Xamarin 開發人員指南

AWS Mobile SDK

AWS Mobile SDK: Xamarin 開發人員指南

Table of Contents

適用於 Xamarin 的 AWS Mobile SDK 現在已包含在 中 適用於 .NET 的 AWS SDK。本指南參考 Mobile SDK for Xamarin 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。

什麼是適用於 .NET 和 Xamarin 的 AWS Mobile SDK？

適用於 Xamarin 的 AWS Mobile SDK 包含在 中 適用於 .NET 的 SDK。如需詳細資訊，請參閱《適用於 .NET 的 AWS SDK 開發人員指南》。

本指南不再更新，參考適用於 Xamarin 的 Mobile SDK 封存版本。

相關指南和主題

- 對於前端和行動應用程式開發，建議使用 AWS Amplify。
- 如需 適用於 .NET 的 AWS SDK 針對 Xamarin 應用程式使用 的特殊考量，請參閱《 適用於 .NET 的 AWS SDK 開發人員指南》中的 Xamarin 支援的特殊考量。
- 基於參考目的，您可以在 GitHub 上找到AWS 適用於 Xamarin 的 Mobile SDK 封存版本。

封存的參考內容

適用於 .NET 和 Xamarin 的封存 AWS Mobile SDK 提供一組 .NET 程式庫、程式碼範例和文件，協助開發人員建置連線的行動應用程式：

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

使用適用於 .NET 和 Xamarin 的 AWS Mobile SDK 撰寫的行動應用程式會呼叫原生平台 APIs，使其具有原生應用程式的外觀和風格。SDK 中的 .NET 程式庫提供圍繞 AWS REST APIs C# 包裝函式。

適用於 .NET 和 Xamarin 的 AWS Mobile SDK 包含哪些內容？

支援的 AWS 服務目前包括但不限於：

- Amazon Cognito
- 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，以及收集和分析用量資料。

適用於 .NET 和 Xamarin 的 AWS Mobile SDK 也可讓您使用適用於 .NET 的 AWS 開發套件支援的大部分 AWS 服務。本開發人員指南會說明行動開發特有的 AWS 服務。如需適用於 .NET 的 AWS 開發套件的詳細資訊，請參閱：

- 適用於 .NET 的 AWS 開發套件入門指南
- 適用於 .NET 開發人員的 AWS 開發套件
- 適用於 .NET 的 AWS 開發套件 API 參考

相容性

適用於 .NET 和 Xamarin 的 AWS Mobile SDK 以可攜式類別程式庫 (PCL) 的形式提供。PCL Support 已在 Xamarin.Android 4.10.1 和 Xamarin.iOS 7.0.4 中新增。可攜式程式庫專案內建於 Visual Studio。

IDE

如需搭配 Xamarin SDK 封存版本使用 IDEs 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 和 Xamarin 的 AWS Mobile SDK？

若要取得適用於 .NET 和 Xamarin 的 AWS Mobile SDK，請參閱設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK。適用於 .NET 和 Xamarin 的 AWS Mobile SDK 以 NuGet 套件形式發佈。您可以在 NuGet 上的 AWS 開發套件套件或適用於 .NET GitHub 儲存庫的 AWS 開發套件中找到 AWS 服務套件的完整清單。

關於 AWS Mobile Services

Amazon Cognito 身分

對 AWS 進行的所有呼叫都需要 AWS 登入資料。建議您使用 Amazon Cognito Identity 為您的應用程式提供 AWS 登入資料，而不是將登入資料硬式編碼到應用程式中。請遵循設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的指示，透過 Amazon Cognito 取得 AWS 登入資料。

Cognito 也可讓您使用 Amazon、Facebook、Twitter 和 Google 等公有登入供應商，以及支援 OpenID Connect 的供應商來驗證使用者。Cognito 也適用於未經驗證的使用者。Cognito 提供臨時登入資料，具有您使用 Identity and Access Management (IAM) 角色指定的有限存取權限。Cognito 是透過建立與 IAM 角色相關聯的身分集區來設定。IAM 角色指定您的應用程式可以存取的資源/服務。

若要開始使用 Cognito Identity，請參閱設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK。

若要進一步了解 Cognito Identity，請參閱 Amazon Cognito Identity。

Amazon Cognito Sync

Cognito Sync 是一種 AWS 服務和用戶端程式庫，可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 來同步跨裝置和跨登入提供者的使用者設定檔資料 - Amazon、Facebook、Google 和您自己的自訂身分提供者。

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

如需 Cognito Sync 的詳細資訊，請參閱 Amazon Cognito Sync。

Mobile Analytics

Amazon Mobile Analytics 可讓您收集、視覺化和了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、應用程式內收入和自訂事件的指標，並可依平台和日期範圍篩選。Amazon Mobile Analytics 專為擴展您的業務而建置，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 追蹤應用程式用量資料。

如需 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 會移除資料儲存體的傳統可擴展性限制，同時維持低延遲和可預測的效能。

若要開始使用 Dynamo DB，請參閱使用 DynamoDB 存放和擷取資料。

如需 Dynamo DB的詳細資訊，請參閱 Amazon DynamoDB。

Amazon Simple Notification Service

Amazon Simple Notification Service (SNS) 是一種快速、靈活、全受管的推播通知服務，可讓您傳送個別訊息或散發訊息給大量收件人。Amazon Simple Notification Service 可讓您以簡單且經濟實惠的方式，將推播通知傳送給行動裝置使用者、電子郵件收件人，甚至傳送訊息至其他分散式服務。

若要開始使用 SNS for Xamarin iOS，請參閱使用 SNS (Xamarin iOS) 接收推播通知。

若要開始使用 SNS for Xamarin Android，請參閱使用 SNS (Xamarin Android) 接收推播通知。

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

您可以設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK，並開始建立新的專案，也可以將 SDK 與現有專案整合。您也可以複製並執行範例，以了解 SDK 的運作方式。請依照下列步驟設定並開始使用適用於 .NET 和 Xamarin 的 AWS Mobile SDK。

先決條件

您必須先執行下列動作，才能使用適用於 .NET 和 Xamarin 的 AWS Mobile SDK：

- 建立 AWS 帳戶。
- 安裝 Xamarin。

完成先決條件之後：

1. 使用 Amazon Cognito 取得 AWS 登入資料。
2. 為您將在應用程式中使用的每個 AWS 服務設定必要的許可。
3. 在 IDE 中建立新的專案。
4. 安裝適用於 .NET 和 Xamarin 的 AWS Mobile SDK。
5. 設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK。

步驟 1：取得 AWS 登入資料

若要在應用程式中呼叫 AWS，您必須先取得 AWS 登入資料。您可以使用 Amazon Cognito，這項 AWS 服務可讓您的應用程式存取 SDK 中的服務，而不必在應用程式中內嵌私有 AWS 登入資料。

若要開始使用 Amazon Cognito，您需要建立身分集區。身分集區是專屬於您 帳戶的資訊存放區，由如下所示的唯一身分集區 ID 識別：

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

1. 登入 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，然後選取核取方塊以啟用對未驗證身分的存取。選擇建立集區以建立您的身分集區。

3. 選擇允許以建立與身分集區相關聯的兩個預設角色，一個用於未驗證的使用者，另一個用於已驗證的使用者。這些預設角色可讓您存取 Amazon Cognito Sync 和 Amazon Mobile Analytics 的身分集區。

一般而言，每個應用程式只會使用一個身分集區。

建立身分集區後，您可以建立CognitoAWSCredentials物件（將身分集區 ID 傳遞給該物件）並將它傳遞給 AWS 用戶端的建構函式，以取得 AWS 登入資料，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

您需要為要在應用程式中使用的每個 AWS 服務設定許可。首先，您需要了解 AWS 如何檢視您應用程式的使用者。

當有人使用您的應用程式並呼叫 AWS 時，AWS 會指派身分給該使用者。您在步驟 1 中建立的身分集區是 AWS 存放這些身分的位置。身分有兩種類型：已驗證和未驗證。已驗證的身分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使用者。

每個身分都與 AWS Identity and Access Management 角色相關聯。在步驟 1 中，您建立了兩個 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 角色都有一個以上的附加政策，指定指派給該角色的身分可以存取哪些 AWS 服務。例如，下列範例政策會授予 Amazon S3 儲存貯體的存取權：

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

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

若要設定要在應用程式中使用的 AWS 服務許可，請修改連接到角色的政策。

1. 前往 IAM 主控台，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。如果您的應用程式允許已驗證和未驗證的使用者，您需要授予這兩個角色的許可。
2. 按一下連接政策，選取所需的政策，然後按一下連接政策。您建立之 IAM 角色的預設政策可讓您存取 Amazon Cognito Sync 和 Mobile Analytics。

如需建立政策或從現有政策清單中選擇的詳細資訊，請參閱 IAM 政策。

步驟 3：建立新的 專案

Windows

您可以使用 Visual Studio 來開發應用程式。

OS X

您可以使用 Visual Studio 來開發應用程式。使用 Xamarin 的 iOS 開發需要存取 Mac 才能執行應用程式。如需詳細資訊，請參閱在 Windows 上安裝 Xamarin.iOS。

> **Note**
>
> JetBrains 的跨平台商業 IDE Rider 在 Windows 和 Mac 平台上都包含 Xamarin 支援。

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

Windows

選項 1：使用 Package Manager 主控台安裝

適用於 .NET 和 Xamarin 的 AWS Mobile SDK 包含一組 .NET 組件。若要安裝適用於 .NET 和 Xamarin 的 AWS Mobile SDK，請在 Package Manager 主控台中為每個套件執行 install-package 命令。例如，若要安裝 Cognito Identity，請執行下列動作：

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

所有專案都需要 AWS Core Runtime 和 Amazon Cognito Identity 套件。以下是每個服務的完整套件名稱清單。

服務	套件名稱
AWS Core 執行期	AWSSDK.Core
Amazon Cognito Sync	AWSSDK.CognitoSync
Amazon Cognito 身分	AWSSDK.CognitoIdentity
Amazon DynamoDB	AWSSDK.DynamoDBv2
Amazon Mobile Analytics	AWSSDK.MobileAnalytics
Amazon S3	AWSSDK.S3
Amazon SNS	AWSSDK.SimpleNotificationService

若要包含發行前套件，請在安裝套件時包含`-Pre`命令列引數，如下所示：

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

您可以在 NuGet 上的 AWS 開發套件套件或適用於 .NET GitHub 儲存庫的 AWS 開發套件中找到 AWS 服務套件的完整清單。

選項 2：使用您的 IDE 安裝

在 Visual Studio 中

1. 在專案上按一下滑鼠右鍵，然後按一下管理 NuGet 套件。
2. 搜尋您要新增至專案的套件名稱。若要包含 prelease NuGet 套件，請選擇包含 Prelease。您可以在 NuGet 的 AWS 開發套件套件中找到 AWS 服務套件的完整清單。
3. 選擇套件，然後選擇 Install (安裝)。

Mac (OS X)

在 Visual Studio 中

1. 在套件資料夾上按一下滑鼠右鍵，然後選擇新增套件。
2. 搜尋您要新增至專案的套件名稱。若要包含 prelease NuGet 套件，請選擇顯示發行前套件。您可以在 NuGet 的 AWS 開發套件套件中找到 AWS 服務套件的完整清單。
3. 選取您想要的套件旁的核取方塊，然後選擇新增套件。

> **Important**
>
> 如果您使用可攜式類別程式庫進行開發，您還必須將 AWSSDK.Core NuGet 套件新增至衍生自可攜式類別程式庫的所有專案。

步驟 5：設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK

設定記錄

您可以使用 `Amazon.AWSConfigs`類別和 `Amazon.Util.LoggingConfig`類別來設定記錄設定。您可以在`AWSSdk.Core`組件中找到這些項目，可透過 Visual Studio 中的 Nuget Package Manager 取得。您可以在 Android 應用程式的 `MainActivity.cs` 檔案中，或 iOS 應用程式的 `AppDelegate.cs`檔案中，將記錄設定程式碼放在 `OnCreate` 方法中。您也應該將 `using Amazon`和 `using Amazon.Util`陳述式新增至 .cs 檔案。

