知。

項目設定

先決條件

您必須完成設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK，然後再開始本教程。

設定 SNS 的許可

按照步驟 2設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK將下面提到的策略附加到應用程序的角色。這將為您的應用程序授予訪問 SNS 的適當權限：

1. 前往IAM 主控台並選擇您想要配置的 IAM 角色。
2. 按一下連接政策下，選擇卓越亞馬遜 SNFullAccess策略，然後單擊連接政策。

> **Warning**
>
> 使用卓越亞馬遜 SNFullAccess不建議在生產環境中使用。我們在此使用它來幫助您快速啟動並執行。如需有關指定 IAM 角色的許可的詳細資訊，請參IAM 角色許可概觀。

獲得蘋果 iOS 開發者計劃的會員資格

您需要在物理設備上運行應用程序才能接收推送通知。要在設備上運行應用程序，您必須在蘋果 iOS 開發者計劃會員。擁有成員資格後，您可以使用 Xcode 來生成簽名身份。如需詳細資訊，請參 Apple 的應用程式分發快速入門文件中)。

創建 iOS 證書

首先，您需要建立 iOS 憑證。然後，您需要創建一個配置為推送通知配置的置備配置文件。作法：

1. 前往蘋果開發者會員中心下，按一下證書、標識符和配置文件。
2. 按一下識別碼下iOS 應用程式，請單擊網頁右上角的加號按鈕以添加新的 iOS 應用程序 ID，然後輸入應用 ID 描述。
3. 向下捲動到新增 ID 後綴部分，然後選擇明確應用程式 ID並輸入捆綁標識符。
4. 向下捲動到應用程式服務部分，然後選擇推送通知。
5. 按一下 Continue (繼續)。
6. 請按 Submit (提交)。
7. 按一下完成。

8. 選擇剛剛創建的應用程序 ID，然後單擊Edit (編輯)。
9. 向下捲動到推送通知區段。按一下建立憑證下開發 SSL 憑證。
10. 按照說明創建證書簽名請求 (CSR)、上傳請求並下載將用於與 Apple 通知服務 (APNS) 通信的 SSL 證書。
11. 返回證書、標識符和配置文件(憑證已建立！) 頁面上的名稱有些許差異。按一下All (全部)下設定檔。
12. 單擊右上角的加號按鈕以添加新的配置文件。
13. 選擇iOS 應用程式開發，然後按一下Continue。
14. 選擇您的應用程式 ID，然後按一下Continue。
15. 選擇您的開發者憑證，然後按一下Continue。
16. 選取您的設備，然後按一下Continue。
17. 輸入配置式名稱，然後按一下Generate。
18. 下載並雙擊置備文件以安裝置配置文件。

有關置備為推送通知配置的配置文件的詳細信息，請參閱 Apple 的配置推播通知文件中)。

使用證書在 SNS 控制台中創建平台 ARN

1. 執行KeyChain訪問應用程序，選擇我的憑證，然後右鍵單擊您生成的 SSL 證書以連接到 APNS，然後選擇匯出。系統將提示您指定文件的名稱和密碼以保護證書。證書將保存在 P12 文件中。
2. 前往SNS 控制台，然後按一下應用程式位於屏幕左側。
3. 按一下建立平台應用程式建立新 SNS 平台應用程式。
4. 輸入應用程式名稱。
5. 選擇Apple 開發為了推播通知平台。
6. 按一下選取檔案，然後選擇導出 SSL 證書時創建的 P12 文件。
7. 輸入您導出 SSL 證書時指定的密碼，然後單擊從檔案載入憑證。
8. 按一下建立平台應用程式。
9. 選擇剛剛創建的平台應用程序並複製應用程序 ARN。您在接下來的步驟中需要此資訊。

AddNuGetSNS 軟件包到您的項目

請按照設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK新增亞馬遜簡單通知服務NuGet軟件包添加至您的項目。

建立 SNS 客户端

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

註冊應用程序以獲得遠程通知

要註冊應用程序，請調用RegisterForRemoteNotifications，如下所示。將下面的代碼放在AppDelegate.cs，在下面提示的位置插入您的平台應用程序 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 */
    });
  }
}
```

從 SNS 控制台向終端節點發送消息

1. 前往SNS 控制台 > 應用程序。

2. 選擇您的平台應用程序，選擇終端節點，然後單擊發佈至端點。
3. 在文字框中輸入短信，然後按一下發佈訊息發佈訊息。

使用 SNS 接收推送通知

本教程介紹瞭如何使用 Amazon Simple Notification Service (SNS) 和適用於 .NET 和 Xamarin 的 AWS 移動軟件開發工具包向 Xamarin Android 應用程序發送推送通知。

專案設定

先決條件

您必須完成設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK，然後再開始本教程。

設定 SNS 的許可

按照步驟 2設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK將下面提到的策略附加到應用程序的角色。這將為您的應用程序授予訪問 SNS 的適當權限：

1. 前往IAM 主控台並選取您想要設定的 IAM 角色。
2. 按一下連接政策，請選取卓越亞馬遜 SNFullAccess策略，然後單擊連接政策。

> **Warning**
>
> 使用卓越亞馬遜 SNFullAccess不建議在生產環境中使用。我們在這裏使用它來幫助您快速啟動並執行。如需為 IAM 角色指定許可的詳細資訊，請參IAM 角色許可概觀。

在谷歌雲端啟用推送通知

首先，添加一個新的谷歌 API 項目：

1. 前往Google 開發人員主控台。
2. 按一下建立專案。
3. 在 中新專案框中，輸入項目名稱，記下項目 ID（稍後需要），然後單擊建立。

接下來，啟用適用於您的專案的 Google Cloud Message (GCM) 服務：

1. 在 中Google 開發人員主控台，您的新專案應該已選取起來。如果沒有，請在頁面頂端的下拉式選取該選項。
2. 選擇API 和驗證位於頁面左側的側欄。
3. 在搜尋方塊中，輸入適用於 Android 的 Google Cloud Message (適用於 Android 的 Google Message)，然後適用於安卓的谷歌雲消息鏈接。
4. 按一下啟用 API。

最後，獲取 API 密鑰：

1. 在谷歌開發人員控制台中，選擇API 和驗證 >登入資料。
2. Unblic公用 API 訪問，請單擊建立新的金鑰。
3. 在 中建立新的金鑰對話框中，單擊服務器密鑰。
4. 在生成的對話框中，單擊建立並複製顯示的 API 密鑰。您稍後將使用此 API 密鑰執行身份驗證。

使用項目 ID 在 SNS 控制台中創建平台 ARN

1. 前往SNS 控制台。
2. 按一下應用程式在畫面左側。
3. 按一下建立平台應用程式建立新 SNS 平台應用程式。
4. 輸入應用程式名稱。
5. 選擇Google Cloud Message (GCM)為了推播通知平台。
6. 將 API 密鑰粘貼至標記為API 金鑰。
7. 按一下建立平台應用程式。
8. 選擇剛剛創建的平台應用程序並複製應用程序 ARN。

AddNuGetSNS 軟件包到您的項目

請按照中的說明步驟 4設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK添加 Amazon Simple Notification ServiceNuGet包添加至您的專案。

建立 SNS 客户端

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

註冊應用程序以獲得遠程通知

若要在 Android 上註冊遠程通知，您需要建立BroadcastReceiver，它可以接收谷歌雲消息。在提示時更改下面的軟件包名稱：

