開發人員指南

AWS 適用於 Unity 的 Mobile SDK

AWS 適用於 Unity 的 Mobile SDK: 開發人員指南

Table of Contents

..... vi
什麼是適用於 Unity 的 AWS Mobile SDK？ 1
相關指南和主題 1
封存的參考內容 1
相容性 2
下載適用於 Unity 的 Mobile SDK 2
適用於 Unity 的 Mobile SDK 包含哪些內容？ 2
設定適用於 Unity 的 AWS Mobile SDK 3
先決條件 3
步驟 1：下載適用於 Unity 的 AWS Mobile SDK 3
步驟 2：設定適用於 Unity 的 AWS Mobile SDK 4
建立場景 4
設定預設 AWS 服務區域 4
設定記錄資訊 4
使用 link.xml 檔案 5
步驟 3：使用 Amazon Cognito 取得身分集區 ID 6
後續步驟 6
適用於 Unity 的 AWS Mobile SDK 入門 7
Amazon Cognito 身分 7
Amazon Cognito Sync 7
使用 CognitoSyncManager 範例 7
Dynamo DB 8
使用 DynamoDB 範例 8
Mobile Analytics 9
設定 Mobile Analytics 9
使用 Mobile Analytics 範例 9
Amazon S3 10
設定 S3 預設簽章 10
使用 S3 範例 10
Amazon Simple Notification Service 11
AWS Lambda 11
Amazon Cognito 身分 12
什麼是 Amazon Cognito Identity？ 12
使用公有供應商來驗證使用者 12

使用開發人員驗證的身分 12
Amazon Cognito 同步 13
Amazon Mobile Analytics 14
整合 Amazon Mobile Analytics 14
在 Mobile Analytics 主控台中建立應用程式 14
將 Mobile Analytics 整合至您的應用程式 14
記錄獲利事件 15
記錄自訂事件 16
錄製工作階段 16
Amazon Simple Storage Service (S3) 18
建立和設定 S3 儲存貯體 18
建立 S3 儲存貯體 18
設定 S3 的許可 18
從主控台上傳檔案 19
（選用） 設定 S3 請求的簽章版本 19
建立 Amazon S3 用戶端 20
列出儲存貯體 20
列出物件 21
下載物件 21
上傳物件 22
Amazon DynamoDB 24
整合 Amazon DynamoDB 24
建立 DynamoDB 資料表 25
建立 DynamoDB 用戶端 25
描述資料表 26
儲存物件 27
建立書籍 27
擷取書籍 28
更新書籍 28
刪除書籍 29
Amazon Simple Notification Service 30
先決條件 3
設定 SNS 的許可 30
iOS 先決條件 30
Android 先決條件 31
設定適用於 iOS 的 Unity 範例應用程式 31

Unity 組態 31
iOS 組態 32
SNS 組態 33
使用 Xcode 33
Unity 範例 (iOS) 34
設定適用於 Android 的 Unity 範例應用程式 34
Unity 組態 34
Android 組態 35
SNS 組態 35
Unity 範例 (Android) 36
AWS Lambda 37
許可 37
專案設定 37
設定 AWS Lambda 的許可 37
建立新的執行角色 38
在 AWS Lambda 中建立函數 38
建立 Lambda 用戶端 39
建立請求物件 39
叫用您的 Lambda 函數 39
故障診斷 40
確保 IAM 角色具有必要的許可 40
使用 HTTP Proxy Debugger 41

適用於 Unity 的 AWS Mobile SDK 現在已包含在 中 適用於 .NET 的 AWS SDK。本指南參考適用於 Unity 的 Mobile SDK 封存版本。如需詳細資訊，請參閱 什麼是適用於 Unity 的 AWS Mobile SDK？

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 Unity 的 AWS Mobile SDK？

適用於 Unity 的 AWS Mobile SDK 現在已包含在 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 Unity 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 Unity 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 Unity 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 Unity 的 Mobile SDK 封存版本。

封存的參考內容

適用於 Unity 的封存 Mobile SDK 包含一組 .NET 類別，可讓使用 Unity 編寫的遊戲利用 AWS 服務。使用適用於 Unity 的 Mobile SDK 撰寫的應用程式可以在 iOS 或 Android 裝置上執行。

支援 AWS 的服務包括：

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

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、傳送推播通知，以及收集和分析用量資料。

相容性

適用於 Unity v3 的 Mobile SDK 與 Unity 4.6 版及更新版本相容。

適用於 Unity 的 Mobile SDK 的最新版本引入了改進功能，這可能需要您在納入專案時變更程式碼。如需這些變更的詳細資訊，請參閱前端 Web 和 AWS Mobile 部落格上的適用於 Unity 的 Mobile SDK 改進。

下載適用於 Unity 的 Mobile SDK

您也可以在此處將適用於 Unity 的 Mobile SDK 下載為 .zip 檔案。

適用於 Unity 的 Mobile SDK 包含哪些內容？

如需適用於 Unity 的 Mobile SDK 中 NuGet 套件、範例和其他檔案的完整清單，請參閱 GitHub 適用於 .NET 的 AWS SDK上的 。

設定適用於 Unity 的 AWS Mobile SDK

若要開始使用適用於 Unity 的 AWS Mobile SDK，您可以設定 SDK 並開始建立新專案，或者您可以將 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。

先決條件

在使用適用於 Unity 的 AWS Mobile SDK 之前，您將需要下列項目：

- AWS 帳戶
- Unity 4.x 或 5.x 版（如果您想要寫入在 iOS 64 位元上執行的應用程式，則需要 Uniity 4.6.4p4 或 Unity 5.0.1p3)

完成先決條件後，您需要執行下列動作才能開始使用：

1. 下載適用於 Unity 的 AWS Mobile SDK。
2. 設定適用於 Unity 的 AWS Mobile SDK。
3. 使用 Amazon Cognito 取得 AWS 登入資料。