設定記錄設定，如下所示：

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

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

當您登入 SystemDiagnostics 時，架構會在內部將輸出列印至 System.Console。如果您想要記錄 HTTP 回應，請設定 LogResponses旗標。這些值可以是 Always、Never 或 OnError。

您也可以使用 LogMetrics 屬性記錄 HTTP 請求的效能指標。您可以使用 LogMetricsFormat 屬性來指定日誌格式。有效值為 JSON 或標準。

設定區域端點

設定所有服務用戶端的預設區域，如下所示：

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

這會設定 SDK 中所有服務用戶端的預設區域。您可以在建立服務用戶端執行個體時明確指定區域，以覆寫此設定，如下所示：

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

設定 HTTP Proxy 設定

如果您的網路位於代理之後，您可以設定 HTTP 請求的代理設定，如下所示。

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

修正時鐘偏移

此屬性會判斷正確的伺服器時間，並以正確的時間重新發出請求，以判斷 SDK 是否應該修正用戶端時鐘扭曲。

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

如果服務呼叫導致例外狀況，且 SDK 判斷本機和伺服器時間之間存在差異，則會設定此欄位。

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

若要進一步了解時鐘偏斜，請參閱 AWS 部落格上的時鐘偏斜修正。

後續步驟

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

- 開始使用。閱讀適用於 .NET 和 Xamarin 的 AWS Mobile SDK 入門，了解如何在適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中使用和設定服務的快速入門說明。
- 探索服務主題。了解適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的每個服務及其運作方式。
- 執行示範。檢視示範常見使用案例的 Xamarin 應用程式範例。若要執行範例應用程式，請如先前所述設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK，然後遵循個別範例的 README 檔案中包含的指示。
- 了解 APIs。檢視 |sdk-xamarin-ref|_。
- 提問：在 AWS Mobile SDK 論壇上張貼問題或在 GitHub 上開啟問題。

適用於 .NET 和 Xamarin 的 AWS Mobile SDK 入門

適用於 .NET 和 Xamarin 的 AWS Mobile SDK 提供從 Xamarin 應用程式呼叫 AWS 服務所需的程式庫、範例和文件。

您必須先完成設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的所有說明，才能開始使用下列服務。

這些入門主題將逐步引導您完成下列各項：

主題

- 使用 Amazon S3 存放和擷取檔案
- 使用 Cognito Sync 同步使用者資料
- 使用 DynamoDB 存放和擷取資料
- 使用 Amazon Mobile Analytics 追蹤應用程式用量資料
- 使用 SNS (Xamarin iOS) 接收推播通知
- 使用 SNS (Xamarin Android) 接收推播通知

如需其他 AWS Mobile SDKs 的資訊，請參閱 AWS Mobile SDK。

使用 Amazon S3 存放和擷取檔案

Amazon Simple Storage Service (Amazon S3) 為行動開發人員提供安全、耐用、高度可擴展的物件儲存。Amazon S3 易於使用，具有簡單的 Web 服務界面，可從 Web 上的任何位置存放和擷取任意數量的資料。

以下教學課程說明如何整合 S3 TransferUtility，這是搭配您的應用程式使用 S3 的高階公用程式。如需從 Xamarin 應用程式使用 S3 的詳細資訊，請參閱 Amazon Simple Storage Service (S3)。

專案設定

先決條件

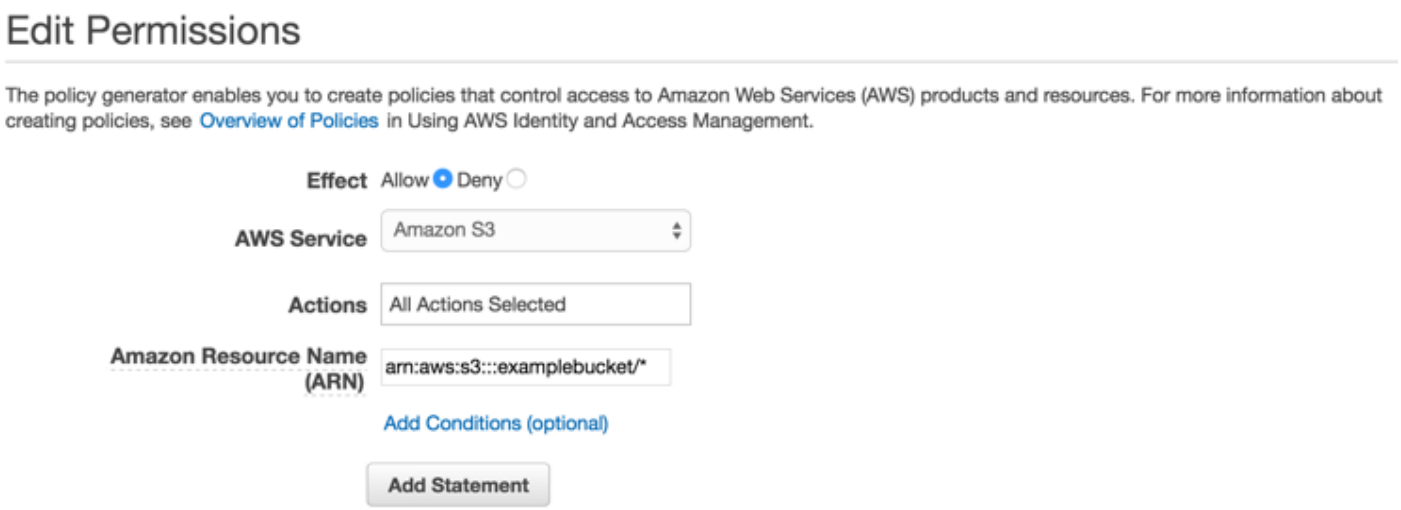
開始本教學課程之前，您必須完成設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 上的所有說明。

本教學課程也假設您已建立 S3 儲存貯體。若要建立 S3 儲存貯體，請造訪 S3 AWS 主控台。

設定 S3 的許可

預設 IAM 角色政策會授予應用程式對 Amazon Mobile Analytics 和 Amazon Cognito Sync 的存取權。若要讓您的 Cognito 身分集區存取 Amazon S3，您必須修改身分集區的角色。

1. 前往 Identity and Access Management Console，然後按一下左側窗格中的角色。
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. 按一下未驗證使用者的角色（身分集區名稱將附加不驗證）。
4. 按一下建立角色政策，選取政策產生器，然後按一下選取。
5. 在編輯許可頁面上，輸入下圖中顯示的設定，以您自己的名稱取代 Amazon Resource Name (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的名稱組成。以下顯示的設定會讓您的身分集區完整存取指定儲存貯體的所有動作。



1. 按一下新增陳述式按鈕，然後按一下下一步。
2. 精靈會顯示您產生的組態。按一下套用政策。

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

將 S3 的 NuGet 套件新增至您的專案

請遵循設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的步驟 4，將 S3 NuGet 套件新增至您的專案。

（選用）設定 S3 請求的簽章版本

與 Amazon S3 的每次互動，可以經過驗證身分或是匿名進行。AWS 使用 Signature 第 4 版或 Signature 第 2 版演算法來驗證對服務的呼叫。

2014 年 1 月之後建立的所有新 AWS 區域僅支援 Signature 第 4 版。不過，許多較舊的區域仍然支援 Signature 第 4 版和 Signature 第 2 版請求。

如果您的儲存貯體位於此頁面列出的其中一個不支援 Signature 第 2 版請求的區域，您必須將 AWSConfigsS3.UseSignatureVersion4 屬性設定為「true」，如下所示：

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

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

初始化 S3 TransferUtility 用戶端

建立 S3 用戶端，將 AWS 登入資料物件傳遞給它，然後將 S3 用戶端傳遞給傳輸公用程式，如下所示：

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

將檔案上傳至 Amazon S3

若要將檔案上傳至 S3，請在 Transfer Utility 物件`Upload`上呼叫 ，並傳遞下列參數：

- `file` - 您要上傳之檔案的字串名稱
- `bucketName` - 儲存檔案的 S3 儲存貯體字串名稱

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

上述程式碼假設目錄 Environment. SpecialFolder.ApplicationData. 上傳會自動在大型檔案上使用 S3 的分段上傳功能，以增強輸送量。

從 Amazon S3 下載檔案

若要從 S3 下載檔案，請在 Transfer Utility 物件`Download`上呼叫 ，並傳遞下列參數：

- `file` - 您要下載之檔案的字串名稱
- `bucketName` - 您要從中下載檔案的 S3 儲存貯體字串名稱
- `key` - 代表要下載之 S3 物件 （在此情況下為 檔案） 名稱的字串

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

如需從 Xamarin 應用程式存取 Amazon S3 的詳細資訊，請參閱 Amazon Simple Storage Service (S3)。

使用 Cognito Sync 同步使用者資料

Amazon Cognito Sync 可讓您輕鬆地在 AWS 雲端中儲存行動使用者資料，例如應用程式偏好設定或遊戲狀態，而無需撰寫任何後端程式碼或管理任何基礎設施。您可以在使用者的裝置上本機儲存資料，讓您的應用程式即使在裝置離線時也能運作。您也可以跨使用者的裝置同步資料，使其應用程式體驗保持一致，無論其使用的裝置為何。

以下教學課程說明如何將 Sync 與您的應用程式整合。

專案設定

先決條件

開始本教學課程之前，您必須完成設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 上的所有說明。

授予對 Cognito 同步資源的存取權

與您在設定期間建立的未驗證和驗證角色相關聯的預設政策，會授予應用程式對 Cognito Sync 的存取權。無需進一步設定。

將用於 Cognito 同步的 NuGet 套件新增至您的專案

請遵循設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的步驟 4，將 Cognito SyncManager NuGet 套件新增至您的專案。

