Manuel du développeur Xamarin

Kit SDK AWS Mobile

Kit SDK AWS Mobile: Manuel du développeur Xamarin

Table of Contents

Le SDK AWS mobile pour Xamarin est désormais inclus dans le. AWS SDK pour .NET Ce guide fait référence à la version archivée du SDK mobile pour Xamarin.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.

Qu'est-ce que le SDK AWS Mobile pour .NET and Xamarin ?

Le SDK AWS mobile pour Xamarin est inclus dans le. SDK pour .NET Pour plus d'informations, consultez le Manuel du développeur AWS SDK pour .NET .

Ce guide n'est plus mis à jour ; il fait référence à la version archivée du SDK mobile pour Xamarin.

Guides et sujets connexes

- Pour le développement d'applications frontales et mobiles, nous vous recommandons d'utiliser AWS Amplify.
- Pour des considérations particulières relatives à l'utilisation de AWS SDK pour .NET pour vos applications Xamarin, consultez la section Considérations spéciales relatives à l'assistance Xamarin dans le guide du développeur.AWS SDK pour .NET
- À des fins de référence, vous pouvez trouver la version archivée du SDK AWS mobile pour Xamarin sur. GitHub

Contenu de référence archivé

Le SDK AWS Mobile archivé pour .NET and Xamarin fournit un ensemble de bibliothèques .NET, des exemples de code et de la documentation pour aider les développeurs à créer des applications mobiles connectées pour :

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

Les applications mobiles écrites à l'aide du SDK AWS Mobile pour .NET et du SDK Xamarin font APIs appel à une plateforme native, ce qui leur donne l'apparence d'applications natives. Les bibliothèques .NET du SDK fournissent des wrappers C# autour d'AWS REST. APIs

Qu'est-ce qui est inclus dans le SDK AWS Mobile pour .NET and Xamarin ?

Les services AWS actuellement pris en charge incluent, entre autres :

- Amazon Cognito
- Amazon S3
- Amazon DynamoDB
- Amazon Mobile Analytics
- Amazon Simple Notification Service

Ces services vous permettent d'authentifier les utilisateurs, d'enregistrer les données relatives aux joueurs et aux jeux, de conserver des objets dans le cloud, de recevoir des notifications Push, et de collecter et d'analyser les données d'utilisation.

Le SDK AWS Mobile pour .NET et Xamarin vous permet également d'utiliser la plupart des services AWS pris en charge par le SDK AWS pour .NET. Ce manuel du développeur détaille les services AWS spécifiques au développement mobile. Pour en savoir plus sur le kit SDK AWS pour .NET, voir :

- AWS SDK for .NET Getting Started Guide
- Manuel du développeur du kit SDK AWS pour .NET
- Référence d'API du kit SDK AWS pour .NET

Compatibilité

Le SDK AWS Mobile pour .NET et Xamarin est fourni sous forme de bibliothèque de classes portable (PCL). Support PCL a été ajouté dans Xamarin.Android 4.10.1 et Xamarin.iOS 7.0.4. Les projets de bibliothèque portable sont intégrés à Visual Studio.

IDEs

Pour plus d'informations sur l'utilisation IDEs de la version archivée du SDK Xamarin, consultez. Configuration du kit SDK AWS Mobile pour .NET and Xamarin

Comment puis-je obtenir le SDK AWS Mobile pour .NET and Xamarin ?

Pour obtenir le SDK AWS Mobile pour .NET and Xamarin, consultez Configuration du SDK AWS Mobile pour .NET and Xamarin. Le SDK AWS Mobile pour .NET et Xamarin NuGet est distribué sous forme de packages. Vous trouverez une liste complète des packages de services AWS dans les packages du SDK AWS NuGet ou dans le référentiel AWS SDK for GitHub .NET.

A propos des services mobiles AWS

Amazon Cognito Identity

Tous les appels renvoyant vers AWS nécessitent la saisie d'informations d'identification AWS. Au lieu de coder en dur vos informations d'identification dans vos applications, nous vous recommandons d'utiliser Amazon Cognito Identity pour fournir les informations d'identification AWS à votre application. Suivez les instructions de la section Configurer le kit SDK AWS Mobile pour .NET et Xamarin pour obtenir des informations d'identification AWS via Amazon Cognito.

Cognito vous permet également d'authentifier les utilisateurs à l'aide de fournisseurs de connexion publics comme Amazon, Facebook, Twitter et Google, ainsi que des fournisseurs prenant en charge OpenID Connect. Cognito fonctionne également avec des utilisateurs non authentifiés. Cognito fournit des informations d'identification temporaires avec des droits d'accès limités, que vous spécifiez avec un rôle IAM (Identity and Access Management). Cognito est configuré en créant un pool d'identités qui est associé à un rôle IAM. Le rôle IAM indique à quoi resources/services votre application peut accéder.

Pour savoir comment utiliser Cognito Identity, consultez la section Configurer le kit SDK AWS Mobile pour .NET et Xamarin.

Pour en savoir plus sur Cognito Identity, consultez la page Amazon Cognito Identity.

Amazon Cognito Sync

Cognito Sync est un service AWS et une bibliothèque client qui permet la synchronisation des données utilisateur liées à une application sur différents appareils. Vous pouvez utiliser l'API Cognito Sync pour synchroniser les données de profil utilisateur pour différents appareils et fournisseurs de connexion comme Amazon, Facebook et Google, et votre propre fournisseur d'identités personnalisées.