步驟 1：下載適用於 Unity 的 AWS Mobile SDK

首先，下載適用於 Unity 的 AWS Mobile SDK。SDK 中的每個套件都需要根據套件的名稱來使用對應的 AWS 服務。例如，aws-unity-sdk-dynamodb-2.1.0.0.unitypackage 套件用於呼叫 AWS DynamoDB 服務。您可以匯入所有套件，或只匯入您要使用的套件。

1. 開啟 Unity 編輯器並建立新的空白專案，使用預設設定。
2. 選取資產 > 匯入套件 > 自訂套件。
3. 在匯入套件對話方塊中，導覽至並選取您要使用的 .unitypackage 檔案。
4. 在匯入套件對話方塊中，確定已選取所有項目，然後按一下匯入。

步驟 2：設定適用於 Unity 的 AWS Mobile SDK

建立場景

使用適用於 Unity 的 AWS Mobile SDK 時，您可以在 `Start`或 單一行為類別的 `Awake`方法中包含以下程式碼行，以開始使用：

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

從檔案功能表中選擇新場景來建立場景。

適用於 Unity 的 AWS 開發套件包含其支援的每個 AWS 服務的用戶端類別。這些用戶端是使用名為 awsconfig.xml 的檔案進行設定。下節說明 awsconfig.xml 檔案中最常用的設定。如需這些設定的詳細資訊，請參閱 Unity SDK API 參考。