初始化 CognitoSyncManager

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

若要同步未驗證的使用者資料：

1. 建立資料集。
2. 將使用者資料新增至資料集。
3. 將資料集與雲端同步。

建立資料集

建立 `Dataset` 的執行個體。openOrCreateDataset 方法用於建立新的資料集，或開啟本機存放在裝置上的資料集的現有執行個體：

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

將使用者資料新增至資料集

使用者資料會以金鑰/值對的形式新增：

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

Cognito 資料集可做為字典運作，其值可透過索引鍵存取：

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

同步資料集

若要同步資料集，請呼叫其同步方法：

```
dataset.SynchronizeAsync();

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

寫入資料集的所有資料都會儲存在本機，直到資料集同步為止。本節中的程式碼假設您使用未經驗證的 Cognito 身分，因此當使用者資料與雲端同步時，每個裝置都會儲存該資料。裝置具有與其相關聯的裝置 ID。當使用者資料同步至雲端時，將會與該裝置 ID 建立關聯。

如需 Cognito Sync 的詳細資訊，請參閱 Amazon Cognito Sync。

使用 DynamoDB 存放和擷取資料

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 會移除資料儲存體的傳統可擴展性限制，同時維持低延遲和可預測的效能。

以下教學課程說明如何將 DynamoDB 物件持久性模型與您的應用程式整合，以將物件存放在 DynamoDB 中。

專案設定

先決條件

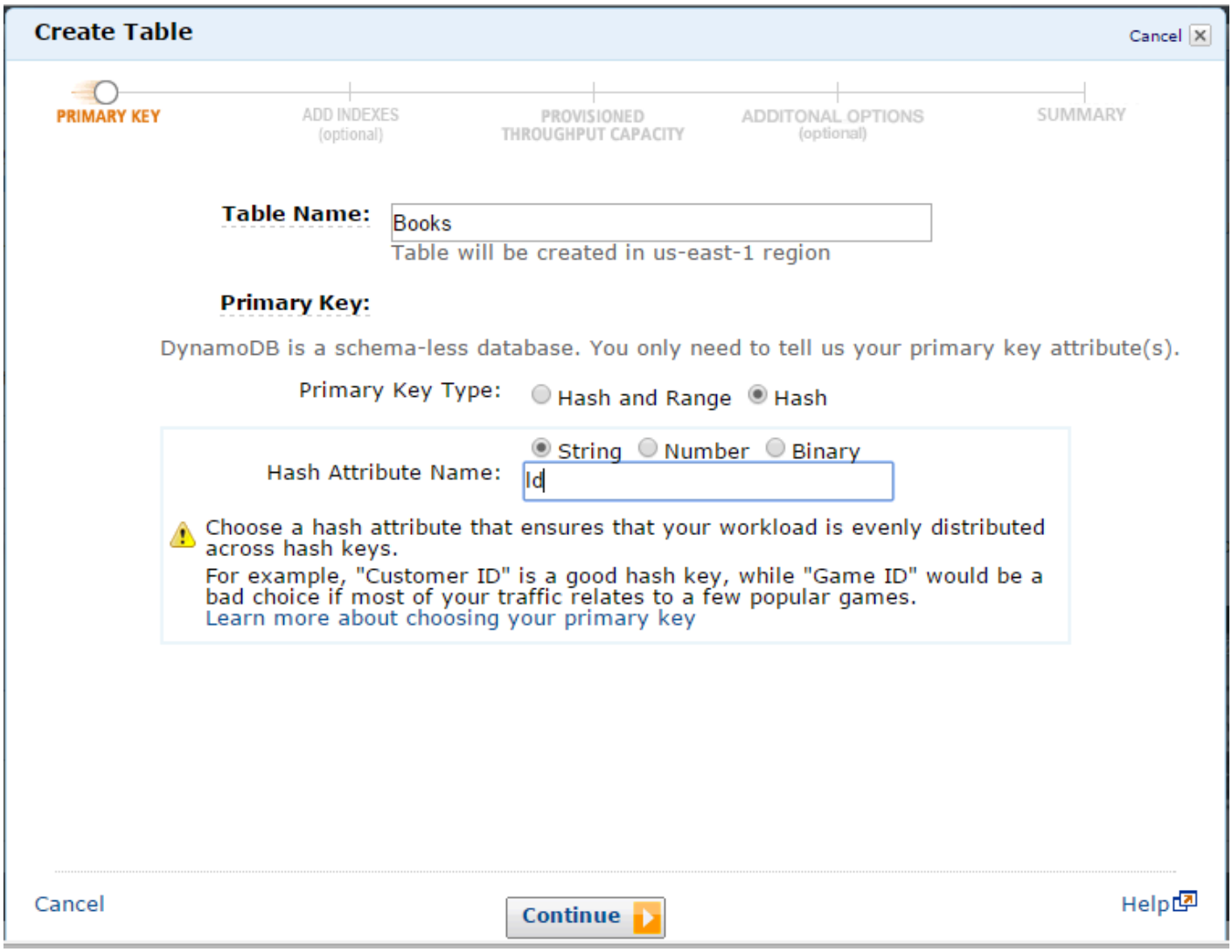
開始本教學課程之前，您必須完成設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 上的所有說明。

建立 DynamoDB 資料表

您必須先建立資料表，才能讀取和寫入資料至 DynamoDB 資料庫。建立資料表時，您必須指定主索引鍵。主索引鍵由雜湊屬性和選用的範圍屬性組成。如需如何使用主要屬性和範圍屬性的詳細資訊，請參閱使用資料表。

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. 指定您的資料表名稱、主索引鍵類型 (Hash) 和雜湊屬性名稱 (「Id」)，如下所示，然後按一下繼續：

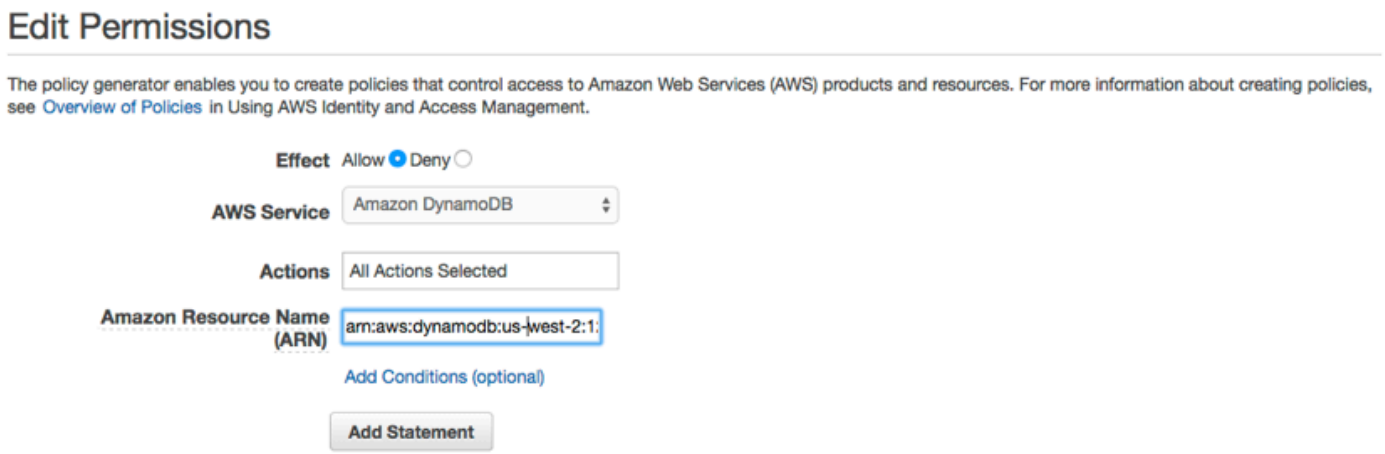


3. 將下一個畫面中的編輯欄位保留空白，然後按一下繼續。
4. 接受讀取容量單位和寫入容量單位的預設值，然後按一下繼續。
5. 在下一個畫面中，在傳送通知至：文字方塊中輸入您的電子郵件地址，然後按一下繼續。檢閱畫面隨即出現。
6. 按一下 Create (建立)。建立資料表可能需要幾分鐘的時間。

設定 DynamoDB 的許可

若要讓身分集區存取 Amazon DynamoDB，您必須修改身分集區的角色。

1. 導覽至 Identity and Access Management Console，然後按一下左側窗格中的角色。搜尋您的身分集區名稱 - 將列出兩個角色，一個用於未驗證的使用者，另一個用於已驗證的使用者。
2. 按一下未驗證使用者的角色（身分集區名稱將附加「不驗證」），然後按一下建立角色政策。
3. 選取政策產生器，然後按一下選取。
4. 在編輯許可頁面上，輸入下圖中顯示的設定。DynamoDB 資料表的 Amazon Resource Name (ARN) 看起來類似`arn:aws:dynamodb:us-west-2:123456789012:table/Books`，由資料表所在的區域、擁有者的 AWS 帳戶號碼，以及格式為 的資料表名稱組成`table/Books`。如需指定 ARNs 的詳細資訊，請參閱 DynamoDB 的 Amazon Resource Names。



5. 按一下新增陳述式，然後按一下下一步。精靈會顯示產生的組態。
6. 按一下套用政策。

將 DynamoDB 的 NuGet 套件新增至您的專案

請遵循設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的步驟 4，將 DynamoDB NuGet 套件新增至您的專案。

初始化 AmazonDynamoDBClient

將初始化的 Amazon Cognito AmazonDynamoDB 登入資料提供者和您的區域傳遞至建構函式，然後將用戶端傳遞至 DynamoDBContext：

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

建立類別

若要將資料列寫入資料表，請定義類別以保留資料列資料。類別也應該包含屬性，這些屬性會保留資料列的屬性資料，並將對應至在主控台中建立的 DynamoDB 資料表。下列類別宣告會說明這類類別：

```
[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

若要儲存項目，請先建立 物件：

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

然後儲存：

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

若要更新資料列，請修改 DDTableRow類別的執行個體並呼叫 `AWSDynamoObjectMapper.save()`，如上所示。

擷取項目

使用主索引鍵擷取項目：

```
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

若要刪除項目：

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

如需從 Xamarin 應用程式存取 DynamoDB 的詳細資訊，請參閱 Amazon DynamoDB。

使用 Amazon Mobile Analytics 追蹤應用程式用量資料

Amazon Mobile Analytics 可讓您測量應用程式用量和應用程式收入。透過追蹤關鍵趨勢，例如新使用者與傳回使用者、應用程式收入、使用者保留和自訂應用程式內行為事件，您可以做出資料驅動型決策，以提高應用程式的參與度和獲利。

以下教學課程說明如何將 Mobile Analytics 與您的應用程式整合。

專案設定

先決條件

開始本教學課程之前，您必須完成設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 上的所有說明。

在 Mobile Analytics 主控台中建立應用程式

前往 Amazon Mobile Analytics 主控台並建立應用程式。請記下 appId值，因為稍後會需要它。當您在 Mobile Analytics 主控台中建立應用程式時，您需要指定身分集區 ID。如需建立身分集區的指示，請參閱設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK。

若要進一步了解如何在 主控台中運作，請參閱 Amazon Mobile Analytics 使用者指南。

設定 Mobile Analytics 的許可

與您在設定期間建立的角色相關聯的預設政策，會授予應用程式對 Mobile Analytics 的存取權。無需進一步設定。

將適用於 Mobile Analytics 的 NuGet 套件新增至您的專案

請遵循設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的步驟 4，將 Mobile Analytics NuGet 套件新增至您的專案。

設定 Mobile Analytics 設定

Mobile Analytics 定義可在 awsconfig.xml 檔案中設定的一些設定：

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

- AllowUseDataNetwork - 指定工作階段事件是否在資料網路上傳送的布林值。
- DBWarningThreshold - 這是資料庫大小的限制，一旦達到此限制，就會產生警告日誌。
- MaxDBSize - 這是 SQLIte 資料庫的大小。當資料庫達到大小上限時，任何其他事件都會遭到捨棄。
- MaxRequestSize - 這是應該在 HTTP 請求中傳輸至行動分析服務的位元組請求大小上限。
- SessionTimeout - 這是應用程式進入背景以及工作階段可以終止的時間間隔。

上面顯示的設定是每個組態項目的預設值。

初始化 MobileAnalyticsManager

若要初始化 MobileAnalyticsManager，請在 上呼叫 GetOrCreateInstance`MobileAnalyticsManager`，並傳入您的 AWS 登入資料、您的區域、您的 Mobile Analytics 應用程式 ID 和您的選用組態物件：