```
[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);
      }
}
```

以下是接收來自BroadcastReceiver並在設備的通知欄上顯示通知：

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

從 SNS 控制台向終端節點發送消息

1. 前往SNS 控制台 > 應用程序。
2. 選擇您的平台應用程序，選擇終端節點，然後單擊發佈至終端。
3. 在文字方塊中輸入文字信息，然後單擊發佈訊息發佈訊息。

Amazon Cognito Identity

什麼是 Amazon Cognito 身分？

Amazon Cognito 身分可讓您為使用者建立唯一身分，並且向身分供應商進行身分驗證。有了身分，您可以取得有限權限的臨時 AWS 登入資料來與 Amazon Cognito 同步資料，或直接存取其他 AWS 服務。Amazon Cognito 身分支援公有身分供應商 (Amazon、Facebook 和 Google)，以及未驗證的身分。它也支援開發人員驗證的身分，可讓您透過自己的後端身分驗證程序來註冊及驗證使用者。

如需 Cognito 身分的詳細資訊，請參 Cognito 身分Amazon Cognito 開發人員指南。

如需 Cognito 驗證區域可用性的相關資訊，請參AWS 服務區域可用性。

使用公共提供程序對用户進行身份驗證

利用 Amazon Cognito 身分，您可以為使用者建立唯一身分並驗證它們，以安全存取您的 AWS 資源，如 Amazon S3 或 Amazon DynamoDB。Amazon Cognito 身分支援公有身分供應商 (Amazon、Facebook、Twitt/ 數字、Google 或任何 OpenID 連接兼容供應商)，以及未驗證的身分。

有關使用亞馬遜、Facebook、Twitt/ 數字或谷歌等公共身份提供商對用户進行身份驗證的信息，請參閱外部供應商在 Amazon Cognito 開發人員指南。

使用開發人員驗證的身分

除了透過 Facebook、Google 和 Amazon 的 Web 聯合身分，Amazon Cognito 還支援開發人員驗證的身分。有了開發人員驗證的身分，您可以透過自己現有的身份驗證程序來註冊及驗證使用者，同時仍使用Amazon Cognito Sync來同步用户數據並訪問 AWS 資源。使用開發人員驗證的身分時，需要最終使用者裝置、身分驗證後端與 Amazon Cognito 之間的互動。

有關開發人員經過身份驗證的身份的信息，請參閱開發人員驗證的身份在 Amazon Cognito 開發人員指南。

Amazon Cognito Sync

什麼是 Amazon Cognito Sync？

Cognito Sync 是 AWS 服務和用户端程式庫，可讓您跨裝置同步用户數據（如遊戲分數、用户首選項、遊戲狀態）。您可以使用 Cognito 同步 API 跨設備同步用户數據。要在應用中使用 Cognito 同步，您必須在項目中包含 |。

如需如何將 Amazon Cognito Sync 集成到應用程式中的指示，請參Amazon Cognito Sync 開發者指南。

Amazon Mobile Analytics

Amazon Mobile Analytics是一種能大規模收集、視覺化、了解和抽取應用程式用量資料的服務。Mobile Analytics 可以輕鬆捕獲標準裝置資料和自訂事件，並代替您自動計算報告。除了下列出的彙總報告，您也可以設定將資料自動匯出到 Redshift 和 S3 以進一步分析。

使用 Amazon Mobile Analytics，您可以跟蹤買家行為、彙總指標、生成數據可視化以及識別有意義的模式。

重要概念

報告類型

開箱即用，Mobile Analytics 在 Mobile Analytics 控制台中提供以下報告：

- 每日作用中使用者 (DAU)、每月作用中使用者 (MAU)，和新使用者
- 黏著度 (DAU 除以 MAU)
- 根據每日作用中使用者的工作階段計數和平均工作階段
- 根據每日作用中使用者的平均收入 (ARPDAU) 和根據每日付費作用中使用者的平均收入 (ARPPDAU)
- 1、3、7 天後留存率和 1、2、3 週後留存率
- 自訂事件

主控台中的六個報告標籤提供以下報告：

- 概觀— 跟蹤九個預選報告simple-to-review儀錶板來快速瞭解參與度：MAU, DAU, 新使用者, 每日工作階段, 黏著度, 1 天後留存率, ARPDAU, 每日工作階段, ARPPDAU.
- 作用中使用者— 追蹤每天和每月有多少使用者使用您的應用程式，並監控黏著度以估計熱烈度、吸引力和獲利。
- 工作階段— 追蹤應用程式在某一天的使用頻率，以及一天中每個使用者開啟應用程式的頻率。
- Retention— 追蹤客户每天和每週再次使用應用程式的速率。
- Revenue (營收)— 追蹤應用程式內收入的趨勢，以找出可改善獲利的領域。
- 自訂事件— 追蹤應用程式特定的自訂自訂使用者動作。

要瞭解有關 Mobile Analytics 報表和在 Mobile Analytics 控制台中使用的詳細信息，請參閱Mobile Analytics 主控台報告概觀(在 Mobile Analytics 開發人員指南中)。

項目設定

先決條件

要在應用程序中使用 Mobile Analytics，您需要將 SDK 添加到您的項目中。若要執行此作業，請依照設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK。

配置 Mobile Analytics 設定

Mobile Analytics 定義了一些可以在 awsconfig.xml 文件中配置的設置：

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

- SessionTimeout-如果應用程序在後台停留的時間大於SessionTimeout，則 Mobile Analytics 客户端將終止當前會話，並在應用返回前台時創建一個新會話。建議使用 5 到 10 之間的值。預設值為 5。
- MaxDBSize-用於事件的本地存儲的資料庫大小上限 (以字節為單位)。如果數據庫大小超過此值，將忽略其他事件。我們建議使用從 1MB 到 10MB 的值。默認值為 5242880（5 MB）。
- DBWarningThreashold-警告閾值。有效值範圍介於 0-1 之間。如果值超過閾值，將生成警告日誌。預設值為 0.9。
- MaxRequest大小-向 Mobile Analytics 服務發出的 HTTP 請求的最大大小。該值以字節為單位指定，範圍介於 1-512KB 之間。默認值的操作系統 不要使用大於 512KB 的值，這可能會導致服務拒絕 HTTP 請求。
- AllowUseDataNetwork-一個值，指示是否允許通過蜂窩移動數據網絡進行服務呼叫。請謹慎使用此選項，因為這可能會增加客户的數據使用率。

上面顯示的設置是每個配置項目的默認值。

將 Mobile Analytics 與您的應用程式整合

以下部分介紹瞭如何將 Mobile Analytics 與您的應用集成。

在 Mobile Analytics 控制台中創建應用

前往Amazon Mobile Analytics 主控台並建立應用程式。請注意appId值，因為以後會用到。在 Mobile Analytics 控制台中創建應用時，您需要指定身份池 ID。如需有關建立身分集區的說明，請參設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK。

若要進一步了解在 Mobile Analytics 主控台中工作，請參Mobile Analytics 主控台報告概觀(在 Mobile Analytics 開發人員指南中)。

建立MobileAnalytics管理員客户端

若要初始化MobileAnalytics經理，呼叫GetOrCreateInstance在您的MobileAnalyticsManager，傳遞您的 AWS 證書、您的區域、Mobile Analytics 應用程序 ID 和可選配置對象：

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

所以此APP_ID在應用程序創建嚮導期間為您生成。這兩個值都必須與 Mobile Analytics 控制台中的值匹配。所以此APP_ID用於在 Mobile Analytics 控制台中對您的數據進行分組。要在 Mobile Analytics 控制台中創建應用後查找應用 ID，請瀏覽至 Mobile Analytics 控制台，單擊屏幕右上角的齒輪圖標。這將顯示應用程序管理頁面，其中列出了所有已註冊的應用程序及其應用 ID。

記錄獲利事件

適用於 .NET 和 Xamarin 的 AWS Mobile SDK 提供MonetizationEvent類，使您可以生成盈利事件以跟蹤在移動應用程序中進行的購買。以下程式碼片段演示如何建立獲利事件：

```
// 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);
```

記錄自訂事件

Mobile Analytics 允許您定義自定義事件。自定義事件完全由您定義；它們可幫助您跟蹤特定於應用或遊戲的用户操作。如需自訂事件的詳細資訊，請參自訂事件。

在這個例子中，我們會說我們的應用程序是一個遊戲，並且我們希望在用户完成一個關卡時記錄一個事件。建立」LevelComplete」事件，方法是創建一個新的AmazonMobileAnalyticsEvent實例：