設定預設 AWS 服務區域

若要為所有服務用戶端設定預設區域：

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

這會設定 Unity 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);
```

設定記錄資訊

記錄設定指定如下：

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

此設定用於在 Unity 中設定記錄。當您登入 時`UnityLogger`，框架會在內部將輸出列印到偵錯日誌。如果您想要記錄 HTTP 回應，請設定 logResponses 旗標 - 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，可以使用 LogMetricsFormat 屬性指定日誌格式，有效值為 JSON 或標準。

下列範例顯示 awsconfig.xml 檔案中最常用的設定。如需特定服務設定的詳細資訊，請參閱下列服務區段：

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

使用 link.xml 檔案

開發套件使用平台特定元件的反射。如果您使用的是 IL2CPP 指令碼後端， 一律`strip bytecode`會在 iOS 上啟用，因此您需要在組件根中具有具有下列項目`link.xml`的檔案：

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

步驟 3：使用 Amazon Cognito 取得身分集區 ID

若要在行動應用程式中使用 AWS 服務，您必須使用 Amazon Cognito Identity 取得身分集區 ID。使用 Amazon Cognito 取得身分集區 ID 可讓您的應用程式存取 AWS 服務，而不必在應用程式中內嵌私有憑證。這也可讓您設定許可，以控制使用者可存取的 AWS 服務。

若要開始使用 Amazon Cognito，您必須建立身分集區。身分集區是您的帳戶專屬的使用者身分資料存放區。每個身分集區都有可設定的 IAM 角色，可讓您指定應用程式使用者可以存取的 AWS 服務。一般而言，開發人員每個應用程式將使用一個身分集區。如需身分集區的詳細資訊，請參閱 Amazon Cognito 開發人員指南。

若要為您的應用程式建立身分集區：

1. 登入 Amazon Cognito 主控台，然後按一下建立新身分集區。
2. 輸入身分集區的名稱，並勾選核取方塊以啟用對未驗證身分的存取。按一下建立集區以建立您的身分集區。
3. 按一下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 Cognito Sync 和 Mobile Analytics 的身分集區。

下一頁顯示的程式碼會建立登入資料提供者，讓您可以輕鬆地將 Cognito Identity 與 Unity 應用程式整合。您可以將登入資料提供者物件傳遞至您正在使用的 AWS 用戶端建構函式。程式碼看起來像這樣：

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

後續步驟

- 入門：閱讀適用於 Unity 的 AWS Mobile SDK 入門，以取得開發套件中包含之服務的更詳細概觀。
- 執行示範：檢視示範常見使用案例的範例 Unity 應用程式。若要執行範例應用程式，請依照上述方式設定適用於 Unity 的 SDK，然後遵循個別範例的 README 檔案中包含的指示。
- 閱讀 API 參考：檢視適用於 Unity 的 AWS Mobile SDK 的 API 參考。
- 提問：在 AWS Mobile SDK 論壇上張貼問題或在 Github 上開啟問題。

適用於 Unity 的 AWS Mobile SDK 入門

此頁面為您提供適用於 Unity 的 AWS Mobile SDK 中每個 AWS 服務的概觀，以及如何設定 Unity 範例的說明。您必須先完成設定適用於 Unity 的 AWS Mobile SDK 頁面上的所有指示，才能開始使用下列服務。

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 Unity 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 等公有登入提供者，以及支援 OpenID Connect 的提供者來驗證使用者。Cognito 也適用於未經驗證的使用者。Cognito 提供臨時登入資料，具有您透過 Identity and Access Management (IAM) 角色指定的有限存取權限。Cognito 是透過建立與 IAM 角色相關聯的新身分集區來設定。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 Amazon Cognito 開發人員指南。

Amazon Cognito Sync

Cognito Sync 可讓您輕鬆地將使用者偏好設定或遊戲狀態等最終使用者資料儲存至 AWS 雲端，以便無論使用者使用的裝置為何，都能將其提供給使用者。Cognito 也可以在本機儲存此資料，讓您的應用程式即使在網際網路連線無法使用時也能運作。當網際網路連線可用時，您的應用程式可以將本機資料同步至雲端。

若要開始使用 Cognito Sync，請參閱 Amazon Cognito 開發人員指南。

使用 CognitoSyncManager 範例

在專案窗格中，導覽至 Assets/AWSSDK/examples/CognitoSync，然後在窗格右側選取 CognitoSync 場景以開啟場景。

若要執行範例，請按一下編輯器畫面頂端的播放按鈕。當應用程式執行時，它會顯示幾個文字方塊和按鈕，允許您輸入一些玩家資訊。下面有一系列按鈕，可在本機儲存玩家資訊、將本機玩家資訊與 Cognito 雲端同步、從 Cognito 雲端重新整理玩家資訊，以及刪除本機玩家資訊。按下每個按鈕來執行操作。範例會在遊戲畫面頂端顯示意見回饋。

若要設定 CognitoSyncManager 範例，您必須指定 Cognito 身分集區 ID。若要指定此值，請在 Unity 編輯器中，選取階層窗格中的 SyncManager，然後將其輸入至 Inspector Pane 中的 IDENTITY_POOL_ID 文字方塊。

> **Note**
>
> CognitoSyncManager 範例包含的程式碼說明如何使用 Facebook 身分提供者，請搜尋「USE_FACEBOOK_LOGIN」巨集。這需要使用適用於 Unity 的 Facebook 開發套件。如需詳細資訊，請參閱適用於 Unity 的 Facebook 開發套件。

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。

適用於 Unity 的 AWS 開發套件提供低階和高階程式庫，可用於 DynamoDB。高階程式庫包含 DynamoDB 物件映射器，可讓您將用戶端類別映射至 DynamoDB 資料表；執行各種建立、讀取、更新和刪除 (CRUD) 操作；以及執行查詢。使用 DynamoDB 物件映射器，您可以撰寫簡單、可讀取的程式碼，將物件存放在雲端。

如需 DynamoDB 的詳細資訊，請參閱 DynamoDB 開發人員指南。

如需從 Unity 應用程式使用 Dynamo DB的詳細資訊，請參閱 Amazon DynamoDB。

使用 DynamoDB 範例

在專案窗格中，導覽至 Assets/AWSSDK/ example/DynamoDB。此範例由下列場景組成：

- DynamoDBExample - 應用程式初始場景
- LowLevelDynamoDbExample - 使用低階 DynamoDBD API 的範例
- TableQueryAndScanExample - 示範如何執行查詢的範例
- HighLevelExample - 使用高階 DynamoDB API 的範例

使用組建設定對話方塊（選取檔案.組建設定開啟），將這些場景新增至組建（依上述順序顯示）。此範例會建立四個資料表：ProductCatalog、Forum、Thread、Resback。

若要執行範例，請按一下編輯器畫面頂端的播放按鈕。當應用程式執行時，它會顯示多個按鈕：

- 低階資料表操作 - 說明如何建立、列出、更新、描述和刪除資料表。
- 中階查詢和掃描操作 - 說明如何執行查詢。
- 高階物件映射器 - 說明如何建立、更新和刪除物件。

Mobile Analytics

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。適用於 Unity 的 AWS 開發套件提供與 Amazon Mobile Analytics 服務的整合。如需有關 Mobile Analytics 的資訊，請參閱 Mobile Analytics 使用者指南。如需從 Unity 應用程式使用 Mobile Analytics 的詳細資訊，請參閱 Amazon Mobile Analytics。

設定 Mobile Analytics

Mobile Analytics 會定義一些可在 awsconfig.xml 檔案中設定的設定：

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

- 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。
- 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。
- 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。
- maxRequestSize - 這是在 HTTP 請求中應傳輸至行動分析服務的位元組請求大小上限。
- allowUseDataNetwork - 指定工作階段事件是否在資料網路上傳送的布林值。

使用 Mobile Analytics 範例

在專案窗格中，導覽至資產/AWSSDK/ 範例/Mobile Analytics，然後在窗格右側選取 Amazon Mobile Analytics 範例場景以開啟場景。若要使用範例，您需要使用 Amazon Mobile Analytics 主控台新增應用程式。如需使用 Mobile Analytics 主控台的詳細資訊，請參閱 Amazon Mobile Analytics 使用者指南。

執行之前，請依照下列步驟設定範例：

1. 選取 AmazonMobileAnalyticsSample 遊戲物件。

2. 在「應用程式 ID」欄位中指定您的應用程式 ID（在 Amazon Mobile Analytics 主控台中建立）。
3. 在「Cognito Identity Pool ID」欄位中指定您的 Cognito Identity Pool ID（使用 Amazon Cognito 主控台建立）。
4. 請確定已驗證和未驗證的角色具有存取 Mobile Analytics 服務的許可。如需將政策套用至 IAM 角色的詳細資訊，請參閱管理角色。

執行範例應用程式時，請注意事件可能不會立即傳輸到後端服務。背景執行緒會在本機緩衝事件，並以定期間隔（預設值為 60 秒）將事件批次傳送至 Amazon Mobile Analytics 後端，以確保您遊戲的效能不會受到負面影響。由於 Amazon Mobile Analytics 對您的資料執行複雜的處理，在初次提交後最多 60 分鐘內，提交的事件和對應的報告可能不會顯示在 AWS 主控台中。

如需 Amazon Mobile Analytics 所提供報告的詳細資訊，請參閱報告和行動指標。

Amazon S3

Amazon Simple Storage Service (Amazon S3) 為開發人員和 IT 團隊提供安全、耐用、高度可擴展的物件儲存。從 Unity，您可以使用 S3 來存放、列出和擷取遊戲所使用的影像、影片、音樂和其他資料。

如需 S3 的詳細資訊，請參閱 Amazon S3 和 S3 入門。

如需從 Unity 應用程式使用 S3 的詳細資訊，請參閱 Amazon Simple Storage Service (S3)。

設定 S3 預設簽章

預設 S3 簽章的設定如下：

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

這是用來指定您是否應該將簽章第 4 版用於 S3 請求。

使用 S3 範例

在專案窗格中，導覽至 Assets/AWSSDK/examples/S3，然後在窗格右側選取 S3Example 場景以開啟場景。此範例說明如何列出儲存貯體、列出儲存貯體中的物件、將物件發佈至儲存貯體，以及從儲存貯體下載物件。執行之前，請依照下列步驟設定範例：

1. 在階層窗格中選取 S3 遊戲物件。

2. 在 Inspector 窗格中，輸入 S3BucketName 和 SampleFileName 的值。S3BucketName 是範例使用的儲存貯體名稱，而 S3SampleFileName 是範例將上傳至指定 S3 儲存貯體的檔案名稱。
3. 請確定已驗證和未驗證的角色具有存取您帳戶中 S3 儲存貯體的許可。如需將政策套用至 IAM 角色的詳細資訊，請參閱管理角色。

若要執行範例，請按一下編輯器畫面頂端的播放按鈕。當應用程式執行時，它會顯示多個按鈕：

- 取得物件 - 取得 AWS 帳戶中所有儲存貯體中所有物件的清單。
- 取得儲存貯體 - 取得 AWS 帳戶中所有儲存貯體的清單。
- 後物件 - 將物件上傳至指定的 S3 儲存貯體。
- 刪除物件 - 從指定的 S3 儲存貯體刪除所有物件。

範例會在遊戲畫面頂端顯示意見回饋。

Amazon Simple Notification Service

Amazon Simple Notification Service 是一種快速、靈活、全受管的推送通知服務，可讓您傳送個別訊息或散發訊息給大量收件人。Amazon Simple Notification Service 讓傳送推播通知給行動裝置使用者、電子郵件收件人，甚至傳送訊息到其他分散式服務變得簡單且經濟實惠。若要開始使用 Amazon Simple Notification Service，請參閱 Amazon Simple Notification Service。

AWS Lambda

AWS Lambda 是一種運算服務，可執行程式碼以回應請求或事件，並自動為您管理運算資源，讓您輕鬆地建置可快速回應新資訊的應用程式。AWS Lambda 函數可以直接從行動、IoT 和 Web 應用程式呼叫，並同步傳送回應，讓您輕鬆為行動應用程式建立可擴展、安全且高可用性的後端，而無需佈建或管理基礎設施。如需詳細資訊，請參閱 AWS Lambda。

Amazon Cognito 身分

什麼是 Amazon Cognito Identity？

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。Cognito 也支援開發人員驗證身分，可讓您使用自己的後端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。

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 Amazon Cognito 身分區域可用性。

使用公有供應商來驗證使用者

如需使用 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

如需開發人員驗證身分的資訊，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

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 同步跨裝置的使用者資料。若要在應用程式中使用 Cognito Sync，您必須在專案中包含適用於 Unity 的 AWS Mobile SDK。

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

Amazon Mobile Analytics

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。如需有關 Mobile Analytics 的資訊，請參閱 AWS Mobile Analytics。

整合 Amazon Mobile Analytics

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。

> **Note**
>
> 若要進一步了解如何在 主控台中使用 ，請參閱 Amazon Mobile Analytics 使用者指南。

當您在 Mobile Analytics 主控台中建立應用程式時，您需要指定 Cognito 身分集區 ID。若要建立新的 Cognito 身分集區並產生 ID，請參閱 Cognito 身分開發人員指南。

將 Mobile Analytics 整合至您的應用程式

若要從 Unity 存取 Mobile Analytics，您需要使用 陳述式：

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

最佳實務是使用 Amazon Cognito 為您的應用程式提供臨時 AWS 登入資料。這些登入資料可讓應用程式存取您的 AWS 資源。若要建立登入資料提供者，請遵循 Amazon Cognito Identity 中的指示。

使用以下資訊執行個體化 MobileAnalyticsManager 執行個體：

- cognitoIdentityPoolId - 應用程式的 Cognito 身分集區的 ID
- cognitoRegion - Cognito Identity Pool 的區域，例如 "RegionEndpoint.USEast1”
- region - Mobile Analytics 服務的區域，例如 "RegionEndpoint.USEast1”

- appId - 新增應用程式時 Mobile Analytics 主控台產生的值

使用 MobileAnalyticsClientContextConfig 來初始化**MobileAnalyticsManager**執行個體，如下列程式碼片段所示：

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

> **Note**
>
> 應用程式 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。

appId 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，然後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 Unity 的 SDK 提供 MonetizationEvent類別，可讓您產生獲利事件，以追蹤在行動應用程式內進行的購買。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，假設您的應用程式是遊戲，而且您想要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

當應用程式失去焦點時，您可以暫停工作階段。OnApplicationFocus 檢查是否暫停應用程式。如果是這樣，請呼叫 PauseSession ResumeSession，如下列程式碼片段所示：

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