```
var manager = MobileAnalyticsManager.GetOrCreateInstance(
```

```
  "APP_ID",
  "Credentials",
  "RegionEndPoint",
  config
);
```

追蹤工作階段事件

Xamarin Android

覆寫活動的 OnPause()和 OnResume()方法來記錄工作階段事件。

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

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

這需要針對應用程式中的每個活動實作。

Xamarin iOS

在您的 AppDelegate.cs:

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

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

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

使用 SNS (Xamarin iOS) 接收推播通知

本文件說明如何使用 Amazon Simple Notification Service (SNS) 和適用於 .NET 和 Xamarin 的 AWS Mobile SDK，將推送通知傳送至 Xamarin iOS 應用程式。

專案設定

先決條件

開始本教學課程之前，您必須完成設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 上的所有說明。

設定 SNS 的許可

請遵循設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的步驟 2，將下列政策連接至應用程式的角色。這將給予您的應用程式存取 SNS 的適當許可：

1. 前往 IAM 主控台，然後選取您要設定的 IAM 角色。
2. 按一下連接政策，選取 AmazonSNSFullAccess 政策，然後按一下連接政策。

> **Warning**
>
> 在生產環境中不建議使用 AmazonSNSFullAccess。我們在這裡使用它，可讓您快速啟動和執行。如需指定 IAM 角色許可的詳細資訊，請參閱 IAM 角色許可概觀。

在 Apple iOS 開發人員計劃中取得成員資格

您需要在實體裝置上執行應用程式，才能接收推播通知。若要在裝置上執行應用程式，您必須擁有 Apple iOS 開發人員計劃成員資格。擁有成員資格後，您可以使用 Xcode 產生簽署身分。如需詳細資訊，請參閱 Apple 的應用程式分佈 Quick Start 文件。

建立 iOS 憑證

首先，您需要建立 iOS 憑證。然後，您需要建立為推送通知設定的佈建設定檔。若要這麼做：

1. 前往 Apple 開發人員成員中心，按一下憑證、識別符和設定檔。
2. 按一下 iOS 應用程式下的識別符，按一下網頁右上角的加號按鈕以新增 iOS 應用程式 ID，然後輸入應用程式 ID 描述。
3. 向下捲動至新增 ID 尾碼區段，然後選取明確應用程式 ID，然後輸入您的套件識別碼。

4. 向下捲動至應用程式服務區段，然後選取推送通知。
5. 按一下 Continue (繼續)。
6. 請按 Submit (提交)。
7. 按一下完成。
8. 選取您剛建立的應用程式 ID，然後按一下編輯。
9. 向下捲動至推送通知區段。按一下開發 SSL 憑證下的建立憑證。
10. 依照指示建立憑證簽署請求 (CSR)、上傳請求，以及下載將用於與 Apple Notification Service (APNS) 通訊的 SSL 憑證。
11. 返回憑證、識別符和設定檔頁面。按一下佈建設定檔下的全部。
12. 按一下右上角的加號按鈕，新增佈建設定檔。
13. 選取 iOS 應用程式開發，然後按一下繼續。
14. 選取您的應用程式 ID，然後按一下繼續。
15. 選取您的開發人員憑證，然後按一下繼續。
16. 選取您的裝置，然後按一下繼續。
17. 輸入設定檔名稱，然後按一下產生。
18. 下載並按兩下佈建檔案以安裝佈建設定檔。

如需佈建針對推送通知設定之設定檔的詳細資訊，請參閱 Apple 的設定推送通知文件。

使用憑證在 SNS 主控台中建立平台 ARN

1. 執行 KeyChain 存取應用程式，選取畫面左下角的我的憑證，然後在您產生連線至 APNS 的 SSL 憑證上按一下滑鼠右鍵，然後選取匯出。系統會提示您指定檔案的名稱和密碼來保護憑證。憑證將儲存在 P12 檔案中。
2. 前往 SNS 主控台，然後按一下畫面左側的應用程式。
3. 按一下建立平台應用程式以建立新的 SNS 平台應用程式。
4. 輸入應用程式名稱。
5. 選取 Apple Development for Push 通知平台。
6. 按一下選擇檔案，然後選取您在匯出 SSL 憑證時建立的 P12 檔案。
7. 輸入匯出 SSL 憑證時指定的密碼，然後按一下從檔案載入憑證。
8. 按一下建立平台應用程式。
9. 選取您剛建立的平台應用程式，並複製應用程式 ARN。在接下來的步驟中，您將需要此項目。

將 SNS 的 NuGet 套件新增至您的專案

請遵循設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的步驟 4，將 Amazon Simple Notification Service NuGet 套件新增至您的專案。

建立 SNS 用戶端

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

註冊您的遠端通知應用程式

若要註冊應用程式，請在 UIApplication 物件上呼叫 RegisterForRemoteNotifications，如下所示。在 AppDelegate.cs,在出現以下提示時插入您的平台應用程式 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 主控台傳送訊息到您的端點

1. 前往 SNS 主控台 > 應用程式。
2. 選取您的平台應用程式，選取端點，然後按一下發佈至端點。
3. 在文字方塊中輸入文字訊息，然後按一下發佈訊息以發佈訊息。

使用 SNS (Xamarin Android) 接收推播通知

本教學課程說明如何使用 Amazon Simple Notification Service (SNS) 和適用於 .NET 和 Xamarin 的 AWS Mobile SDK，將推送通知傳送至 Xamarin Android 應用程式。

專案設定

先決條件

開始本教學課程之前，您必須完成設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 上的所有說明。

設定 SNS 的許可

請遵循設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的步驟 2，將下列政策連接至應用程式的角色。這將給予您的應用程式存取 SNS 的適當許可：

1. 前往 IAM 主控台，然後選取您要設定的 IAM 角色。
2. 按一下連接政策，選取 AmazonSNSFullAccess 政策，然後按一下連接政策。

> **Warning**
>
> 在生產環境中不建議使用 AmazonSNSFullAccess。我們在這裡使用它，可讓您快速啟動和執行。如需指定 IAM 角色許可的詳細資訊，請參閱 IAM 角色許可概觀。

在 Google Cloud 上啟用推播通知

首先，新增新的 Google API 專案：

1. 前往 Google 開發人員主控台。
2. 按一下建立專案。

3. 在新增專案方塊中，輸入專案名稱，記下專案 ID （稍後需要），然後按一下建立。

接著，為您的專案啟用 Google Cloud Messaging (GCM) 服務：

1. 在 Google 開發人員主控台中，應該已選取您的新專案。如果沒有，請在頁面頂端的下拉式清單中選取它。
2. 從頁面左側的側邊列中選取 APIs & 驗證。
3. 在搜尋方塊中，輸入「Google Cloud Messaging for Android」，然後按一下 Google Cloud Messaging for Android 連結。
4. 按一下啟用 API。

最後，取得 API 金鑰：

1. 在 Google 開發人員主控台中，選取 APIs & 驗證 > 登入資料。
2. 在公有 API 存取下，按一下建立新金鑰。
3. 在建立新的金鑰對話方塊中，按一下伺服器金鑰。
4. 在產生的對話方塊中，按一下建立並複製顯示的 API 金鑰。您將使用此 API 金鑰稍後執行身分驗證。

使用專案 ID 在 SNS 主控台中建立平台 ARN

1. 前往 SNS 主控台。
2. 按一下畫面左側的應用程式。
3. 按一下建立平台應用程式以建立新的 SNS 平台應用程式。
4. 輸入應用程式名稱。
5. 針對推播通知平台選取 Google Cloud Messaging (GCM)。
6. 將 API 金鑰貼到標記為 API 金鑰的文字方塊中。
7. 按一下建立平台應用程式。
8. 選取您剛建立的平台應用程式，並複製應用程式 ARN。

將 SNS 的 NuGet 套件新增至您的專案

請遵循設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的步驟 4，將 Amazon Simple Notification Service NuGet 套件新增至您的專案。

建立 SNS 用戶端

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

註冊您的遠端通知應用程式

若要在 Android 上註冊遠端通知，您需要建立可接收 Google Cloud 訊息的 BroadcastReceiver。在出現提示的情況下變更以下套件名稱：

```
[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);
      }
```

```
}
```

以下是從 BroadcastReceiver 接收推播通知並在裝置的通知列上顯示通知的服務：

```
[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 主控台傳送訊息到您的端點

1. 前往 SNS 主控台 > 應用程式。
2. 選取您的平台應用程式，選取端點，然後按一下發佈至端點。
3. 在文字方塊中輸入文字訊息，然後按一下發佈訊息以發佈訊息。

Amazon Cognito 身分

什麼是 Amazon Cognito Identity？

Amazon Cognito Identity 可讓您為使用者建立唯一身分，並使用身分提供者對其進行驗證。使用 身分，您可以取得暫時、有限權限的 AWS 登入資料，以與 Amazon Cognito Sync 同步資料，或直接存取其他 AWS 服務。Amazon Cognito Identity 支援公有身分提供者—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 Identity 的詳細資訊，請參閱 Amazon Cognito 開發人員指南。

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

使用公有供應商來驗證使用者

使用 Amazon Cognito Identity，您可以為使用者建立唯一身分，並對其進行驗證，以安全地存取您的 AWS 資源，例如 Amazon S3 或 Amazon DynamoDB。Amazon Cognito Identity 支援公有身分提供者—Amazon、Facebook、Twitter/Digits、Google 或任何 OpenID Connect 相容提供者—以及未驗證的身分。

如需有關使用 Amazon、Facebook、Twitter/Digits 或 Google 等公有身分提供者來驗證使用者的資訊，請參閱《Amazon Cognito 開發人員指南》中的外部提供者。

使用開發人員驗證的身分

除了透過 Facebook、Google 和 Amazon 的 Web 聯合身分之外，Amazon Cognito 還支援開發人員驗證的身分。透過開發人員驗證的身分，您可以透過自己的現有身分驗證程序註冊和驗證使用者，同時仍使用 Amazon Cognito Sync 同步使用者資料並存取 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。

如需開發人員驗證身分的相關資訊，請參閱《Amazon Cognito 開發人員指南》中的開發人員驗證身分。

Amazon Cognito 同步

什麼是 Amazon Cognito Sync？

Cognito Sync 是一種 AWS 服務和用戶端程式庫，可跨裝置同步使用者資料（例如遊戲分數、使用者偏好設定、遊戲狀態）。您可以使用 Cognito Sync API 跨裝置同步使用者資料。若要在應用程式中使用 Cognito Sync，您必須在專案中包含 |。

如需如何在應用程式中整合 Amazon Cognito Sync 的說明，請參閱 Amazon Cognito Sync 開發人員指南。

Amazon Mobile Analytics

Amazon Mobile Analytics 是一項服務，用於大規模收集、視覺化、了解和擷取應用程式用量資料。Mobile Analytics 可以輕鬆擷取標準裝置資料和自訂事件，並代表您自動計算報告。除了下列彙總報告之外，您也可以設定將資料自動匯出至 Redshift 和 S3 以進行進一步分析。

使用 Amazon Mobile Analytics，您可以追蹤客戶行為、彙總指標、產生資料視覺化，以及識別有意義的模式。

重要概念

報告類型

Mobile Analytics 現成可在 Mobile Analytics 主控台中提供下列報告：

- 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
- 每日作用中使用者的平均收入 (ARPDAU) 和每日付款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

這些報告是透過主控台中的六個報告索引標籤提供：

- 概觀 – 在simple-to-review的儀表板中追蹤九個預先選取的報告，以快速了解參與度：MAA、DAU、新使用者、每日工作階段、黏性因素、1-Day保留期、ARPDAU、每日付款使用者、ARPPDAU。
- 作用中使用者 – 追蹤每天和每月有多少使用者與您的應用程式互動，並監控黏性因素來衡量參與度、吸引力和獲利。
- 工作階段 – 追蹤應用程式在特定日期的使用頻率，以及每位使用者在一天內開啟應用程式的頻率。
- 保留 – 追蹤客戶每天和每週回到應用程式的速率。
- 收入 – 追蹤應用程式內收入趨勢，以識別獲利改善的領域。
- 自訂事件 – 追蹤您應用程式特有的自訂定義使用者動作。

若要進一步了解 Mobile Analytics 報告並在 Mobile Analytics 主控台中運作，請參閱 Mobile Analytics 開發人員指南中的 Mobile Analytics 主控台報告概觀。 Mobile Analytics

專案設定

先決條件

若要在應用程式中使用 Mobile Analytics，您需要將 SDK 新增至專案。若要這樣做，請遵循設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的指示。