```
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);
```

記錄工作階段

Xamarin iOS

當應用程序失去焦點時，您可以暫停會話。對於 iOS 應用程序，請在 AppDelegate.cs 文件, 覆蓋DidEnterBackground和WillEnterForeground呼叫MobileAnalyticsManager.PauseSesion和MobileAnalyticsManager.ResumeSession，如下程式碼片段所示：

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

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

Xamarin

對於安卓應用程序調用MobileAnalyticsManager.PauseSesion中的OnPause() 方法和MobileAnalyticsManager.ResumeSession中的OnResume() 方法，如下程式碼片段所示：

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

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

默認情況下，如果用户將焦點從應用程序切換到不到 5 秒，並切換回應用程序，則會話將恢復。如果用户將焦點從應用程序切換為 5 秒或更長時間，將創建一個新會話。通過將「SESSION_DELTA」屬性設置為創建新會話之前等待的秒數，可以在 aws_Mobile_json 配置文件中配置此設置。

Amazon Simple Storage Service (S3)

什麼是 S3？

Amazon Simple Storage Service (Amazon S3)，為開發人員提供安全、耐用、高度可擴充的物件儲存。Amazon S3 易於使用，透過簡單好用的 Web 服務界面，可從 Web 中的任意位置，來儲存和檢索任意數量的資料。使用 Amazon S3，您只需按實際使用的儲存容量付費。沒有最低費用也沒有設定費。

Amazon S3 為各種使用案例提供經濟高效的對象存儲，包括雲應用程序、內容分發、備份和歸檔、災難恢復和大數據分析。

如需 AWS S3 區域可用性的相關資訊，請參AWS 服務區域可用性。

重要概念

儲存貯體

您存放在 Amazon S3 的每個物件都位在儲存貯體內。您可以使用儲存貯體將相關物件分組，其方式與使用目錄將檔案系統中的檔案分組相同。存儲桶具有諸如訪問權限和版本控制狀態等屬性，您可以指定希望它們駐留的區域。

若要進一步了解 S3 儲存貯體，請參使用儲存貯體《S3 開發人員指南》中的資訊。

物件

物件是您存放在 Amazon S3 的資料。每個物件都位在您於特定 AWS 區域中建立的儲存貯體內。

除非您明確地將存放在區域中的物件傳輸到其他區域，否則物件絕對不會離開該區域。例如，存放在 EU (愛爾蘭) 區域的物件絕不會離開此區域。存放在 Amazon S3 區域的物件實際上仍會留在該區域。Amazon S3 不會保留複本，或是將物件移動到任何其他的區域。不過，只要您有必要許可，就可以從任何位置存取物件。

物件可以是任意檔案類型，包括影像、備份資料、影片等等。物件最大可達 5 TB。儲存貯體中可以有不限數目的物件。

您必須具備儲存貯體的寫入許可，才能將物件上傳至 Amazon S3。如需設定儲存貯體許可的詳細資訊，請參編輯儲存貯體許可《S3 開發人員指南》中的資訊。

若要進一步了解 S3 物件，請參使用物件《S3 開發人員指南》中的資訊。

物件中繼資料

Amazon S3 中的每個物件都有一組代表其中繼資料的鍵值對代表其中繼資料。有兩種類型的中繼資料：

- 系統元數據— 有時由 Amazon S3 處理，例如內容類型和內容長度。
- 使用者資料— 從不會由 Amazon S3 進行處理。使用者中繼資料會與物件一起存放，並與其一起傳回。使用者中繼資料的大小上限為 2 KB，而且索引鍵與其值必須符合 US-ASCII 標準。

若要進一步了解 S3 物件中繼資料，請參編輯物件中繼資料。

項目設定

先決條件

要在應用程序中使用 Amazon S3，您需要將開發工具包添加到您的項目中。若要執行此作業，請依照設定適用 .NET 和 Xamarin 的 AWS 移動開發套件。

建立 S3 儲存貯體

Amazon S3 將應用程序的資源存儲在 Amazon S3 存儲桶中-存儲在特定區域。每個 Amazon S3 儲存貯體都必須具有全域唯一的名稱。您可以使用Amazon S3 主控台建立儲存貯體。

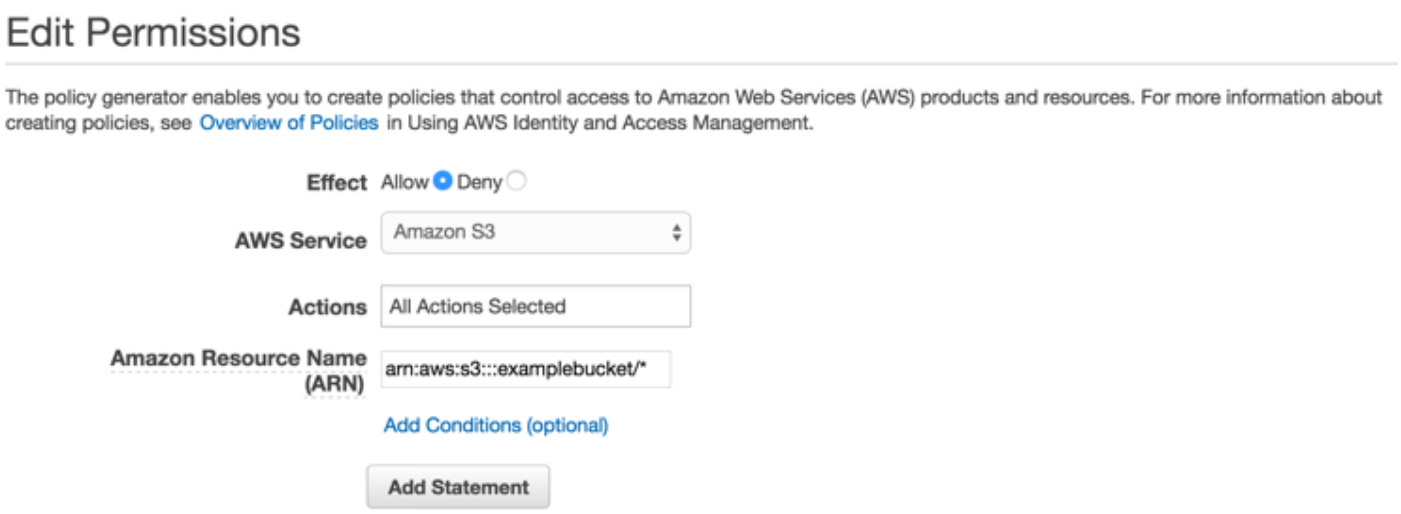
1. 登入Amazon S3 主控台然後按一下建立儲存貯體。
2. 輸入儲存貯體名稱，選擇區域，然後按一下建立。

設定 S3 的許可

默認 IAM 角色政策授權您的應用程式訪問 Amazon Mobile Analytics 和 Amazon Cognito Sync。要使您的 Cognito 身份池能夠訪問 Amazon S3，您必須修改身份池的角色。

1. 前往Identity and Access Management Console然後按一下角色在左側窗格中。
2. 在搜尋方塊中輸入您的身分集區名稱。將會列出兩個角色：一個用於未經驗證的使用者，另一個用於通過驗證的使用者。

3. 按一下未經身份驗證的用戶的角色 (它會在您的身份池名稱後面附加無身份驗證)。
4. 按一下建立角色政策，選擇政策產生器，然後按一下選擇。
5. 在編輯許可頁面上，輸入下圖中顯示的設定，將 Amazon Resource Name (ARN) 替換為您自己的 Amazon Resource Name (ARN)。S3 儲存貯體的 ARN 看起來像`arn:aws:s3:::examplebucket/*`中繼資料，由儲存貯體所在的區域和儲存貯體名稱組成。下面顯示的設置將為您的身份池提供完全訪問指定存儲桶的所有操作。



1. 按一下添加陳述式按鈕，然後單擊後續步驟。
2. 嚮導會顯示您產生的配置。按一下應用政策。

如需授予 S3 訪問權限的詳細資訊，請參[授予對 Amazon S3 儲存貯體的存取權]。

（可選）配置 S3 請求的簽名版本

與 Amazon S3 的每次互動，可以經過驗證身份或是匿名進行。AWS 使用簽名版本 4 或簽名版本 2 算法對服務的調用進行身份驗證。

2014 年 1 月之後創建的所有新 AWS 區域僅支持簽名版本 4。但是，許多較舊的區域仍然支援 Signature 第 4 版和 Signature 第 2 版請求。

如果您的存儲桶位於不支持簽名版本 2 請求的區域之一，如[此頁面]時，您必須設定 AWSConfigsS3.UseSignature版本 4 屬性設置為「true」，如下所示：

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

如需 AWS Signature 版本的詳細資訊，請參驗證請求（AWS Signature 第 4 版）。

將 S3 與您的應用程式整合

有兩種方式可以在 Xamarin 應用程序中與 S3 進行交互。在以下主題中深入探討了這兩種方法：

使用 S3 Transfer File

通過 S3 Transfer Utility，您可以更輕鬆地從 Xamarin 應用程序上傳和下載文件到 S3。

初始化TransferUtility

創建一個 S3 客户端，將其傳遞您的 AWS 證書對象，然後將 S3 客户端傳遞給傳輸實用程序，如下所示：

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

（可選）配置TransferUtility

您可以配置三個可選屬性：

- ConcurrentService請求-確定將使用多少活動線程或併發異步 Web 請求的數量來上傳/下載文件。預設值為 10。
- MinSizeBeforePart上傳-獲取或設置上傳段的最小分段大小（以字節為單位）。預設為 16 MB。減小最小部件大小會導致分段上傳被拆分為更多較小的部分。將此值設置為太低會對傳輸速度產生負面影響，從而導致每個部件的額外延遲和網絡通信。
- NumberOfUploadThreads-獲取或設置執行線程的數量。此屬性確定將使用多少個活動線程來上傳文件。預設值為 10 個線程。

配置 S3TransferUtility客户端，創建一個配置對象，設置屬性，然後將對象傳遞給TransferUtility構造函數，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);
```