```
    }
}
```

根據預設，如果使用者將焦點切換離開應用程式不到 5 秒，且切換回應用程式，則工作階段將會繼續。如果使用者將焦點切換離開應用程式 5 秒或更長時間，則會建立新的工作階段。此設定可在 awsconfig.xml 檔案中設定。如需詳細資訊，請參閱適用於 Unity 的 AWS Mobile SDK 入門中的設定 Mobile Analytics 一節。

Amazon Simple Storage Service (S3)

Amazon Simple Storage Service (Amazon S3) 為開發人員和 IT 團隊提供安全、耐用、高度可擴展的物件儲存。Unity 開發人員可以利用 S3 動態載入其遊戲所使用的資產。這可讓遊戲一開始從應用程式存放區更快速地下載。

如需 S3 的詳細資訊，請參閱 Amazon S3。

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。

> **Note**
>
> 本文件中的一些範例假設使用名為 ResultText 的文字方塊變數來顯示追蹤輸出。

建立和設定 S3 儲存貯體

Amazon S3 會將您的資源存放在 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。

建立 S3 儲存貯體

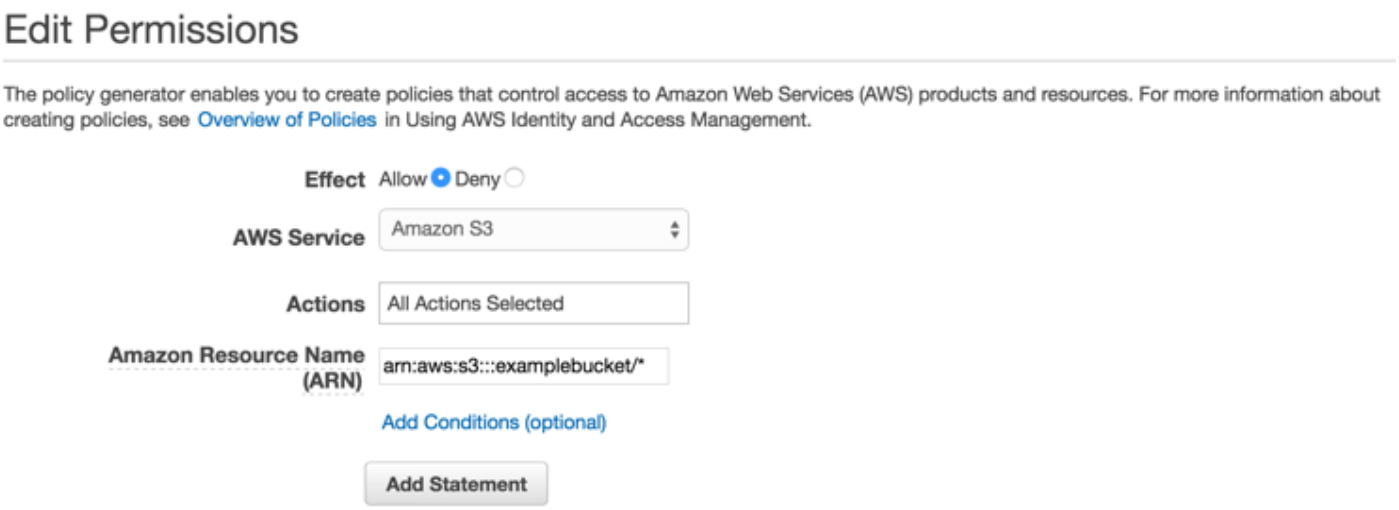
1. 登入 Amazon S3 主控台，然後按一下建立儲存貯體。
2. 輸入儲存貯體名稱，選取區域，然後按一下建立。

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經驗證）。
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，將 Amazon Resource Name (ARN) 取代為您自己的設定。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。以下顯示的設定將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。

從主控台上傳檔案

若要將測試檔案上傳至您的儲存貯體：

1. 在 S3 主控台的儲存貯體檢視中，按一下上傳。
2. 按一下新增檔案並選取要上傳的測試檔案。在本教學課程中，我們將假設您正在上傳名為 的映像`myImage.jpg`。
3. 選取測試映像後，按一下開始上傳。

（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」。

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

建立 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：

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

`AmazonS3Client` 類別是高階 S3 API 的進入點。

列出儲存貯體

若要列出 AWS 帳戶中的儲存貯體，請呼叫 `AmazonS3Client.ListBucketsAsync`方法，如下列範例程式碼所示：

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

列出物件

若要列出儲存貯體中的所有物件，請呼叫 AmazonS3Client.ListObjectsAsync方法，如下列範例程式碼所示：

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

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

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

下載物件

若要下載物件，請建立 GetObjectRequest、指定儲存貯體名稱和金鑰，並將物件傳遞給 Client.GetObjectAsync：

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

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

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

GetObjectAsync 會取得 GetObjectRequest 的執行個體、回呼和 AsyncOptions 執行個體。回呼的類型必須為：`AmazonServiceCallback<GetObjectRequest, GetObjectResponse>`。AsyncOptions 執行個體是選用的。如果指定，它會判斷回呼是否會在主執行緒上執行。

上傳物件

若要上傳物件，請將物件寫入串流、建立新的 PostObjectRequest，並指定金鑰、儲存貯體名稱和串流資料。

適用於 Unity 的 AWS 開發套件使用不支援 HTTP PUT 操作的 WWW HTTP 用戶端。若要將物件上傳至 S3 儲存貯體，您需要使用 S3 的瀏覽器貼文，如下所示。

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

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

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