設定 Mobile Analytics 設定

Mobile Analytics 定義可在 awsconfig.xml 檔案中設定的一些設定：

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

- SessionTimeout - 如果應用程式停留在背景的時間大於 SessionTimeout，則 Mobile Analytics 用戶端會終止目前的工作階段，並在應用程式回到前景時建立新的工作階段。建議使用介於 5 到 10 之間的值。預設值為 5。
- MaxDBSize - 用於事件本機儲存的資料庫大小上限（以位元組為單位）。如果資料庫大小超過此值，則會忽略其他事件。建議使用介於 1MB 到 10MB 之間的值。預設值為 5242880 (5MB)。
- DBWarningThreashold - 警告閾值。有效值的範圍介於 0 到 1 之間。如果值超過閾值，將產生警告日誌。預設值為 0.9。
- MaxRequestSize - 對 Mobile Analytics 服務提出的 HTTP 請求大小上限。此值以位元組為單位指定，範圍介於 1-512KB 之間。預設值 os 102400 (100KB)。請勿使用大於 512KB 的值，這可能會導致服務拒絕 HTTP 請求。
- AllowUseDataNetwork - 指出是否允許透過行動數據網路進行服務呼叫的值。請謹慎使用此選項，因為這可能會增加客戶的資料用量。

上面顯示的設定是每個組態項目的預設值。

將 Mobile Analytics 與您的應用程式整合

以下各節說明如何將 Mobile Analytics 與您的應用程式整合。

在 Mobile Analytics 主控台中建立應用程式

前往 Amazon Mobile Analytics 主控台並建立應用程式。請記下 appId值，因為稍後會需要它。當您在 Mobile Analytics 主控台中建立應用程式時，您需要指定身分集區 ID。如需建立身分集區的指示，請參閱設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK。

若要進一步了解如何使用 Mobile Analytics 主控台，請參閱 Mobile Analytics 開發人員指南中的 Mobile Analytics 主控台報告概觀。 Mobile Analytics

建立 MobileAnalyticsManager 用戶端

若要初始化 MobileAnalyticsManager，請在 上呼叫 GetOrCreateInstanceMobileAnalyticsManager，並傳入您的 AWS 登入資料、您的區域、您的 Mobile Analytics 應用程式 ID 和您的選用組態物件：

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

APP_ID 會在應用程式建立精靈期間為您產生 。這兩個值都必須符合 Mobile Analytics 主控台中的值。APP_ID 用於在 Mobile Analytics 主控台中將您的資料分組。若要在 Mobile Analytics 主控台中建立應用程式後尋找您的應用程式 ID，請瀏覽至 Mobile Analytics 主控台，按一下畫面右上角的齒輪圖示。這會顯示應用程式管理頁面，其中列出所有已註冊的應用程式及其應用程式 IDs。

記錄獲利事件

適用於 .NET 和 Xamarin 的 AWS Mobile SDK 提供 MonetizationEvent類別，可讓您產生獲利事件來追蹤行動應用程式中的購買。下列程式碼片段示範如何建立獲利事件：

```
// 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 可讓您定義自訂事件。自訂事件完全由您定義；它們可協助您追蹤應用程式或遊戲特定的使用者動作。如需自訂事件的詳細資訊，請參閱自訂事件。

在此範例中，我們會說我們的應用程式是遊戲，而且我們希望在使用者完成關卡時記錄事件。透過建立新的`AmazonMobileAnalyticsEvent`執行個體來建立「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);
```

錄製工作階段

Xamarin iOS

當應用程式失去焦點時，您可以暫停工作階段。對於 iOS 應用程式，請在 AppDelegate.cs 檔案中覆寫 `DidEnterBackground` 和 `WillEnterForeground`以呼叫 `MobileAnalyticsManager.PauseSesion`和 `MobileAnalyticsManager.ResumeSession`，如下列程式碼片段所示：

```
public override void DidEnterBackground(UIApplication application)
{
```

```
  // ...
  _manager.PauseSesson();
  // ...
}

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

Xamarin Android

對於 Android 應用程式，`MobileAnalyticsManager.PauseSesion`請在 OnPause() 方法和 OnResume() 方法`MobileAnalyticsManager.ResumeSession`中呼叫 ，如下列程式碼片段所示：

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

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

根據預設，如果使用者從應用程式切換焦點不到 5 秒，並且切換回應用程式，則工作階段將會繼續。如果使用者將焦點從應用程式切換 5 秒或更久，則會建立新的工作階段。此設定可在 aws_mobile_analytics.json 組態檔案中設定，方法是將 "SESSION_DELTA" 屬性設定為建立新工作階段之前等待的秒數。

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、高度可擴展的物件儲存。Amazon S3 易於使用，具有簡單的 Web 服務界面，可從 Web 上的任何位置存放和擷取任意數量的資料。使用 Amazon S3，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實惠的物件儲存，包括雲端應用程式、內容分發、備份和封存、災難復原和大數據分析。

如需 AWS S3 區域可用性的相關資訊，請參閱 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。儲存貯體具有屬性，例如存取許可和版本控制狀態，您可以指定要儲存貯體所在的區域。

若要進一步了解 S3 儲存貯體，請參閱S3 開發人員指南》中的使用儲存貯體。

物件

物件是您存放在 Amazon S3 中的資料。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，存放在歐洲（愛爾蘭）區域的物件絕不會離開它。存放在 Amazon S3 區域中的物件實際保留在該區域中。Amazon S3 不會保留副本或將其移至任何其他區域。不過，只要您有必要許可，就可以從任何位置存取物件。

物件可以是任何檔案類型：映像、備份資料、電影等。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，請參閱 S3 開發人員指南中的編輯儲存貯體許可。

若要進一步了解 S3 物件，請參閱S3 開發人員指南》中的使用物件。

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。有兩種類型的中繼資料：

- 系統中繼資料 – 有時由 Amazon S3 處理，例如 Content-Type 和 Content-Length。
- 使用者中繼資料 – 從不由 Amazon S3 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，請參閱編輯物件中繼資料。

專案設定

先決條件

若要在應用程式中使用 Amazon S3，您需要將 SDK 新增至專案。若要這樣做，請遵循設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的指示。

建立 S3 儲存貯體章節

Amazon S3 會將您應用程式的資源存放在 Amazon S3 儲存貯體 - 位於特定區域的雲端儲存容器。每個 Amazon S3 儲存貯體都必須具有全域唯一名稱。您可以使用 Amazon S3 主控台來建立儲存貯體。

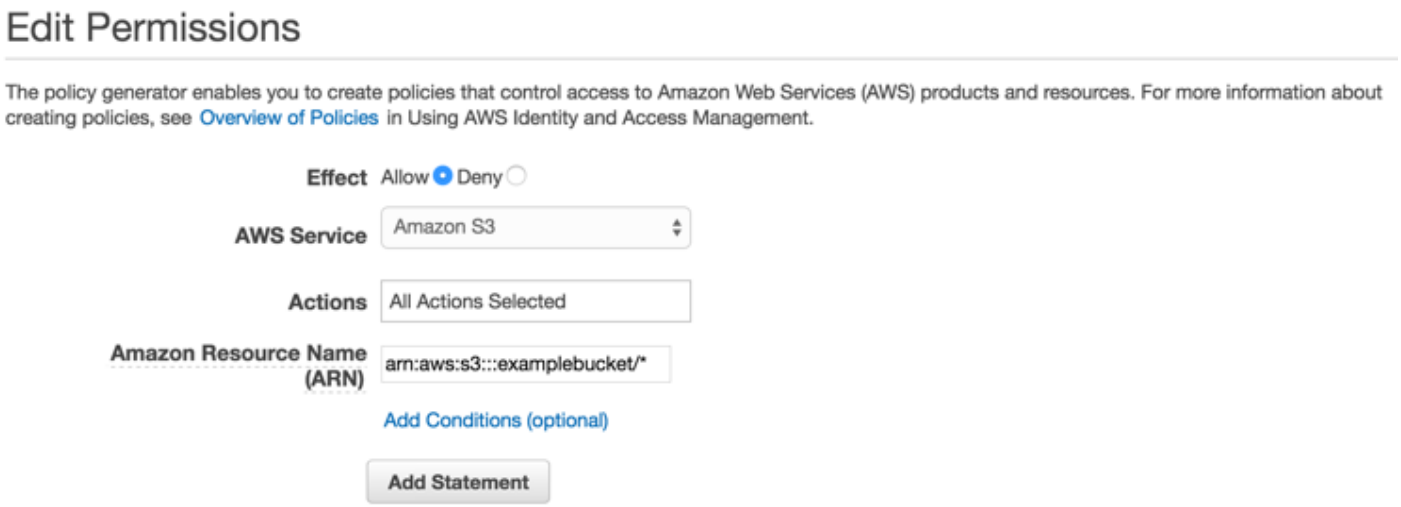
1. 登入 Amazon S3 主控台，然後按一下建立儲存貯體。
2. 輸入儲存貯體名稱，選取區域，然後按一下建立。

設定 S3 的許可

預設 IAM 角色政策會授予應用程式對 Amazon Mobile Analytics 和 Amazon Cognito Sync 的存取權。若要讓您的 Cognito 身分集區存取 Amazon S3，您必須修改身分集區的角色。

1. 前往 Identity and Access Management Console，然後按一下左側窗格中的角色。
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. 按一下未驗證使用者的角色（身分集區名稱將附加不驗證）。

4. 按一下建立角色政策，選取政策產生器，然後按一下選取。
5. 在編輯許可頁面上，輸入下圖中顯示的設定，以您自己的名稱取代 Amazon Resource Name (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的名稱組成。以下顯示的設定會讓您的身分集區完整存取指定儲存貯體的所有動作。



1. 按一下新增陳述式按鈕，然後按一下下一步。
2. 精靈會顯示您產生的組態。按一下套用政策。

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

（選用）設定 S3 請求的簽章版本

與 Amazon S3 的每次互動，可以經過驗證身分或是匿名進行。AWS 使用 Signature 第 4 版或 Signature 第 2 版演算法來驗證對服務的呼叫。

2014 年 1 月之後建立的所有新 AWS 區域僅支援 Signature 第 4 版。不過，許多較舊的區域仍然支援 Signature 第 4 版和 Signature 第 2 版請求。

如果您的儲存貯體位於此頁面列出的其中一個不支援 Signature 第 2 版請求的區域，您必須將 AWSConfigsS3.UseSignatureVersion4 屬性設定為「true」，如下所示：

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

如需 AWS Signature 版本的詳細資訊，請參閱驗證請求 (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 應用程式中與 S3 互動。在下列主題中深入探索這兩種方法：

使用 S3 Transfer 公用程式

S3 Transfer Utility 可讓您更輕鬆地從 Xamarin 應用程式上傳和下載檔案至 S3。

初始化 TransferUtility

建立 S3 用戶端，將 AWS 登入資料物件傳遞給它，然後將 S3 用戶端傳遞給傳輸公用程式，如下所示：

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

（選用）設定 TransferUtility

您可以設定三個選用屬性：

- ConcurrentServiceRequests - 決定將使用多少個作用中執行緒或非同步 Web 請求數目來上傳/下載檔案。預設值為 10。
- MinSizeBeforePartUpload - 取得或以位元組為單位設定上傳組件的最小組件大小。預設為 16 MB。減少最小部分大小會導致分段上傳分割成更多較小的部分。將此值設為太低會對傳輸速度產生負面影響，導致每個部分的額外延遲和網路通訊。
- NumberOfUploadThreads - 取得或設定執行執行緒的數量。此屬性會決定將用於上傳檔案的作用中執行緒數量。預設值為 10 個執行緒。

若要設定 S3 TransferUtility 用戶端，請建立組態物件、設定屬性，並將物件傳遞至 TransferUtility 建構函數，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，請在 Transfer Utility 物件Download上呼叫 ，並傳遞下列參數：

- file - 您要下載之檔案的字串名稱
- bucketName - 您要從中下載檔案的 S3 儲存貯體字串名稱
- key - 代表要下載之 S3 物件 （在此情況下為 檔案） 名稱的字串

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

上傳檔案

若要將檔案上傳至 S3，請在 Transfer Utility 物件Upload上呼叫 ，並傳遞下列參數：

- file - 您要上傳之檔案的字串名稱
- bucketName - 儲存檔案的 S3 儲存貯體字串名稱

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

上述程式碼假設目錄 Environment. SpecialFolder.ApplicationData. 上傳會自動在大型檔案上使用 S3 的分段上傳功能，以增強輸送量。

使用服務層級 S3 APIs

除了使用 S3 TransferUtility 之外，您也可以使用低階 S3 APIs 與 S3 互動。

初始化 Amazon S3 用戶端

若要使用 Amazon S3，我們首先需要建立 AmazonS3Client 執行個體，該執行個體參考您先前建立的 CognitoAWSCredentials 執行個體和您的區域：