下載檔案

若要從 S3 下載檔案，請調用Download，傳遞以下參數：

- file-您想要下載的檔案的字符串名稱
- bucketName-您要從中下載檔案的 S3 儲存貯體的字符串名稱
- key-一個字符串，表示要下載的 S3 對象（在本例中是一個文件）的名稱

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

上傳檔案

若要上傳檔案到 S3，請調用Upload，傳遞以下參數：

- file-您要上傳的檔案字符串名稱
- bucketName-儲存貯體的 S3 儲存貯體字符串名稱

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

上面的代碼假設目錄環境中有一個檔案。SpecialFolder。ApplicationData。上傳功能會自動對大文件使用 S3 的多段上傳功能，以提高吞吐量。

使用服務級別 S3 API

除了使用 S3TransferUtility，您還可以使用低階 S3 API 與 S3 交互。

初始化 Amazon S3 用户端

要使用 Amazon S3，我們首先需要創建一個亞馬遜 3 客户端實例，該實例需要引用 CognitoAWSCredentials實例和您所在的區域：

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

下載檔案

若要從 S3 下載檔案：

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

上傳檔案

若要上傳檔案至 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);
```

刪除項目

刪除 S3 中的項目：

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

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

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

刪除多個物件

要使用單個 HTTP 請求從存儲桶中刪除多個對象，請執行以下操作：

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

您最多可指定 1000 個密鑰。

列出儲存貯體

若要返回已驗證的請求發送者擁有的所有儲存貯體清單：

```
// 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);
```

```
}
```

列出物件

您可以返回 S3 儲存貯體中的部分或全部物件 (最多 1000 個)。若要這麼做，您必須具有儲存儲存貯體的讀取權限。

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

獲取儲存貯體區域

要獲取存儲桶所在的區域，請執行以下操作：

```
// 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);
```

獲取儲存貯體策略

要獲取存儲桶的策略：

```
// 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

什麼是 Amazon DynamoDB？

Amazon DynamoDB是一種快速、可輕鬆擴展的非關聯式資料庫服務。DynamoDB 移除資料儲存體的傳統擴展性限制，而仍維持低延遲及可預期的效能。

重要概念

DynamoDB 資料模型概念包括資料表、項目與屬性。

資料表

在 Amazon DynamoDB 中，資料庫是資料庫是資料庫資料表的集合。資料表是項目的集合，而每個項目則是屬性的集合。

在關係數據庫中，表具有預定義的模式，例如表名稱、主鍵、其列名列表及其數據類型。存儲在資料表中的所有記錄必須具備相同的列集。相比之下，DynamoDB 只要求表具有主鍵，但不要求您提前定義所有屬性名稱和數據類型。

若要進一步了解使用資料表，請參在 DynamoDB 中處理資料表。

項目和屬性

DynamoDB 表中的單個項目可以具有任意數量的屬性，儘管項目大小限製為 400 KB。項目大小是其屬性名稱和值長度的總和（二進制和 UTF-8 長度）。

項目中的每個屬性都是名稱-值對。屬性可以是單一值或多重值集。例如，圖書項目可以具有標題和作者屬性。每本書都有一個標題，但可以有許多作者。多值屬性是一個集合；不允許使用重複值。

例如，請考慮在 DynamoDB 中存儲產品目錄。您可以創建一個表ProductCatalog，Id 屬性作為其主索引鍵。主索引鍵可唯一識別每個項目，因此資料表中的兩種產品不得擁有相同的 ID。

若要進一步了解項目的使用，請參在 DynamoDB 中處理項目。

資料類型

Amazon DynamoDB 支援下列資料類型：

- 純量類型— 數字、字符串、二進制、布爾值和空值。
- 多值類型— 字符串集、數字集和二進制集。
- 文件類型— 列表和地圖。

有關標量數據類型、多值數據類型和文檔數據類型的詳細信息，請參閱DynamoDB 資料類型。

主索引鍵

當您建立資料表時，除了資料表名稱，您還必須指定資料表的主索引鍵。主索引鍵可唯一識別資料表中的每個項目，因此沒有兩個項目的索引鍵是相同的。DynamoDB 支援下列兩種類型的主索引鍵：

- 雜湊鍵：主鍵由一個屬性（哈希屬性）組成。DynamoDB 在此主鍵屬性上構建無序哈希索引。資料表中的每個項目都由其散列鍵值進行唯一識別。
- 哈希和範圍鍵：主鍵由兩個屬性組成。第一個屬性是散列屬性，第二個屬性是範圍屬性。DynamoDB 在散列主索引鍵屬性上建立一個無序的散列索引，並建立範圍主索引鍵屬性的排序範圍索引。資料表中的每個項目都是由其散列索引鍵和範圍索引鍵的組合進行唯一識別。兩個項目可能具有相同的哈希鍵值，但這兩個項目必須具有不同的範圍鍵值。

次要索引

使用散列索引鍵和範圍索引鍵建立資料表時，您可以選擇性地在該資料表上定義一或多個次要索引。次要索引可讓您在除了使用主索引鍵查詢外，也可使用備用索引鍵查詢資料表中的資料。

DynamoDB 支持兩種類型的次要索引：本機次要索引和全域次要索引。

- 本機次要索引：一種與資料表具有相同哈希索引鍵但不同範圍索引鍵的索引。
- 全域次要索引：散列索引鍵和範圍索引鍵可與資料表不同的索引。

每個資料表最多可以定義 5 個全域次要索引和 5 個本機次要索引。如需詳細資訊，請參閱「」在 DynamoDB 中使用二級索引來改善資料存取(在 DynamoDB 開發人員指南中)。

查詢和掃描

除了使用主鍵訪問項目之外，Amazon DynamoDB 還提供了兩個用於搜索數據的 API：查詢和掃描。建議您仔細閱讀查詢和掃描指南，以熟悉一些最佳實踐。

Query

查詢操作只使用主索引鍵屬性值在資料表或次要索引中尋找項目。您必須提供一個哈希鍵屬性名稱和一個要搜索的不同值。您可以選擇性地提供範圍索引鍵屬性名稱和值，並使用比較運算子優化搜尋結果。

如需範例查詢，請參：

- 使用文件模型
- 使用物件持久性模型
- 使用 DynamoDB 服務級別 API

如需查詢的詳細資訊，請參查詢(在 DynamoDB 開發人員指南中)。

Scan

掃描操作會讀取資料表或輔助索引中的每個項目。默認情況下，掃描操作會傳回資料表或索引中每個項目的所有資料屬性。您可以使用ProjectionExpression參數，以便 Scan 只傳回某些屬性，而不是全部。

有關示例掃描，請參閱：

- 使用文件模型
- 使用物件持久性模型
- 使用 DynamoDB 服務級別 API

如需掃描的詳細資訊，請參掃描(在 DynamoDB 開發人員指南中)。

項目設定

先決條件

要在應用程序中使用 DynamoDB，您需要將開發工具包添加到您的項目中。若要執行此作業，請遵循設定適用於 .NET 和 Xamarin 的 AWS 移動開發套件。