```
        CannedACL = S3CannedACL.Private
    };

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

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

Amazon DynamoDB

Amazon DynamoDB 是一種快速、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。如需 DynamoDB 的相關資訊，請參閱 Amazon DynamoDB。

適用於 Unity 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。

> **Note**
>
> 本文件中的一些範例假設使用名為 ResultText 的文字方塊變數來顯示追蹤輸出。

整合 Amazon DynamoDB

若要在 Unity 應用程式中使用 DynamoDB，您需要將 Unity 開發套件新增至您的專案。如果您尚未這麼做，請下載適用於 Unity 的 SDK，並遵循設定適用於 Unity 的 AWS Mobile SDK 中的指示。建議使用 Amazon Cognito Identity 為您的應用程式提供臨時 AWS 登入資料。這些登入資料可讓您的應用程式存取 AWS 服務和資源。

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

```
}
```

此政策應套用至指派給 Cognito 身分集區的角色，但您需要將 **`Resource`**值取代為 DynamoDB 資料表的正確 ARN。Cognito 會自動為您的新身分集區建立角色，您可以在 IAM 主控台將政策套用至此角色。

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，請參閱使用 IAM 控制對 DynamoDB 資源的存取。

建立 DynamoDB 資料表

現在我們已設定許可和登入資料，讓我們為應用程式建立 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 `Bookstore`做為資料表的名稱。
3. 選取雜湊做為主索引鍵類型。
4. 選取數字並`id`輸入雜湊屬性名稱。按一下 Continue (繼續)。
5. 再次按一下繼續以略過新增索引。
6. 將讀取容量設定為 `10`，將寫入容量設定為 `5`。按一下 Continue (繼續)。
7. 輸入通知電子郵件，然後按一下繼續以建立輸送量警示。
8. 按一下 Create (建立)。DynamoDB 將建立您的資料庫。