```
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

若要將檔案上傳至 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

若要刪除 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);
```

刪除多個項目

若要使用單一 HTTP 請求從儲存貯體刪除多個物件：

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

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

```
    }
};

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

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

您最多可以指定 1000 個金鑰。

列出儲存貯體

若要傳回請求已驗證寄件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

您可以傳回部分或全部（最多 1000 個）存放在 S3 儲存貯體中的物件。若要這樣做，您必須具有儲存貯體的讀取存取權。

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

取得儲存貯體的區域

若要取得儲存貯體所在的區域：

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

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

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

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

取得儲存貯體的政策

若要取得儲存貯體的政策：

```
// 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、高度可擴展的非關聯式資料庫服務。DynamoDB 會移除資料儲存體的傳統可擴展性限制，同時維持低延遲和可預測的效能。

重要概念

DynamoDB 資料模型概念包括資料表、項目和屬性。

表格

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

在關聯式資料庫中，資料表具有預先定義的結構描述，例如資料表名稱、主索引鍵、資料欄名稱清單及其資料類型。存放在資料表中的所有記錄都必須有相同的資料欄集。相反地，DynamoDB 只需要資料表具有主索引鍵，但不需要您事先定義所有屬性名稱和資料類型。

若要進一步了解使用資料表，請參閱在 DynamoDB 中使用資料表。

項目和屬性

DynamoDB 資料表中的個別項目可以有任意數量的屬性，但項目大小限制為 400 KB。項目大小是其屬性名稱和值（二進位和 UTF-8 長度）的長度總和。

項目中的每個屬性都是名稱/值對。屬性可以是單一值或多值集。例如，書籍項目可以有標題和作者屬性。每本書都有一個標題，但可以有許多作者。多值屬性是集合；不允許重複值。

例如，請考慮在 DynamoDB 中存放產品目錄。您可以建立資料表 ProductCatalog，並以 ID 屬性做為其主索引鍵。主索引鍵可唯一識別每個項目，因此資料表中沒有任何兩個產品可以具有相同的 ID。

若要進一步了解使用項目，請參閱在 DynamoDB 中使用項目。

資料類型

Amazon DynamoDB 支援下列資料類型：

- 純量類型 – 數字、字串、二進位、布林值和 Null。
- 多值類型 – 字串集、數字集和二進位集。
- 文件類型 – 清單和映射。

如需純量資料類型、多值資料類型和文件資料類型的詳細資訊，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，因此兩個項目不能具有相同的索引鍵。DynamoDB 支援下列兩種主要金鑰類型：

- 雜湊索引鍵：主索引鍵是由一個屬性組成，即雜湊屬性。DynamoDB 會在此主索引鍵屬性上建置未排序的雜湊索引。資料表中的每個項目都由其雜湊索引鍵值唯一識別。
- 雜湊和範圍索引鍵：主索引鍵由兩個屬性組成。第一個屬性是雜湊屬性，第二個屬性是範圍屬性。DynamoDB 在雜湊主索引鍵屬性上建置未排序的雜湊索引，並在範圍主索引鍵屬性上建置排序範圍索引。資料表中的每個項目都是由其雜湊和範圍索引鍵值組合所唯一識別。兩個項目可能具有相同的雜湊索引鍵值，但這兩個項目必須具有不同的範圍索引鍵值。

次要索引

當您使用雜湊和範圍索引鍵建立資料表時，您可以選擇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 支援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：與資料表具有相同雜湊索引鍵，但範圍索引鍵不同的索引。
- 全域次要索引：具有雜湊和範圍索引鍵的索引，可能與資料表上的索引鍵不同。

每個資料表最多可定義 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DynamoDB 開發人員指南》中的改善 DynamoDB 中次要索引的資料存取。 DynamoDB

查詢和掃描

除了使用主索引鍵存取項目之外，Amazon DynamoDB 還提供兩個 APIs來搜尋資料：查詢和掃描。我們建議您閱讀 DynamoDB 開發人員指南中的查詢和掃描準則，以熟悉一些最佳實務。

Query

查詢操作只會使用主索引鍵屬性值，在資料表或次要索引中尋找項目。您必須提供雜湊索引鍵屬性名稱和要搜尋的不同值。您可以選擇性地提供範圍索引鍵屬性名稱和值，並使用比較運算子來精簡搜尋結果。

如需範例查詢，請參閱：

- 使用文件模型
- 使用物件持久性模型
- 使用 DynamoDB 服務層級 APIs

如需查詢的詳細資訊，請參閱《DynamoDB 開發人員指南》中的查詢。

Scan

掃描操作會讀取資料表或次要索引中的每個項目。根據預設，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，讓掃描只傳回一些屬性，而不是所有屬性。

如需範例掃描，請參閱：

- 使用文件模型
- 使用物件持久性模型
- 使用 DynamoDB 服務層級 APIs

如需掃描的詳細資訊，請參閱《DynamoDB 開發人員指南》中的掃描。

專案設定

先決條件

若要在應用程式中使用 DynamoDB，您需要將 SDK 新增至專案。若要這樣做，請遵循設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的指示。

建立 DynamoDB 資料表

若要建立資料表，請前往 DynamoDB 主控台並遵循下列步驟：

1. 按一下 Create Table (建立資料表)。
2. 輸入資料表的名稱。
3. 選取雜湊做為主索引鍵類型。
4. 選取類型，然後輸入雜湊屬性名稱的值。按一下 Continue (繼續)。
5. 在新增索引頁面上，如果您計劃使用全域次要索引，請將索引類型設定為「全域次要索引」，然後在索引雜湊索引鍵下，輸入次要索引的值。這可讓您同時使用主要索引和次要索引來查詢和掃描。按一下將索引新增至資料表，然後按一下繼續。若要略過使用全域次要索引，請按一下繼續。
6. 將讀取和寫入容量設定為所需的層級。如需設定容量的詳細資訊，請參閱 Amazon DynamoDB 中的佈建輸送量。按一下 Continue (繼續)。
7. 在下一個畫面上，視需要輸入通知電子郵件以建立輸送量警示。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，您必須設定正確的許可。下列 IAM 政策允許使用者刪除、取得、放置、查詢、掃描和更新特定 DynamoDB 資料表中的項目，該資料表由 ARN 識別：

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

您可以在 IAM 主控台中修改政策。您應該根據應用程式的需求新增或移除允許的動作。

若要進一步了解 IAM 政策，請參閱使用 IAM。

若要進一步了解 DynamoDB 特定政策，請參閱DynamoDB 開發人員指南》中的使用 IAM 控制對 DynamoDB 資源的存取。 DynamoDB

將 DynamoDB 與您的應用程式整合

適用於 .NET 和 Xamarin 的 AWS Mobile SDK 提供高階程式庫，可用於 DynamoDB。您也可以直接對低階 DynamoDB API 提出請求，但對於大多數使用案例，建議使用高階程式庫。AmazonDynamoDBClient 是高階程式庫中特別實用的部分。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 Mobile SDK 可讓您使用適用於 .NET 的 AWS 開發套件中的 APIs 進行呼叫，以使用 DynamoDB。所有 APIs都可在 AWSSDK.dll 中使用。如需下載適用於 .NET 的 AWS 開發套件的詳細資訊，請參閱適用於 .NET 的 AWS 開發套件。

您可以透過三種方式在 Xamarin 應用程式中與 DynamoDB 互動：

- 文件模型：此 API 提供低階 DyanmoDB API 的包裝函式類別，以進一步簡化您的程式設計任務。資料表和文件是金鑰包裝函式類別。您可以使用 文件模型進行資料操作，例如建立、擷取、更新和刪除項目。API 可在 Amazon.DynamoDB.DocumentModel 命名空間中使用。
- 物件持久性模型：物件持久性 API 可讓您將用戶端類別映射至 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 類別提供方法，可讓您將用戶端物件儲存到資料表、將項目擷取為物件，以及執行查詢和掃描。您可以使用物件持久性模型進行資料操作，例如建立、擷取、更新和刪除項目。您必須先使用服務用戶端 API 建立資料表，然後使用物件持久性模型將類別映射至資料表。API 可在 Amazon.DynamoDB.DataModel 命名空間中使用。
- 服務用戶端 API：這是通訊協定層級 API，密切對應至 DynamoDB API。您可以將此低階 API 用於所有資料表和項目操作，例如建立、更新、刪除資料表和項目。您也可以查詢和掃描資料表。此 API 可在 Amazon.DynamoDB 命名空間中使用。

這三個模型會在下列主題中深入探索：

使用文件模型

文件模型提供低階 .NET API 的包裝函式類別。資料表和文件是金鑰包裝函式類別。您可以使用文件模型來建立、擷取、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 的說明，請參閱使用 DynamoDB 服務層級 APIs。Amazon.DynamoDB.DocumentModel 命名空間提供低階 API。

若要進一步了解文件模型，請參閱 .NET 文件模型。

建立 DynamoDB 用戶端

若要建立 DynamoDB 用戶端：

```
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";
```

將項目儲存至 DynamoDB 資料表：

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

擷取項目

若要擷取項目：

```
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);
}
```

若要依條件更新項目：

```
public async Task UpdateItemConditionallyAsync(AWSCredentials credentials,
 RegionEndpoint region) {
      var book = new Document();
      book["Id"] = id;
      book["Price"] = "30";

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

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

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

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

若要查詢和擷取作者為「Mark Twain」的所有書籍：

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

下面的掃描範例程式碼會傳回我們資料表中的所有書籍：

```
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 Mobile SDK 提供物件持久性模型，可讓您將用戶端類別映射至 DynamoDB 資料表。然後，每個物件執行個體都會對應至對應資料表中的項目。若要將用戶端物件儲

存至資料表，物件持久性模型會提供 DynamoDBContext 類別，這是 DynamoDB 的進入點。此類別可讓您連線至 DynamoDB，並可讓您存取資料表、執行各種 CRUD 操作，以及執行查詢。

物件持久性模型不提供 API 來建立、更新或刪除資料表。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 的說明，請參閱使用 DynamoDB 服務層級 APIs。

概觀

物件持久性模型提供一組屬性，可將用戶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 物件持久性模型屬性。此外，對於非主要金鑰屬性，如果您類別中的屬性名稱和您要映射的對應資料表屬性不同，則必須明確新增 DynamoDBProperty 屬性來定義映射。
- 預設映射 - 物件持久性模型預設會將類別屬性映射至資料表中同名的屬性。

您不需要映射每個單一類別屬性。您可以新增 DynamoDBIgnore 屬性來識別這些屬性。儲存和擷取物件的執行個體會省略任何標示此屬性的屬性。

支援的資料類型

物件持久性模型支援一組基本 .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
- byte
- char
- DateTime
- 小數、雙數、浮點數
- Int16, Int32, Int64
- SByte
- string
- UInt16, UInt32, UInt64

物件持久性模型也支援具有下列限制的 .NET 集合類型：

- 集合類型必須實作 ICollection 介面。
- 集合類型必須由支援的基本類型組成。例如，ICollection<string>、ICollection<bool>。
- 集合類型必須提供無參數建構函數。

如需物件持久性模型的詳細資訊，請參閱 .NET 物件持久性模型。

建立 DynamoDB 用戶端

若要建立 DynamoDB 用戶端：

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

CRUD 操作

儲存物件

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
};
```