建立 DynamoDB 資料表

若要建立資料表，請前往DynamoDB 主控台並按照下列步驟進行：

1. 按一下 Create Table (建立資料表)。
2. 輸入資料表的名稱。
3. 選擇雜湊作為主索引鍵類型。
4. 選擇類型並輸入哈希屬性名稱的值。按一下 Continue (繼續)。
5. 在添加索引頁面上，如果計劃使用全局二級索引，請將索引類型設置為「全球二級指數」和索引散列鍵中，輸入二級索引的值。這將允許您使用主索引和二級索引進行查詢和掃描。按一下將索引添加到表，然後按一下Continue。要跳過使用全局二級索引，請單擊Continue。
6. 將讀取和寫入容量設置為所需級別。如需配置容量的詳細資訊，請參卓越 Amazon DynamoDB 中的預配置吞吐量。按一下 Continue (繼續)。
7. 在下一個屏幕上，輸入通知電子郵件以創建吞吐量警報（如果需要）。按一下 Continue (繼續)。
8. 在摘要頁面上，按一下建立。DynamoDB 將建立您的資料庫。

設定 DynamoDB 的許可

若要在應用程式中使用 DynamoDB，您必須設定正確的權限。以下 IAM 策略允許用户刪除、獲取、放置、查詢、掃描和更新特定 DynamoDB 表中的項目，該表由ARN：

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

您可以修改IAM 主控台。您應根據應用程式的需要而新增或移除允許的操作。

若要進一步了解 IAM 政策，請參閱使用 IAM。

若要進一步了解 DynamoDB 特定策略，請參使用 IAM 控制對 DynamoDB 資源的存取(在 DynamoDB 開發人員指南中)。

將 DynamoDB 與您的應用程序集成

適用於 .NET 和 Xamarin 的 AWS 移動開發套件為使用 DynamoDB 提供了一種高階資料庫。您也可以直接針對低級 DynamoDB API 發出請求，但對於大多數用例，建議使用高級庫。所以此 AmazonDynamoDBClient 是高級庫中特別有用的部分。使用此類別，您可以執行各種建立、讀取、更新和刪除 (CRUD) 操作以及執行查詢。

適用於 .NET 和 Xamarin 的 AWS 移動開發套件可讓您使用適用 AWS SDK for .NET 中的 API 進行調用，以與 DynamoDB 配合使用。所有 API 都可以在AWSSDK.dll。如需適用於 .NET 的 AWS 開發套件的相關下載詳細資訊，請參適用於 .NET 的 AWS 開發套件。

您有三種方式可以與 Xamarin 應用程式中的 DynamoDB 互動：

- 文件模型：此 API 提供了圍繞低級 DyanMoDB API 的包裝類，以進一步簡化您的編程任務。表和文檔是關鍵的包裝類。您可以將文檔模型用於數據操作，例如創建、檢索、更新和刪除項目。API 可於使用。DocumentModel命名空間。
- 物件持久性模型：對象持久性 API 可讓您將用户端類別映射至 DynamoDB 資料表。然後每個物件執行個體會映射至相對應資料表中的某個項目。此 API 中的 DynamoDBContext 類提供了將客户端對象保存到表中、將項目作為對象檢索以及執行查詢和掃描的方法。您可以將對象持久化模型用於數據操作，例如創建、檢索、更新和刪除項目。您必須先使用服務客户端 API 建立資料表，然後使用對象持久性模型將類別映射至資料表。API 可於使用。DataModel命名空間。
- 服務客户端 API：這是與 DynamoDB API 緊密映射的協定層級 API。您可以將此低級 API 用於所有表和項目操作，例如創建、更新、刪除表和項目。您還可以查詢和掃描您的表。此 API 可在卓越亞馬遜命名空間中使用。

在以下主題中深入探討了這三種模式：

使用文件模型

文檔模型提供了圍繞低級別 .NET API 的包裝類。表和文檔是關鍵的包裝類。您可以使用文件模型來建立、檢索、更新和刪除項目。若要建立、更新和刪除資料表，您必須使用低階 API。如需如何使用低階 API 的資訊，請參使用 DynamoDB 服務級別 API。低階 API 在亞馬遜 DynamoDB 中提供。DocumentModel命名空間。

若要進一步了解文件模型，請參.NET 文件模型。

建立 DynamoDB 用户端

若要建立 DynamoDB 用户端，請執行下列

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

CRUD 操作

儲存項目

建立項目：

```
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";
```

將某個項目保存到 DynamoDB 資料表：

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

擷取項目

要檢索項目：

```
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);
}
```

更新項目

若要更新項目：

```
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);
}
```

若要更新項目，請執行以下操作：

```
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);
}
```

刪除項目

若要刪除項目：

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

查詢和掃描

要查詢和檢索作者為「馬克·吐温」的所有圖書，請執行以下操作：

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

下面的掃描示例代碼返回我們表格中的所有圖書：

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

使用物件持久性模型

適用於 .NET 和 Xamarin 的 AWS 移動開發套件提供物件持久性模型，讓您能夠將用户端類別映射至 DynamoDB 資料表。然後每個物件執行個體會映射至相對應資料表中的某個項目。若要將用户端物件保存至資料表，物件持久性模型會提供 DynamoDBContext 類別，即 DynamoDB 的入口點。此類別可讓您連線至 DynamoDB，繼而存取資料表、執行各種 CRUD 操作以及執行查詢。

物件持久性模型不提供用於建立、更新或刪除資料表的 API。它只提供資料操作。若要建立、更新和刪除資料表，您必須使用低階 API。如需如何使用低階 API 的資訊，請參使用 DynamoDB 服務級別 API。

概觀

物件持久性模型提供了一組屬性，可將用户端類別映射至資料表，並將屬性/欄位映射至資料表屬性。物件持久性模型支援類別屬性和資料表屬性之間的明確映射和預設映射。

- 明確映射：若要將屬性映射至主索引鍵，您必須使用 DynamoDBHashKey和 DynamoDBRangeKey 對象持久性模型屬性。此外，對於非主索引鍵屬性，如果類別中的屬性名稱和要對應的資料表屬性不相同，則必須明確新增 DynamoDBProperty ties 屬性來定義映射。
- 預設對應-預設情況下，物件持久性模型會將類別屬性映射至資料表中具有相同名稱的屬性。

您不必映射每個類別屬性。您可以新增 DynamoDBIgnore 屬性來識別這些屬性。保存和檢索對象的實例將忽略任何標記為此屬性的屬性。

支援的資料類型

物件持久性模型會支援一組基本的 .NET 資料類型、集合和任意資料類型。模型目前支援下列基本資料類型。

- bool
- 位元組
- char
- DateTime
- 十進制, 雙精度, 浮點
- 國際 32, 國際 64 國際
- 字節
- 字串
- UINT32, 聯合 64

對象持久化模型還支持具有以下限制的 .NET 集合類型：

- 集合類型必須實現 ICCollection 接口。
- 集合類型必須由受支持的基本體類型組成。例如，ICOLD 分析<string>，圖形切除<bool>。

- 集合類型必須提供一個無參數的構造函數。

如需物件持久性模型的詳細資訊，請參.NET 物件持久性模型。

建立 DynamoDB 用户端

建立 DynamoDB 用户端：

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

CRUD 操作

保存對象

建立物件：

```
[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
};
```

將物件保存至 DynamoDB 資料表：

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

索取物件

若要檢索物件：

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

更新物件

要更新對象：

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

刪除物件

若要刪除物件：

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

查詢和掃描

查詢和檢索作者是「查爾斯·狄更斯」的所有圖書，請執行以下操作：

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

下面的掃描示例代碼返回我們表格中的所有圖書：

```
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 服務級別 API

Dynamoo 服務級別 API 允許您建立、更新與刪除資料表。您還可以使用此 API 對資料表項目執行一般的建立、讀取、更新與刪除 (CRUD) 操作。

建立 DynamoDB 用户端

若要建立 DynamoDB 用户端：

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

CRUD 操作

儲存項目

若要將某個項目保存到 DynamoDB 資料表：

```
// 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);
```

擷取項目

要檢索項目：

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

更新項目

若要更新項目：