建立 DynamoDB 用戶端

若要讓應用程式與 DynamoDB 資料表互動，我們需要用戶端。我們可以建立預設 DynamodDB 用戶端，如下所示：

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

AmazonDynamoDBClient 類別是 DynamoDB API 的進入點。類別提供執行個體方法來建立、描述、更新和刪除資料表，以及其他操作。內容會在用戶端上新增另一層抽象，並可讓您使用其他功能，例如物件持久性模型。

描述資料表

若要取得 DynamoDB 資料表的描述，我們可以使用以下程式碼：

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

       }, null);
   }
```

在此範例中，我們會建立用戶端和 DescribeTableRequest 物件，將資料表的名稱指派給 **`TableName`** 屬性，然後將請求物件傳遞至 AmazonDynamoDBClient 物件上的 DescribeTableAsync 方法。DescribeTableAsync 也會接受委派，在非同步操作完成時呼叫該委派。

> **Note**
>
> AmazonDynamoDBClient 上的所有非同步方法都會接受非同步操作完成時呼叫的委派。

儲存物件

若要將物件儲存至 DynamoDB，請使用 AmazonDynamoDBClient 物件的 SaveAsync<T> 方法，其中 T 是您儲存的物件類型。

我們稱之為資料庫「書店」，為了符合該主題，我們將實作記錄書本相關屬性的資料模型。以下是定義資料模型的類別。

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

當然，對於真實的書店應用程式，我們需要其他欄位，才能用於作者和價格等項目。Book 類別以【DynamoDBTable】 屬性裝飾，這會定義 Book 將寫入的資料庫資料表物件類型。Book 類別的每個執行個體的金鑰都是使用 【DynamoDBHashKey】 屬性來識別。屬性是以 【DynamoDBProperty】 屬性識別，這些屬性會指定要寫入屬性之資料庫資料表中的資料欄。模型就位後，我們可以編寫一些方法來建立、擷取、更新和刪除 Book 物件。

建立書籍

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

    // Save the book.
```