將物件儲存至 DynamoDB 資料表：

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

擷取物件

若要擷取物件：

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

更新物件

若要更新物件：

```
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

若要查詢和擷取作者為「Charles Dickens」的所有書籍：

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

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

  Console.WriteLine("items retrieved");

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

以下掃描範例程式碼會傳回我們資料表中的所有書籍：

```
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 服務層級 APIs

Dynamo 服務層級 APIs 可讓您建立、更新和刪除資料表。您也可以使用此 API，對資料表中的項目執行一般的建立、讀取、更新和刪除 (CRUD) 操作。

建立 DynamoDB 用戶端

若要建立 DynamoDB 用戶端：

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

CRUD 操作

儲存項目

若要將項目儲存至 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

若要擷取項目：

```
// 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

若要刪除項目：

```
// 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);
```

查詢和掃描

若要查詢和擷取作者為「Mark Twain」的所有書籍：

```
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);
    });

  }
}
```

下面的掃描範例程式碼會傳回我們資料表中的所有書籍：

```
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 和適用於 .NET 和 Xamarin 的 AWS Mobile SDK，您可以撰寫可接收行動推播通知的應用程式。如需 SNS 的相關資訊，請參閱 Amazon Simple Notification Service。

重要概念

Amazon SNS 允許不同裝置上的應用程式和最終使用者透過 Mobile Push 通知 (Apple、Google 和 Kindle Fire 裝置）、HTTP/HTTPS、電子郵件/電子郵件 JSON、SMS 或 Amazon Simple Queue Service (SQS) 佇列或 AWS Lambda 函數接收通知。SNS 可讓您傳送個別訊息或散發訊息給訂閱單一主題的大量收件人。

主題

主題是「存取點」，可讓收件人動態訂閱相同通知的相同副本。一個主題可以支援交付到多個端點類型 – 例如，您可以將 iOS、Android 和 SMS 收件人分組在一起。

Subscriptions

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、電子郵件地址或可從 Amazon SNS 接收通知訊息的 Amazon SQS 佇列。 Amazon SNS 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。

發布

當您發佈至主題時，SNS 會將訊息的適當格式副本提供給該主題的每個訂閱者。對於行動推播通知，您可以直接發佈至端點或訂閱端點至主題。

專案設定

先決條件

若要在應用程式中使用 SNS，您需要將 SDK 新增至專案。若要這樣做，請遵循設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的指示。

設定 SNS 的許可

如需設定 SNS 許可的資訊，請參閱管理對 Amazon SNS 主題的存取。

將 SNS 的 NuGet 套件新增至您的專案

請遵循設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的步驟 4，將 Amazon Simple Notification Service NuGet 套件新增至您的專案。

將 SNS 與您的應用程式整合

有許多方式可以在 Xamarin 應用程式中與 SNS 互動：

傳送推播通知 (Xamarin Android)

本文件說明如何使用 Amazon Simple Notification Service (SNS) 和適用於 .NET 和 Xamarin 的 AWS Mobile SDK，將推送通知傳送至 Xamarin Android 應用程式。

專案設定

先決條件

開始本教學課程之前，您必須完成設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 上的所有說明。

設定 SNS 的許可

請遵循設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的步驟 2，將下列政策連接至應用程式的角色。這將授予您的應用程式存取 SNS 的適當許可：

1. 前往 IAM 主控台，然後選取您要設定的 IAM 角色。
2. 按一下連接政策，選取 AmazonSNSFullAccess 政策，然後按一下連接政策。

> **Warning**
>
> 不建議在生產環境中使用 AmazonSNSFullAccess。我們在這裡使用它，可讓您快速啟動和執行。如需指定 IAM 角色許可的詳細資訊，請參閱 IAM 角色許可概觀。

在 Google Cloud 上啟用推播通知

首先，新增 Google API 專案：

1. 前往 Google 開發人員主控台。
2. 按一下建立專案。
3. 在新增專案方塊中，輸入專案名稱，記下專案 ID （稍後需要），然後按一下建立。

接著，為您的專案啟用 Google Cloud Messaging (GCM) 服務：

1. 在 Google 開發人員主控台中，應該已選取您的新專案。如果沒有，請在頁面頂端的下拉式清單中選取它。
2. 從頁面左側的側邊列選取 APIs & 驗證。
3. 在搜尋方塊中，輸入「Google Cloud Messaging for Android」，然後按一下 Google Cloud Messaging for Android 連結。
4. 按一下啟用 API。

最後，取得 API 金鑰：

1. 在 Google 開發人員主控台中，選取 APIs & 驗證 > 登入資料。
2. 在公有 API 存取下，按一下建立新金鑰。
3. 在建立新的金鑰對話方塊中，按一下伺服器金鑰。
4. 在產生的對話方塊中，按一下建立並複製顯示的 API 金鑰。您將使用此 API 金鑰稍後執行身分驗證。

使用專案 ID 在 SNS 主控台中建立平台 ARN

1. 前往 SNS 主控台。
2. 按一下畫面左側的應用程式。
3. 按一下建立平台應用程式以建立新的 SNS 平台應用程式。
4. 輸入應用程式名稱。
5. 針對推播通知平台選取 Google Cloud Messaging (GCM)。
6. 將 API 金鑰貼到標示為 API 金鑰的文字方塊中。
7. 按一下建立平台應用程式。
8. 選取您剛建立的平台應用程式，並複製應用程式 ARN。

將 SNS 的 NuGet 套件新增至您的專案

請遵循設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的步驟 4，將 Amazon Simple Notification Service NuGet 套件新增至您的專案。

建立 SNS 用戶端

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

註冊您的遠端通知應用程式

若要在 Android 上註冊遠端通知，您需要建立可接收 Google Cloud 訊息的 BroadcastReceiver。在出現提示的情況下變更以下套件名稱：

```
[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);
      }
}
```