```
// 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);
```

刪除項目

刪除項目：

```
// 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);
```

查詢及掃描

要查詢和檢索作者為「馬克·吐溫」的所有圖書，請執行以下操作：

```
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);
    });

  }
}
```

下面的掃描示例代碼返回我們表格中的所有圖書：

```
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 和《適用 .NET 和 Xamarin 的 AWS 行動開發套件》，您可以編寫可接收行動推送通知的應用程式。如需有關 SNS 的詳細資訊，請參Amazon Simple Notification Service。

重要概念

Amazon SNS 允許不同設備上的應用程序和最終用户通過移動推送通知（蘋果、谷歌和 Kindle Fire 設備）、HTTP/HTTPS、電子郵件/電子郵件-JSON、短信或 Amazon Simple Queue Service (SQS) 隊列或 AWS Lambda 功能接收通知。SNS 允許您向訂閱單個主題的大量收件人發送單個郵件或扇出郵件。

主題

主題是一個「接入點」，允許收件人動態訂閱相同通知的相同副本。一個主題可以支持向多個終端節點類型的交付，例如，您可以將 iOS、Android 和 SMS 收件人組合在一起。

訂閱

若要接收發佈到主題的訊息，您必須訂閱端點至該主題。端點是行動應用程式、Web 伺服器、電子郵件地址或 Amazon SQS 隊列，用來接收來自 Amazon SNS 的通知訊息。一旦訂閱端點至主題並且確認訂閱，端點將會接收發佈到該主題的全部訊息。

發佈

發佈到主題時，SNS 會向該主題的每個訂閱者提供格式適當的郵件副本。對於移動推送通知，您可以直接發佈到終端節點或將終端節點訂閱主題。

項目設定

先決條件

要在應用程序中使用 SNS，您需要將 SDK 添加到您的項目中。若要執行此作業，請按照設定適用 .NET 和 Xamarin 的 AWS 行動開發套件。

設置 SNS 的權限

如需設定 SNS 許可的資訊，請參管理 Amazon SNS 主題的存取。

AddNuGetSNS 軟件包到您的項目

請按照中的說明步驟 4設定適用 .NET 和 Xamarin 的 AWS 行動開發套件添加 Amazon Simple Notification ServiceNuGet軟件包添加到您的項目。

整合 SNS 與您的應用程式

有多種方式可以在 Xamarin 應用程式中與 SNS 進行交互：

發送推送通知 (Xamarin 安卓)

本文檔介紹如何使用 Amazon Simple Notification Service (SNS) 和適用於 .NET 和 Xamarin 的 AWS 移動軟件開發工具包向 Xamarin Android 應用程序發送推送通知。

專案設定

先決條件

您必須完成設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK，然後再開始本教程。

設定 SNS 的許可

按照設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK將下面提到的策略附加到應用程序的角色。這將為您的應用程序授予訪問 SNS 的適當權限：

1. 前往IAM 主控台並選取您想要配定的 IAM 角色。
2. 按一下連接政策，請選取卓越亞馬遜 SNFullAccess策略，然後單擊連接政策。

> **Warning**
>
> 使用卓越亞馬遜 SNFullAccess不建議在生產環境中使用。我們使用此處可讓您快速啟動並執行。如需指定 IAM 角色的許可的詳細資訊，請參IAM 角色許可概觀。

在谷歌雲端啟用推送通知

首先，添加一個新的谷歌 API 項目：

1. 前往Google 開發人員主控台。
2. 按一下建立專案。
3. 在 中新專案框中，輸入項目名稱，記下項目 ID（稍後需要），然後單擊建立。

然後，啟用適用於您的專案的 Google 雲訊息 (GCM) 服務：

1. 在 中Google 開發人員主控台，您的新專案應該已選取起來。如果沒有，請在頁面頂端的下拉式菜單中選取它。
2. 選擇API & 驗證從頁面左側側欄中輸入。
3. 在搜尋方塊中，輸入適用於 Android 的 Google 雲訊息傳送，然後按一下適用於安卓的谷歌雲消息鏈接。
4. 按一下啟用 API。

最後，獲取 API 密鑰：

1. 在谷歌開發人員控制台中，選擇API & 驗證 >登入資料。
2. 根據Public API 存取，請單擊建立新的金鑰。
3. 在 中建立新的金鑰對話框中，單擊服務器密鑰。
4. 在生成的對話框中，單擊建立並複製顯示的 API 密鑰。您稍後將使用此 API 密鑰執行身份驗證。

使用項目 ID 在 SNS 控制台中創建平台 ARN

1. 前往SNS 控制台。
2. 按一下應用程式在畫面左側。
3. 按一下建立平台應用程式以建立新 SNS 平台應用程式。
4. 輸入應用程式名稱。
5. 選擇Google 雲訊息傳遞 (GCM)為了推送通知平台。
6. 將 API 密鑰粘貼至標記為API 金鑰。
7. 按一下建立平台應用程式。
8. 選擇剛剛創建的平台應用程序並複製應用程序 ARN。

AddNuGetSNS 軟件包到您的項目

請按照中的說明步驟 4設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK添加 Amazon Simple Notification ServiceNuGet軟件包添加至您的專案。

建立 SNS 客户端

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

註冊應用程序以獲得遠程通知

若要在 Android 上註冊遠程通知，您需要建立BroadcastReceiver，它可以接收谷歌雲消息。在提示時更改下面的軟件包名稱：

```
[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);
      }
}
```

以下是接收來自BroadcastReceiver並在設備的通知欄上顯示通知：

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

從 SNS 控制台向終端節點發送消息

1. 前往SNS 控制台 > 應用程序。
2. 選擇您的平台應用程序，選擇終端節點，然後單擊發佈至終端。
3. 在文字方塊中輸入文字信息，然後單擊發佈訊息以發佈訊息。

發送推送通知 (Xamarin iOS)

本文檔介紹如何使用 Amazon Simple Notification Service (SNS) 和適用於 .NET 和 Xamarin 的 AWS 移動軟件開發工具包向 Xamarin iOS 應用程序發送推送通知。

項目設定

先決條件

您必須完成設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK，然後再開始本教程。

設置 SNS 的權限

按照步驟 2設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK將下面提到的策略附加到應用程序的角色。這將為您的應用程序授予訪問 SNS 的適當權限：

1. 前往IAM 主控台並選取您想要設定的 IAM 角色。
2. 按一下連接政策下，選擇卓越亞馬遜 SNFullAccess策略，然後單擊連接政策。

> **Warning**
>
> 使用卓越亞馬遜 SNFullAccess不建議在生產環境中使用。我們在此使用它來幫助您快速啟動並執行。如需有關指定 IAM 角色權限的詳細資訊，請參IAM 角色權限概觀。

獲得蘋果 iOS 開發者計劃的會員資格

您需要在物理設備上運行應用程序才能接收推送通知。要在設備上運行應用程序，您必須在蘋果 iOS 開發者計劃會員。擁有成員資格後，您可以使用 Xcode 來生成簽名標識。如需詳細資訊，請參 Apple 的應用程式分發快速入門文件中)。

創建 iOS 證書

首先，您需要建立 iOS 憑證。然後，您需要創建一個配置為推送通知配置的置備配置文件。作法：

1. 前往蘋果開發者會員中心，請按一下證書、標識符和配置文件。
2. 按一下識別碼下iOS 應用程式，請單擊網頁右上角的加號按鈕以添加新的 iOS 應用程序 ID，然後輸入應用 ID 描述。
3. 向下捲動至新增 ID 後置詞部分，然後選擇明確應用程式 ID並輸入捆綁標識符。
4. 向下捲動至應用程式服務部分，然後選擇推送通知。
5. 按一下 Continue (繼續)。
6. 請按 Submit (提交)。
7. 按一下完成。
8. 選擇剛剛創建的應用程序 ID，然後單擊Edit (編輯)。
9. 向下捲動至推送通知區段。按一下建立憑證下開發 SSL 憑證。
10. 按照說明創建證書簽名請求 (CSR)、上傳請求並下載將用於與 Apple 通知服務 (APNS) 通信的 SSL 證書。
11. 返回證書、標識符和配置文件(憑證已建立！) 頁面上的名稱有些許差異。按一下All (全部)下設定檔。
12. 單擊右上角的加號按鈕以添加新的配置文件。
13. 選擇iOS 應用程式開發，然後按一下Continue。
14. 選取您的應用程式 ID，然後按一下Continue。
15. 選取您的開發者憑證，然後按一下Continue。
16. 選取您的設備，然後按一下Continue。
17. 輸入配置式名稱，然後按一下Generate。
18. 下載並雙擊置備文件以安裝置配置文件。