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

擷取書籍

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

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

    }, null);
}
```

更新書籍

```
private void PerformUpdateOperation()
```

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

刪除書籍

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

Amazon Simple Notification Service

使用 Amazon Simple Notification Service (SNS) 和 Unity SDK，您可以撰寫可接收行動推播通知的 iOS 和 Android 應用程式。如需 SNS 的相關資訊，請參閱 Amazon Simple Notification Service。

本主題將逐步引導您設定適用於 Unity 的 AWS 開發套件範例應用程式 SNSExample.unity,以透過 Amazon SNS 接收行動推播通知。

您可以使用 SNSExample.unity 範例建立 iOS 和 Android 應用程式。iOS 和 Android 之間的組態步驟不同，請閱讀以下適用於您目標平台的適當區段。

先決條件

使用此解決方案需要下列先決條件。

設定 SNS 的許可

當您建立 Cognito 身分集區時，會產生兩個 IAM 角色：

- Cognito/_<Identity-Pool-Name>Auth_DefaultRole - 已驗證使用者的預設 IAM 角色
- Cognito/_<Identity-Pool-Name>Unauth_DefaultRole - 未驗證使用者的預設 IAM 角色

您必須新增許可，才能存取 Amazon SNS 服務至這些角色。若要執行此作業：

1. 瀏覽至 IAM 主控台，然後選取要設定的 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。

> **Note**
>
> 不建議在生產環境中使用 AmazonSNSFullAccess，我們在此處使用它來允許您快速啟動和執行。如需指定 IAM 角色許可的詳細資訊，請參閱 IAM 角色許可概觀。

iOS 先決條件

- Apple iOS 開發人員計劃的成員資格

- 產生簽署身分
- 建立為推送通知設定的佈建設定檔

您需要在實體裝置上執行應用程式，才能接收推送通知。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。接下來，您將需要為推送通知設定的佈建設定檔，以取得詳細資訊，請參閱 Apple 的設定推送通知文件。

Android 先決條件

- 安裝 Android 開發套件
- 安裝 JDK
- android-support-v4.jar
- google-play-services.jar

設定適用於 iOS 的 Unity 範例應用程式

開啟 Unity 編輯器並建立新的專案。選取資產/匯入套件/自訂套件，然後選取 aws-unity-sdk-sns-2.0.0.1.unitypackage，以匯入適用於 Unity 套件的 AWS 開發套件。確定已選取匯入套件對話方塊中的所有項目，然後按一下匯入。

Unity 組態

執行下列步驟來設定 Unity 專案：

1. 在專案窗格中，導覽至資產/AWSSDK/範例，並開啟 SNSExample 場景。
2. 在階層窗格中，選取 SNSExample。
3. 在檢測器窗格中，指定您的 Cognito 身分集區 ID。
4. 請注意，iOS 平台應用程式 ARN 有一個文字方塊，稍後將產生該資訊。
5. 選取檔案/組建設定，在組建設定對話方塊中，按一下組建清單中場景下方的新增目前按鈕，將目前場景新增至組建。
6. 在平台下，選取 iOS 並按一下玩家設定...按鈕，在 Unity 編輯器的檢查器窗格中，按一下 iPhone 圖示並向下捲動至識別區段，並指定套件識別符。

iOS 組態

執行下列步驟來設定範例以設定 iOS 特定設定：

1. 在 Web 瀏覽器中，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. 按一下繼續按鈕。
6. 按一下提交按鈕。
7. 按一下完成按鈕。
8. 選取您剛建立的應用程式 ID，然後按一下編輯按鈕。
9. 向下捲動至推送通知區段。
10. 按一下開發 SSL 憑證下的建立憑證按鈕。
11. 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。
12. 返回憑證、識別符和設定檔網頁，按一下佈建設定檔下的全部。
13. 按一下右上角的加號按鈕，新增佈建設定檔。
14. 選取 iOS 應用程式開發，然後按一下繼續按鈕。
15. 選取您的應用程式 ID，然後按一下繼續按鈕。
16. 選取您的開發人員憑證，然後按一下繼續按鈕。
17. 選取您的裝置，然後按一下繼續按鈕。
18. 輸入設定檔名稱，然後按一下產生按鈕。
19. 下載並按兩下佈建檔案以安裝佈建設定檔。

新增設定檔後，您可能需要重新整理 Xcode 中的佈建設定檔。在 Xcode 中：

1. 選取 Xcode/Preferences 選單項目。
2. 選取帳戶索引標籤，選取您的 Apple ID，然後按一下檢視詳細資訊。
3. 按一下對話方塊左下角的重新整理按鈕，以重新整理您的佈建設定檔，並確保顯示新的設定檔。

SNS 組態

1. 執行 KeyChain 存取應用程式，選取畫面左下角的我的憑證，在產生的 SSL 憑證上按一下滑鼠右鍵以連線至 APNS，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. 在 Web 瀏覽器中，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. 針對推送通知平台選取 Apple 推送通知服務沙盒 (APNS_SANDBOX)。
6. 按一下選擇檔案，然後選取匯出 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登入資料。
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。
10. 返回 Unity Editor 中的專案，在階層窗格中選取 SNSExample，然後在檢查器窗格中將平台應用程式 ARN 貼到標示為 iOS 平台應用程式 ARN 的文字方塊中。
11. 選取檔案/建置設定，然後按一下建置按鈕，這會建立 Xcode 專案。

使用 Xcode

1. 開啟 Xcode 專案，然後在 Project Navigator 中選取專案。
2. 確認套件識別符已正確設定
3. 確認您的 Apple 開發人員帳戶已在 團隊中指定 - 這是佈建設定檔生效的必要項目。
4. 建置專案並在您的裝置上執行。
5. 點選註冊通知，點選確定以允許通知，應用程式會顯示您的裝置字符

在 SNS 主控台中，按一下應用程式，選取您的平台應用程式，然後按一下建立平台端點，然後輸入應用程式顯示的裝置字符。

此時，您的應用程式、APNS 和 NSN 已完全設定。您可以選取平台應用程式、選取端點，然後按一下發佈至端點，將推播通知傳送至您的裝置。

Unity 範例 (iOS)

此範例會建立 CognitoAWSCredentials 執行個體，以產生暫時性限制範圍登入資料，允許應用程式呼叫 AWS 服務。它也會建立 AmazonSimpleNotificationServiceClient 執行個體，以與 SNS 通訊。應用程式會顯示兩個標示為註冊通知和取消註冊的按鈕。

當點選註冊通知按鈕時，會呼叫 `RegisterDevice()`方法。 會`RegisterDevice()`呼叫 `UnityEngine.iOS.NotificationServices.RegisterForNotifications`，指定要使用的通知類型 （警示、音效或徽章）。它也會對 APNS 進行非同步呼叫，以取得裝置字符。由於未定義回呼，`CheckForDeviceToken`因此會重複呼叫 （最多 10 次） 以檢查裝置字符。

擷取字符`AmazonSimpleNotificationServiceClient.CreatePlatformEndpointAsync()`時，呼叫 來建立 SNS 平台應用程式的端點。

範例現在已設定為接收推送通知。您可以瀏覽至 SNS 主控台，按一下頁面左側的應用程式，選取平台應用程式，選取端點，然後按一下發佈至端點。選取要使用的端點，然後按一下發佈至端點。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。

設定適用於 Android 的 Unity 範例應用程式