以下是從 BroadcastReceiver 接收推播通知並在裝置的通知列上顯示通知的服務：

```
[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 主控台傳送訊息到您的端點

1. 前往 SNS 主控台 > 應用程式。
2. 選取您的平台應用程式，選取端點，然後按一下發佈至端點。
3. 在文字方塊中輸入文字訊息，然後按一下發佈訊息以發佈訊息。

傳送推播通知 (Xamarin iOS)

本文件說明如何使用 Amazon Simple Notification Service (SNS) 和適用於 .NET 和 Xamarin 的 AWS Mobile SDK，將推送通知傳送至 Xamarin iOS 應用程式。

專案設定

先決條件

開始本教學課程之前，您必須完成設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 上的所有說明。

設定 SNS 的許可

請遵循設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的步驟 2，將下列政策連接至應用程式的角色。這將給予您的應用程式存取 SNS 的適當許可：

1. 前往 IAM 主控台，然後選取您要設定的 IAM 角色。
2. 按一下連接政策，選取 AmazonSNSFullAccess 政策，然後按一下連接政策。

> **Warning**
>
> 在生產環境中不建議使用 AmazonSNSFullAccess。我們在這裡使用它，可讓您快速啟動和執行。如需指定 IAM 角色許可的詳細資訊，請參閱 IAM 角色許可概觀。

在 Apple iOS 開發人員計劃中取得成員資格

您需要在實體裝置上執行應用程式，才能接收推播通知。若要在裝置上執行應用程式，您必須擁有 Apple iOS 開發人員計劃成員資格。擁有成員資格後，您可以使用 Xcode 產生簽署身分。如需詳細資訊，請參閱 Apple 的應用程式分佈 Quick Start 文件。

建立 iOS 憑證

首先，您需要建立 iOS 憑證。然後，您需要建立為推送通知設定的佈建設定檔。若要這麼做：

1. 前往 Apple 開發人員成員中心，按一下憑證、識別符和設定檔。
2. 按一下 iOS 應用程式下的識別符，按一下網頁右上角的加號按鈕以新增 iOS 應用程式 ID，然後輸入應用程式 ID 描述。
3. 向下捲動至新增 ID 尾碼區段，然後選取明確應用程式 ID，然後輸入您的套件識別碼。
4. 向下捲動至應用程式服務區段，然後選取推送通知。
5. 按一下 Continue (繼續)。
6. 請按 Submit (提交)。
7. 按一下完成。
8. 選取您剛建立的應用程式 ID，然後按一下編輯。
9. 向下捲動至推送通知區段。按一下開發 SSL 憑證下的建立憑證。
10. 依照指示建立憑證簽署請求 (CSR)、上傳請求，以及下載將用於與 Apple Notification Service (APNS) 通訊的 SSL 憑證。
11. 返回憑證、識別符和設定檔頁面。按一下佈建設定檔下的全部。
12. 按一下右上角的加號按鈕，新增佈建設定檔。
13. 選取 iOS 應用程式開發，然後按一下繼續。
14. 選取您的應用程式 ID，然後按一下繼續。
15. 選取您的開發人員憑證，然後按一下繼續。
16. 選取您的裝置，然後按一下繼續。
17. 輸入設定檔名稱，然後按一下產生。
18. 下載並按兩下佈建檔案以安裝佈建設定檔。

如需佈建針對推送通知設定之設定檔的詳細資訊，請參閱 Apple 的設定推送通知文件。

使用憑證在 SNS 主控台中建立平台 ARN

1. 執行 KeyChain 存取應用程式，選取畫面左下角的我的憑證，然後在您產生連線至 APNS 的 SSL 憑證上按一下滑鼠右鍵，然後選取匯出。系統會提示您指定檔案的名稱和密碼來保護憑證。憑證將儲存在 P12 檔案中。
2. 前往 SNS 主控台，然後按一下畫面左側的應用程式。
3. 按一下建立平台應用程式以建立新的 SNS 平台應用程式。
4. 輸入應用程式名稱。
5. 選取 Apple Development for Push 通知平台。
6. 按一下選擇檔案，然後選取您在匯出 SSL 憑證時建立的 P12 檔案。
7. 輸入匯出 SSL 憑證時指定的密碼，然後按一下從檔案載入憑證。
8. 按一下建立平台應用程式。
9. 選取您剛建立的平台應用程式，並複製應用程式 ARN。在接下來的步驟中，您將需要此項目。

將 SNS 的 NuGet 套件新增至您的專案

請遵循設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的步驟 4，將 Amazon Simple Notification Service NuGet 套件新增至您的專案。

建立 SNS 用戶端

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

註冊您的遠端通知應用程式

若要註冊應用程式，請在 UIApplication 物件上呼叫 RegisterForRemoteNotifications，如下所示。在 AppDelegate.cs,在出現以下提示時插入您的平台應用程式 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 主控台傳送訊息到您的端點

1. 前往 SNS 主控台 > 應用程式。
2. 選取您的平台應用程式，選取端點，然後按一下發佈至端點。
3. 在文字方塊中輸入文字訊息，然後按一下發佈訊息以發佈訊息。

傳送和接收簡訊通知

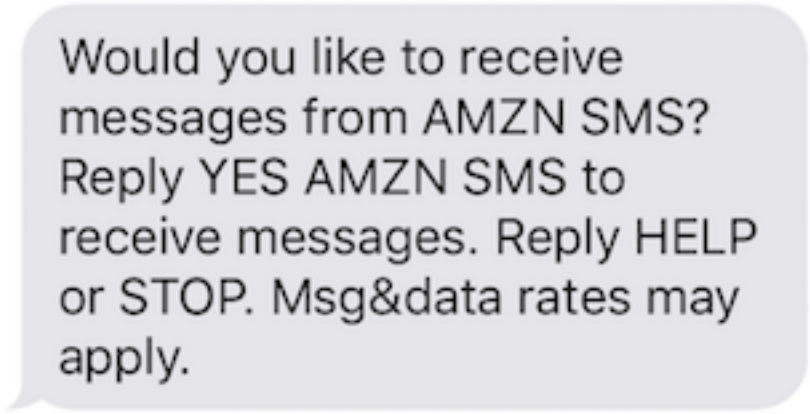
您可以使用 Amazon Simple Notification Service (Amazon SNS) 傳送和接收簡訊服務 (SMS) 通知給啟用 SMS 功能的行動電話和智慧型手機。

> **Note**
>
> 美國的電話號碼目前支援簡訊通知。SMS 訊息只能從在美國東部（維吉尼亞北部）區域建立的主題傳送。不過，您可以將訊息發佈到您在美國東部（維吉尼亞北部）區域從任何其他區域建立的主題。

建立主題

建立主題：

1. 在 Amazon SNS 主控台中，按一下建立新主題。建立新主題對話方塊隨即出現。
2. 在 Topic name (主題名稱) 方塊中，輸入主題名稱。
3. 在顯示名稱方塊中，輸入顯示名稱。主題必須指派顯示名稱，因為顯示名稱的前十 (10) 個字元用作文字訊息字首的初始部分。您輸入的顯示名稱會顯示在 SNS 傳送給使用者的確認訊息中 （以下顯示名稱為「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，因為您在下一個步驟中訂閱主題時會需要它。

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

使用 SMS 通訊協定訂閱主題

建立 SNS 用戶端，傳遞您的登入資料物件和身分集區的區域：

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

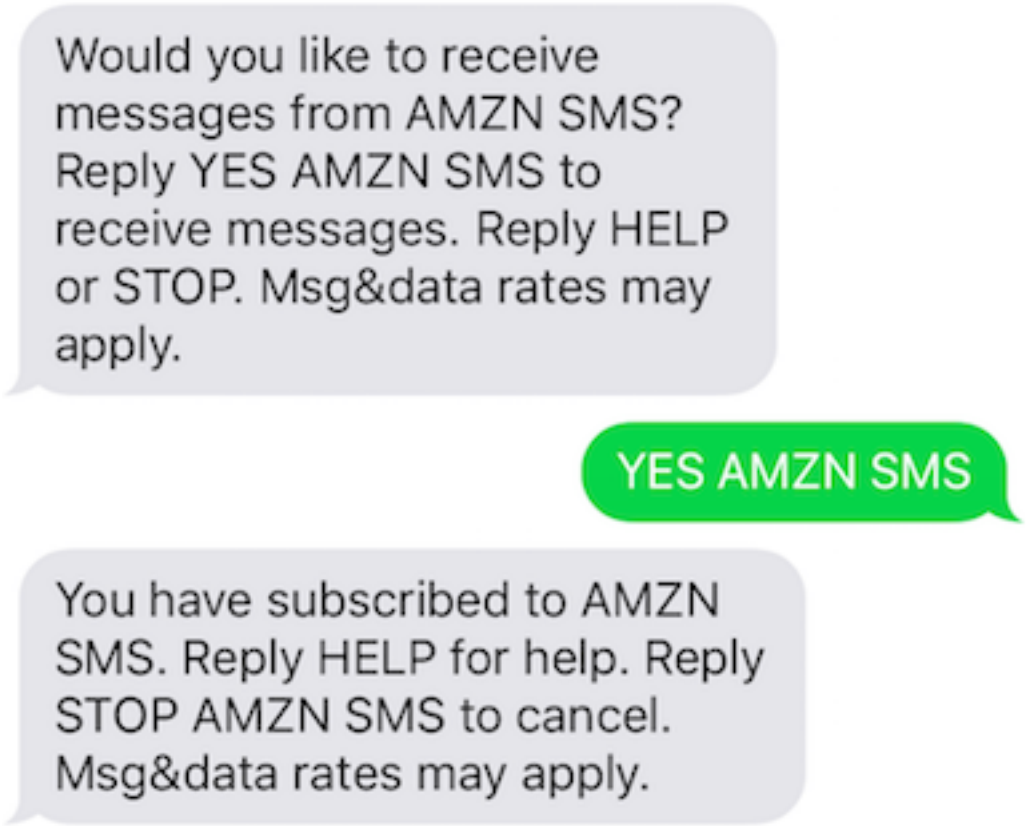
若要訂閱主題，請叫用SubscribeAsync並傳遞您要訂閱之主題的 ARN、通訊協定 (「sms」) 和電話號碼：

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

您會在訂閱回應物件中收到訂閱 Arn。您的訂閱 ARN 如下所示：

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

當裝置訂閱主題時，SNS 會傳送確認訊息至裝置，使用者必須確認他們想要接收通知，如下所示：



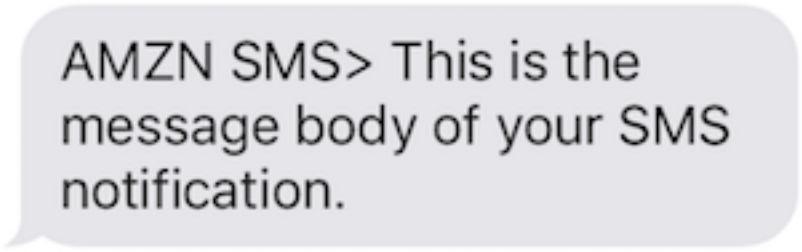
使用者訂閱主題後，他們會在您發佈到該主題時收到簡訊。

發佈訊息

若要發佈訊息至主題：

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. 按一下發佈至主題。
4. 在主旨方塊中，輸入主旨。

5. 在訊息方塊中，輸入訊息。Amazon SNS 會將您在訊息方塊中輸入的文字傳送給簡訊訂閱者，除非您也在主旨方塊中輸入文字。由於 Amazon SNS 包含顯示名稱字首與您傳送的所有簡訊，因此顯示名稱字首和訊息承載的總和不能超過 140 個 ASCII 字元或 70 個 Unicode 字元。Amazon SNS 會截斷超過這些限制的訊息。
6. 按一下 Publish message (發佈訊息)。Amazon SNS 會顯示確認對話方塊。SMS 訊息會顯示在已啟用 SMS 的裝置上，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. 設定您的端點以接收 Amazon SNS 訊息。
2. 訂閱 HTTP/HTTPS 端點至主題。
3. 確認您的訂閱。
4. 發佈通知至 主題。然後，Amazon SNS 會傳送 HTTP POST 請求，將通知的內容傳送至訂閱的端點。

設定您的 HTTP/HTTPS 端點以接收 Amazon SNS 訊息

遵循傳送 Amazon SNS 訊息至 HTTP/HTTPS 端點的步驟 1 中的指示來設定您的端點。

將您的 HTTP/HTTPS 端點訂閱到您的 Amazon SNS 主題

建立 SNS 用戶端，傳遞您的登入資料物件和身分集區的區域：

```
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。您可以使用端點的 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

訂閱端點之後，Amazon SNS 會將訂閱確認訊息傳送至端點。端點的程式碼必須從訂閱確認訊息擷取SubscribeURL值，並造訪 SubscribeURL本身指定的位置，或供您使用，以便您可以手動造訪 SubscribeURL（例如，如果使用 Web 瀏覽器）。

在確認訂閱之前，Amazon SNS 不會傳送訊息至端點。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

您可以發佈至主題，將訊息傳送至主題的訂閱。叫用PublishAsync並傳遞主題 ARN 和您的訊息。

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

SNS 故障診斷

在 Amazon SNS 主控台中使用交付狀態

Amazon SNS 主控台包含交付狀態功能，可讓您收集訊息成功和失敗交付至行動推播通知平台 (Apple (APNS)、Google (GCM)、Amazon (ADM)、Windows (WNS 和 MPNS) 和百度的意見回饋。

它也提供其他重要資訊，例如 Amazon SNS 中的停留時間。此資訊會在 Amazon SNS 透過 Amazon SNS 主控台或透過 Amazon SNS API 啟用此功能時，自動建立的 Amazon CloudWatch Log 群組中擷取。 Amazon SNS APIs

如需使用交付狀態功能的指示，請參閱 AWS Mobile 部落格上的使用 Amazon SNS 的交付狀態功能。

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

使用適用於 .NET 和 Xamarin 的 AWS Mobile SDK 時，只有幾個基本概念和最佳實務有助於了解。

- 使用 Amazon Cognito 取得 AWS 登入資料，而不是硬式編碼應用程式中的登入資料。如果您在應用程式中硬式編碼登入資料，最終可能會公開給大眾，讓其他人使用您的登入資料呼叫 AWS。如需如何使用 Amazon Cognito 取得 AWS 登入資料的指示，請參閱設定適用於 .NET 和 Xamarin 的 AWS Mobile SDK。
- 如需使用 S3 的最佳實務，請參閱 AWS 部落格上的這篇文章。
- 如需使用 DynamoDB 的最佳實務，請參閱DynamoDB 開發人員指南》中的 DynamoDB 最佳實務。DynamoDB

我們一直致力於協助客戶獲得成功和歡迎的意見回饋，因此請隨時在 AWS 論壇上張貼文章或在 GitHub 上開啟問題。

AWS 服務文件庫

適用於 .NET 和 Xamarin 的 AWS Mobile SDK 中的每個服務都有單獨的開發人員指南和服務 API 參考，提供您可能會發現有用的其他資訊。

Amazon Cognito 身分

- Cognito 開發人員指南
- Cognito Identity Service API 參考

Amazon Cognito Sync

- Cognito 開發人員指南
- Cognito Sync Service API 參考

Amazon Mobile Analytics

- Mobile Analytics 開發人員指南

- Mobile Analytics Service API 參考

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 Simply Notification Service (SNS)

- SNS 開發人員指南
- SNS 服務 API 參考

其他實用連結

- AWS 術語詞彙表
- 關於 AWS 登入資料

故障診斷

本主題說明使用適用於 .NET 和 Xamarin 的 AWS Mobile SDK 時可能遇到的問題進行故障診斷的一些想法。

確保 IAM 角色具有必要的許可

呼叫 AWS 服務時，您的應用程式應該使用來自 Cognito 身分集區的身分。集區中的每個身分都與 IAM（身分和存取管理）角色相關聯。

角色有一或多個與其相關聯的政策檔案，可指定指派給角色的使用者可存取的 AWS 資源。根據預設，每個身分集區會建立兩個角色：一個用於已驗證的使用者，另一個用於未驗證的使用者。

您需要修改現有的政策檔案，或將新的政策檔案與應用程式所需的許可建立關聯。如果您的應用程式允許已驗證和未驗證的使用者，則必須授予這兩個角色存取應用程式所需 AWS 資源的許可。

下列政策檔案顯示如何授予對 S3 儲存貯體的存取權：

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

下列政策檔案顯示如何授予 DynamoDB 資料庫的存取權：

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

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

如需指定政策的詳細資訊，請參閱 IAM 政策。

使用 HTTP Proxy Debugger

如果您應用程式呼叫的 AWS 服務具有 HTTP 或 HTTPS 端點，您可以使用 HTTP/HTTPS 代理偵錯工具來檢視請求和回應，以深入了解發生的情況。有許多 HTTP 代理偵錯工具可供使用，例如：

- Charles - Windows 和 OSX 的 Web 偵錯代理
- Fiddler - Windows 的 Web 偵錯 Proxyfidd

Charles 和 Fiddler 都需要一些組態才能檢視 SSL 加密流量，請閱讀這些工具的文件以取得進一步資訊。如果您使用的 Web 偵錯代理無法設定為顯示加密流量，請開啟 aws_endpoints_json 檔案，並將您需要偵錯的 AWS 服務 HTTP 標籤設定為 true。

文件歷史記錄

下表說明自上次發行適用於 .NET 和 Xamarin 的 AWS Mobile SDK 以來文件的重要變更。

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

變更	API 版本	描述	發行日期
已封存	2015-08-27	適用於 Xamarin 的 AWS Mobile SDK 包含在 中 適用於 .NET 的 AWS SDK。本指南參考 Mobile SDK for Xamarin 的封存版本。	2021-02-23
GA 版本	2015-08-27	GA 版本	2015-08-27
Beta 版	2015-07-28	Beta 版	rn 2015-07-28 1