有關置備為推送通知配置的配置文件的詳細信息，請參閱 Apple 的配置推送通知文件中)。

使用證書在 SNS 控制台中創建平台 ARN

1. 執行KeyChain訪問應用程序，選擇我的憑證，然後右鍵單擊您生成的 SSL 證書以連接到 APNS，然後選擇匯出。系統將提示您指定文件的名稱和密碼以保護證書。證書將保存在 P12 文件中。
2. 前往SNS 控制台，然後按一下應用程式在畫面左側。
3. 按一下建立平台應用程式來建立新 SNS 平台應用程式。
4. 輸入應用程式名稱。
5. 選擇Apple 開發為了推播通知平台。
6. 按一下選取檔案，然後選擇導出 SSL 證書時創建的 P12 文件。
7. 輸入您導出 SSL 證書時指定的密碼，然後單擊從檔案載入憑證。
8. 按一下建立平台應用程式。
9. 選擇剛剛創建的平台應用程序並複製應用程序 ARN。您將在接下來的步驟中需要此資訊。

AddNuGetSNS 軟件包到您的項目

請按照中的說明步驟 4設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK新增 Amazon Simple Notification ServiceNuGet軟件包添加到您的項目。

建立 SNS 客户端

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

註冊應用程序以獲得遠程通知

要註冊應用程序，請調用RegisterForRemoteNotifications，如下所示。將下面的代碼放在AppDelegate.cs，在下面提示的位置插入您的平台應用程序 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 */
    });
  }
}
```

從 SNS 控制台向終端節點發送消息

1. 前往SNS 控制台 > 應用程序。
2. 選擇您的平台應用程序，選擇終端節點，然後單擊發佈至終端。
3. 在文字框中輸入短信，然後按一下發佈訊息來發佈訊息。

發送和接收短信通知

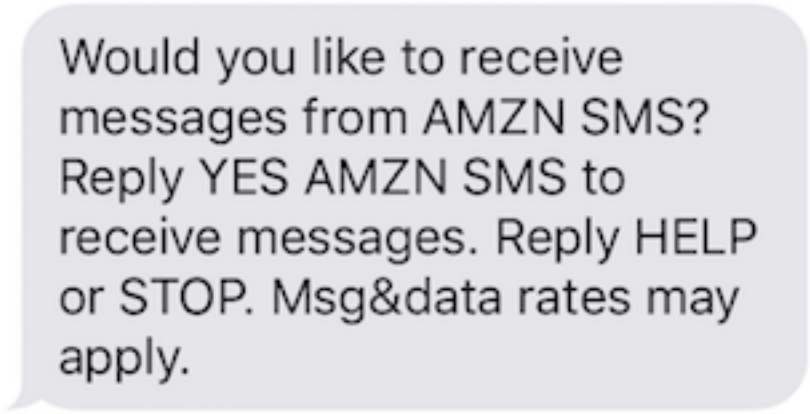
您可以使用 Amazon SMS 通知服務 (Amazon SNS) 將 SMS 通知傳送到啟用 SMS 的行動電話和智慧型手機。

> **Note**
>
> 美國的電話號碼目前支援 SMS 通知。SMS 訊息只能從美國東部 (維吉尼亞北部) 區域中建立的主題中發送。但是，您可以從任何其他區域將訊息發佈到您在美國東部 (維吉尼亞北部) 區域中建立的主題。

建立主題

建立主題：

1. 在 Amazon SNS 主控台中，單擊建立新主題。此時將出現建立新主題對話方塊。
2. 在 Topic name (主題名稱) 方塊中，輸入主題名稱。
3. 在「顯示名稱」框中，鍵入顯示名稱。主題必須具有為其分配的顯示名稱，因為顯示名稱的前十 (10) 個字符將用作文本消息前綴的初始部分。您輸入的顯示名稱將出現在 SNS 發送給用户的確認消息中（下面的顯示名稱為「AMZN SMS」）。



1. 按一下 Create topic (建立主題)。新主題顯示在 Topics (主題) 頁面上。
2. 選取新主題，然後按一下主題 ARN。出現 Topic Details (主題詳細資訊) 頁面。
3. 複製主題 ARN，因為您在下一步中訂閱主題時將需要它。

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

使用 SMS 協議訂主題

創建 SNS 客户端，傳遞您的憑據對象和身份池的區域：

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

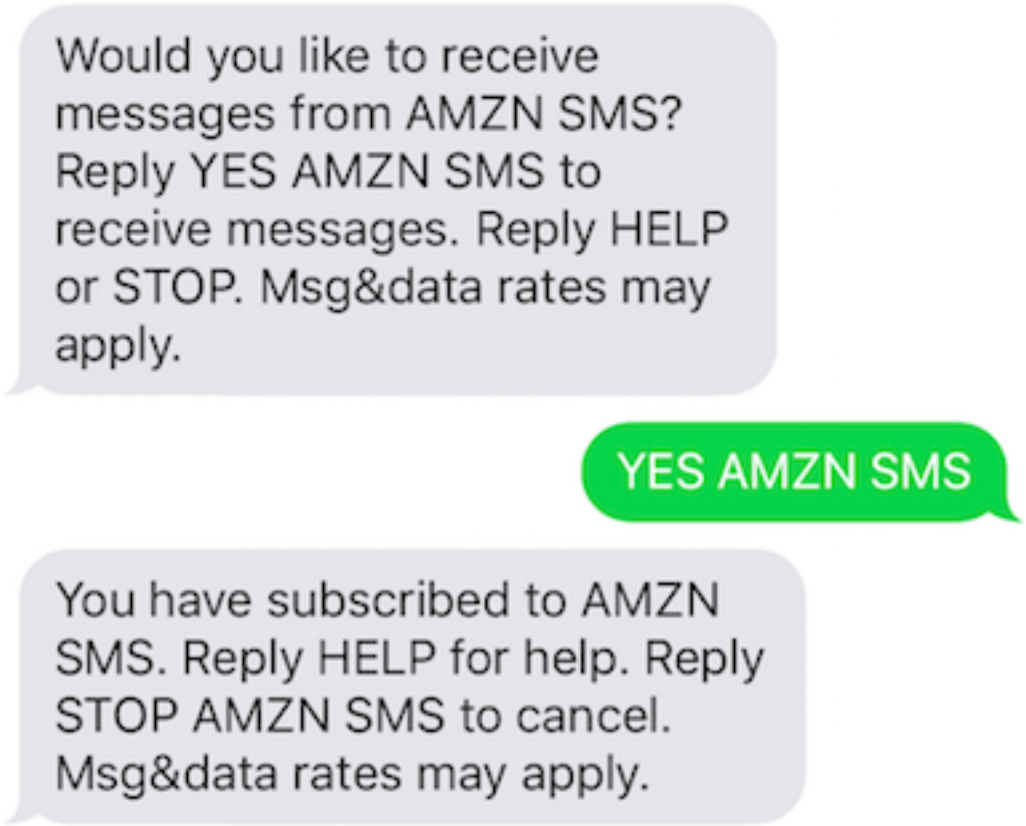
若要訂主題，請調用SubscribeAsync並將您想訂閱的主題的 ARN、協議（「SMS」）和電話號碼傳遞給它：

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

您將在訂閱響應對象中收到一個訂閱 arn。您的訂如下所示：

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

當設備訂閱主題時，SNS 將向設備發送確認消息，用户必須確認他們希望接收通知，如下所示：



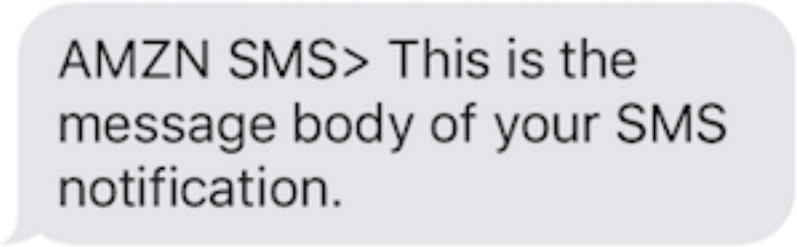
用户訂閱主題後，當您將 SMS 消息發佈到該主題時，他們將收到 SMS 消息。

發佈訊息

將訊息發佈至主題：

1. 登入 AWS 管理主控台並開啟Amazon SNS 主控台。
2. 在左導覽窗格中，按一下 Topics (主題)，然後選取您想要發佈的主題。
3. 按一下發佈到主題。
4. 在主題方塊中，輸入主題。

5. 在 Message (訊息) 方塊內，輸入訊息。Amazon SNS 會將您在「消息」框中輸入的文本發送給 SMS 訂閱者，除非您同時在「主題」框中輸入文本。由於 Amazon SNS 包含您發送的所有 SMS 消息的顯示名稱前綴，因此顯示名稱前綴和消息負載的總和不能超過 140 個 ASCII 字符或 70 個 Unicode 字符。Amazon SNS 會截斷超過這些限制的郵件。
6. 按一下 Publish message (發佈訊息)。Amazon SNS 會顯示確認對話方塊。SMS 消息將顯示在啟用短信的設備上，如下所示。



傳送訊息至 HTTP/HTTPS 端點

您可使用 Amazon SNS 將通知訊息傳送至一個或多個 HTTP 或 HTTPS 端點。程序如下：

1. 設定端點以接收 Amazon SNS 訊息。
2. 讓 HTTP/HTTPS 端點訂主題
3. 確認您的訂。
4. 發佈通知到主題 Amazon SNS 會傳送 HTTP POST 請求以傳遞通知內容到訂的端點。

配置您的 HTTP/HTTPS 終端節點以接收 Amazon SNS 消息

請按照傳送 Amazon SNS 訊息至 HTTP/HTTPS 端點來配置端點。

訂 HTTP/HTTPS 端點至您的 Amazon SNS 主題

創建 SNS 客户端，傳遞您的憑據對象和身份池的區域：

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

若要透過主題傳送訊息至 HTTP 或 HTTPS 端點，您必須訂閱端點至 Amazon SNS 主題。您可以使用其 URL 來指定端點：