開啟 Unity 編輯器並建立新的專案。選取資產/匯入套件/自訂套件，然後選取 aws-unity-sdk-sns-2.0.0.1.unitypackage，以匯入適用於 Unity 套件的 AWS 開發套件。確定已選取匯入套件對話方塊中的所有項目，然後按一下匯入。

Unity 組態

執行下列步驟來設定 Unity 專案：

1. 在專案窗格中，導覽至資產/AWSSDK/範例，並開啟 SNSExample 場景。
2. 在階層窗格中，選取 SNSExample。
3. 在檢測器窗格中，指定您的 Cognito 身分集區 ID。
4. 請注意，Android 平台應用程式 ARN 和 Google 主控台專案 ID 有一個文字方塊，稍後您將產生該資訊。
5. 選取檔案/組建設定，在組建設定對話方塊中，按一下組建清單中場景下方的新增目前按鈕，將目前場景新增至組建。
6. 在平台下，選取 Android 並按一下玩家設定... 按鈕，在 Unity 編輯器的檢查器窗格中，按一下 Android 圖示並向下捲動至識別區段，並指定套件識別符。

7. 將 android-support-v4.jar 和 google-play-services.jar 複製到專案窗格中的 Assets/Plugins/Android 目錄。

如需尋找 android-support-v4.jar 位置的詳細資訊，請參閱 Android Support Library Setup。如需如何尋找 google-play-services.jar 的詳細資訊，請參閱適用於 Android 設定的 Google APIs。

Android 組態

首先新增 Google API 專案：

1. 在 Web 瀏覽器中，前往 Google 開發人員主控台，按一下建立專案。
2. 在新增專案方塊中，輸入專案名稱，記下專案編號（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，否則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」，然後按一下下方的 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 API 金鑰稍後執行身分驗證。

SNS 組態

1. 在 Web 瀏覽器中，前往 SNS 主控台，然後按一下畫面左側的應用程式。
2. 按一下建立平台應用程式以建立新的 SNS 平台應用程式。
3. 輸入應用程式名稱

4. 針對推播通知平台選取 Google Cloud Messaging (GCM)
5. 將 API 金鑰貼到標示為 API 金鑰的文字方塊中。
6. 按一下建立平台應用程式
7. 選取您剛建立的平台應用程式，並複製應用程式 ARN。
8. 返回 Unity Editor 中的專案，在階層窗格中選取 SNSExample，然後在檢查器窗格中將平台應用程式 ARN 貼到標記 Android 平台應用程式 ARN 的文字方塊中，並將您的專案編號貼到標記 Google 主控台專案 ID 的文字方塊中。
9. 將 Android 裝置連接到電腦，選取檔案/建置設定，然後按一下建置和執行。

Unity 範例 (Android)

此範例會建立 CognitoAWSCredentials 執行個體，以產生暫時性限制範圍登入資料，允許應用程式呼叫 AWS 服務。它也會建立 AmazonSimpleNotificationServiceClient 執行個體，以與 SNS 通訊。

應用程式會顯示兩個標示為註冊通知和取消註冊的按鈕。當點選註冊通知按鈕時，會呼叫 `RegisterDevice()`方法。 會`RegisterDevice()`呼叫 `GCM.Register`，其會向 GCM 註冊應用程式。GCM 是在範例程式碼中定義的類別。它會進行非同步呼叫，以向 GCM 註冊應用程式。

當呼叫回呼
時`AmazonSimpleNotificationServiceClient.CreatePlatformEndpointAsync`，呼叫 來建立平台端點以接收 SNS 訊息。

範例現在已設定為接收推送通知。您可以瀏覽至 SNS 主控台，按一下頁面左側的應用程式，選取平台應用程式，選取端點，然後按一下發佈至端點。選取要使用的端點，然後按一下發佈至端點。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。

AWS Lambda

AWS Lambda 是一種運算服務，可執行程式碼以回應請求或事件，並自動為您管理運算資源，讓您輕鬆地建置可快速回應新資訊的應用程式。AWS Lambda 函數可以直接從行動、IoT 和 Web 應用程式呼叫，並同步傳送回應，讓您輕鬆地為行動應用程式建立可擴展、安全且高可用性的後端，而無需佈建或管理基礎設施。

AWS Lambda 可以執行您的 Lambda 函數，以回應下列其中一項：

- 事件，例如離散更新（例如 Amazon S3 或 CloudWatch 提醒中物件建立的事件）或串流更新（例如，網站點擊串流或來自連線裝置的輸出）。
- 來自自訂應用程式的 JSON 輸入或 HTTPS 命令。

AWS Lambda 只有在需要時才會執行程式碼，可自動從每天數項請求擴展成每秒數千項請求。透過這些功能，您可以使用 Lambda 輕鬆為 Amazon S3 和 Amazon DynamoDB 等 AWS 服務建立觸發條件、處理存放在 Amazon Kinesis 中的串流資料，或建立您自己的後端，以 AWS 規模、效能和安全性運作。

若要進一步了解 AWS Lambda 的運作方式，請參閱 AWS Lambda：運作方式。

許可

有兩種與 Lambda 函數相關的許可類型：

- 執行許可 — 您的 Lambda 函數存取帳戶中其他 AWS 資源所需的許可。您可以透過建立稱為執行角色的 IAM 角色來授予這些許可。
- 調用許可 — 事件來源與 Lambda 函數通訊所需的許可。根據調用模型（推送或提取模型），您可以使用執行角色或資源政策（與您的 Lambda 函數相關聯的存取政策）授予這些許可。

專案設定

設定 AWS Lambda 的許可

1. 開啟 AWS IAM 主控台。
2. 將此自訂政策連接至您的角色，讓您的應用程式可以呼叫 AWS Lambda。

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

建立新的執行角色

此角色適用於您將在下一個步驟中建立的 Lambda 函數，並決定該函數可以存取哪些 AWS 資源。

1. 開啟 AWS IAM 主控台。
2. 按一下角色。
3. 按一下建立新角色。
4. 依照畫面上的指示，選取 Lambda 函數需要存取的服務和對應的政策。例如，如果您希望 Lambda 函數建立 S3 儲存貯體，您的政策將需要 S3 的寫入存取權。
5. 按一下建立角色。

在 AWS Lambda 中建立函數

1. 開啟 AWS Lambda 主控台。
2. 按一下建立 Lambda 函數。
3. 按一下略過以略過建立藍圖。
4. 在下一個畫面上設定您自己的函數。輸入函數名稱、描述，然後選擇您的執行時間。根據您選擇的執行時間，遵循畫面上的指示。將新建立的執行角色指派給函數，以指定您的執行許可。
5. 完成後，請按一下下一步。
6. 按一下建立函數。

建立 Lambda 用戶端

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

建立請求物件

建立請求物件以指定叫用類型和函數名稱：

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

叫用您的 Lambda 函數

呼叫呼叫，傳遞請求物件：

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

故障診斷

由於適用於 Unity 的 AWS 開發套件所使用的 Unity.WWW 類別限制，當呼叫 AWS 服務時發生問題時，不會傳回詳細的錯誤訊息。本主題說明疑難排解此類問題的一些想法。

確保 IAM 角色具有必要的許可

呼叫 AWS 服務時，您的應用程式會使用來自 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，指定指派給角色的使用者可存取的 AWS 資源。根據預設，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 - OSX 的 Web 偵錯代理
- Fiddler - 適用於 Windows 的 Web 偵錯 Proxyfidd

> **Important**
>
> 執行 Charles Web 偵錯代理時，Cognito 登入資料提供者有已知問題，導致登入資料提供者無法正常運作。

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 除錯代理無法設定為顯示加密流量，請開啟 aws_endpoints_json 檔案 （位於 AWSUnitySDK/AWSCore/Resources)，並將您需要除錯的 AWS 服務 HTTP 標籤設為 true