```
var response = await snsClient.SubscribeAsync(
```

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

確認您的 訂閱

在您訂端點之後，Amazon SNS 會傳送訂確認訊息至端點。端點上的代碼必須檢索SubscribeURL值，然後訪問SubscribeURL本身或將其提供給您，以便您可以手動訪問SubscribeURL（例如，如果使用 Web 瀏覽器）。

Amazon SNS 不會傳送訊息至端點，除非已經確認訂。當您造訪 SubscribeURL 時，回應將會包含 XML 文件，其中含有為訂閱指定 ARN 的元素 SubscriptionArn。

傳送訊息至 HTTP/HTTPS 端點

您可以通過發佈至主題，來傳送訊息至主題的訂。呼叫PublishAsync並將其傳遞給主題 ARN 和您的消息。

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

SNS 故障診斷

在 Amazon SNS 控制台中使用配送狀態

Amazon SNS 控制台包含一個配送狀態功能，您可以收集有關郵件成功和失敗嘗試發送到移動推送通知平台（蘋果 (APNS)、谷歌 (GCM)、亞馬遜 (ADM)、Windows（WNS 和 MPNS）和百度的反饋。

它還提供了其他重要信息，如 Amazon SNS 中的停留時間。此信息在亞馬遜CloudWatch當通過 Amazon SNS 控制台或 Amazon SNS API 啟用此功能時，Amazon SNS 自動創建的日誌組。

如需使用「配送狀態」功能的說明，請參「」使用 Amazon SNS 的配送狀態功能在 AWS Mobile Bobile Bobile Bobile

使用適用 .NET 和 Xamarin 的 AWS Mobile SDK 的最佳實務

使用適用於 .NET 和 Xamarin 的 AWS 移動軟件開發工具包時，只有少數幾個基本知識和最佳實踐需要瞭解。

- 使用 Amazon Cognito 獲取 AWS 證書，而不是在應用程序中對您的證書進行硬編碼。如果您在應用程序中對證書進行硬編碼，則最終可能會向公眾公開證書，從而允許其他人使用您的證書調用 AWS。如需如何使用 Amazon Cognito 獲取 AWS 登入資料的說明，請參設定適用 .NET 和 Xamarin 的 AWS 行動開發套件。
- 如需使用 S3 的最佳實務，請參AWS 博客上的這篇文章。
- 如需使用 DynamoDB 的最佳實務，請參DynamoDB 最佳實務（在 DynamoDB 開發人員指南中）。

我們一直希望幫助我們的客户取得成功並歡迎反饋，所以請隨時在 AWS 論壇上發佈帖子或者打開一個問題GitHub。

AWS 服務文檔庫

適用於 .NET 和 Xamarin 的 AWS 移動開發工具包中的每個服務都有單獨的開發人員指南和服務 API 參考，其中提供了您可能會發現有幫助的其他信息。

Amazon Cognito 身分

- Cognito 開發人員指南
- Cognito 份服務 API 參考

Amazon Cognito Sync

- Cognito 開發人員指南
- Cognito 同步服務 API 參考

Amazon Mobile Analytics

- Mobile Analytics 開發人員指南
- Mobile Analytics 服務 API 參考

Simple Storage Service (Amazon S3)

- S3 開發人員指南
- S3 入門指南
- S3 服務 API 參考

Amazon DynamoDB

- DynamoDB 開發人員指南
- DynamoDB 入門指南
- DynamoDB 服務 API 參考

Amazon Side Notification Service (SNS)

- SNS 開發人員指南
- SNS 服務 API 參考

其他有用鏈接

- AWS 詞彙表
- 關於 AWS 登入資料

故障診斷

本主題介紹如何解決使用適用 .NET 和 Xamarin 的 AWS 行動開發套件時可能遇到的問題。

確保 IAM 角色具有所需的權限

調用 AWS 服務時，您的應用程序應使用 Cognito 身份池中的身份。池中的每個身份都與 IAM（Identity and Access Management）角色相關聯。

角色具有一個或多個與其關聯的策略文件，用於指定分配給該角色的用户有權訪問哪些 AWS 資源。默認情況下，每個身分集區會創建兩個角色：一個用於通過驗證的使用者，另一個用於未驗證的使用者。

您需要修改現有策略文件，或者將新策略文件與應用程序所需的權限相關聯。如果您的應用程序允許經過身份驗證的用户和未經身份驗證的用户，則必須向這兩個角色授予訪問您的應用程序所需的 AWS 資源的權限。

以下政策文件説明如何授予 S3 儲存儲體：

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

以下策略文件顯示瞭如何授予對 DynamoDB 數據庫的訪問權限：

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

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

如需政策的詳細資訊，請參IAM 政策。

使用 HTTP 代理調試器

如果您的應用程序調用的 AWS 服務具有 HTTP 或 HTTPS 終端節點，則可以使用 HTTP/HTTPS 代理調試器查看請求和響應，以便更深入地瞭解正在發生的事情。有許多 HTTP 代理調試器可用，例如：

- 查爾斯-適用於 Windows 和 OSX 的網頁調試代理
- 提琴手-一個網絡調試代理軟件

查爾斯和 Fiddler 都需要一些配置才能查看 SSL 加密流量，請閱讀這些工具的文檔以瞭解更多信息。如果您使用的 Web 調試代理無法配置為顯示加密流量，請打開 aws_endpoint _json 文件，並將需要調試的 AWS 服務的 HTTP 標籤設置為 true。

文件歷史記錄

下表說明自上次發行適用於 .NET 和 Xamarin 的 AWS 移動開發工具包之後，文件的重要變更。

- API 版本：2015-08-27
- 上次文件更新：2021-02-23

變更	API 版本	描述	發行日期
已封存	2015-08-27	所以此AWS適用於 Xamarin 的移動 SDK 包含在AWS SDK for .NET。本指南引用 Xamarin 移動 SDK 的歸檔版本。	2021-02-23
GA 版本	2015-08-27	GA 版本	2015-08-27
試用版	2015-07-28	試用版	rn2015-07-28