Pour savoir comment utiliser Cognito Sync, consultez la page Synchroniser les données utilisateur avec Cognito Sync.

Pour plus d'informations sur Cognito Sync, consultez la page Amazon Cognito Sync.

Mobile Analytics

Amazon Mobile Analytics vous permet de collecter, de visualiser et de comprendre les données d'utilisation de vos applications mobiles. Les rapports fournissent des mesures relatives aux utilisateurs actifs, aux sessions, à la fidélisation, aux revenus intégrés aux applications et aux événements personnalisés. Ils peuvent être filtrés par plateforme et par période. Conçu pour évoluer avec votre activité, Amazon Mobile Analytics permet de collecter et de traiter des milliards d'événements provenant de millions de points de terminaison.

Pour savoir comment utiliser Mobile Analytics, consultez la page Suivi des données d'utilisation des applications avec Amazon Mobile Analytics.

Pour plus d'informations sur Mobile Analytics, consultez la page Amazon Mobile Analytics.

Dynamo DB

Amazon DynamoDB est un service de base de données non relationnelle rapide, économique, très évolutif et hautement disponible. DynamoDB permet de s'affranchir des limites habituelles du dimensionnement de stockage de données, tout en conservant une faible latence et des performances prévisibles.

Pour commencer à utiliser Dynamo DB, consultez la page Stocker et récupérer des données avec DynamoDB.

Pour plus d'informations sur Dynamo DB, consultez la page Amazon DynamoDB.

Amazon Simple Notification Service

Amazon Simple Notification Service (SNS) est un service de notification Push rapide, flexible et entièrement géré, qui vous permet d'envoyer des messages individuels ou de diffuser des messages à un grand nombre de destinataires. Amazon Simple Notification Service permet d'envoyer des notifications Push de manière simple et économique à des utilisateurs d'appareils mobiles ou titulaires d'adresses e-mail, et même d'envoyer des messages à d'autres services distribués.

Pour savoir comment utiliser SNS pour Xamarin iOS, consultez la page Recevoir des notifications Push via SNS (Xamarin iOS).

Pour savoir comment utiliser SNS pour Xamarin Android, consultez la page Recevoir des notifications Push via SNS (Xamarin Android).

Pour plus d'informations sur SNS, consultez la page Amazon Simple Notification Service (SNS).

Configuration du kit SDK AWS Mobile pour .NET and Xamarin

Vous pouvez configurer le SDK AWS Mobile pour .NET et Xamarin et commencer à créer un nouveau projet ou vous pouvez intégrer le SDK à un projet existant. Vous pouvez également cloner et exécuter les exemples pour comprendre comment fonctionne le kit de développement logiciel. Suivez ces étapes pour configurer et commencer à utiliser le SDK AWS Mobile pour .NET and Xamarin.

Prérequis

Avant de pouvoir utiliser le SDK AWS Mobile pour .NET and Xamarin, vous devez effectuer les opérations suivantes :

- Créer un compte AWS.
- Installez Xamarin.

Après avoir exécuté les prérequis :

1. Obtenez des informations d'identification AWS à l'aide d'Amazon Cognito.
2. Définissez les autorisations requises pour chaque service AWS que vous allez utiliser dans votre application.
3. Créez un projet dans votre IDE.
4. Installez le kit SDK AWS Mobile pour .NET and Xamarin.
5. Configurez le SDK AWS Mobile pour .NET and Xamarin.

Étape 1 : Obtention des informations d'identification AWS

Pour appeler AWS dans votre application, vous devez d'abord obtenir les informations d'identification AWS. Pour ce faire, utilisez Amazon Cognito, un service AWS qui permet à votre application d'accéder aux services du SDK sans avoir à intégrer vos informations d'identification AWS privées dans l'application.

Pour commencer à utiliser Amazon Cognito, vous devez créer un pool d'identités. Un groupe d'identités est une banque d'informations propre à votre compte et identifiée par un ID de groupe d'identités unique qui ressemble à ceci :

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

1. Connectez-vous à la console Amazon Cognito, choisissez Gérer les identités fédérées et choisissez Créer un groupe d'identités.
2. Saisissez un nom pour votre pool d'identités, puis cochez la case autorisant l'accès pour les identités non authentifiées. Choisissez Créer un groupe pour créer votre pool d'identités.
3. Choisissez Autoriser pour créer les deux rôles par défaut associés à votre groupe d'identités, un pour les utilisateurs non authentifiés et l'autre pour les utilisateurs authentifiés. Ces rôles par défaut permettent à votre pool d'identités d'accéder à Amazon Cognito Sync et Amazon Mobile Analytics.

En règle générale, vous utilisez un seul pool d'identités par application.

Après avoir créé votre pool d'identités, vous pouvez obtenir des informations d'identification AWS en créant un objet `CognitoAWSCredentials` (en lui transmettant votre ID de pool d'identités), puis en le transmettant au constructeur d'un client AWS comme suit :

```
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
);
```

Étape 2 : Définition des autorisations

Vous devez définir des autorisations pour tous les services AWS que vous souhaitez utiliser dans votre application. Tout d'abord, vous devez comprendre comment AWS voit les utilisateurs de votre application.

Lorsqu'une personne utilise votre application et appelle AWS, AWS lui affecte une identité. Le pool d'identités que vous avez créé au cours de l'Étape 1 correspond à l'emplacement dans lequel AWS stocke ces identités. Il existe deux types d'identités : authentifiées et non authentifiées. Les identités authentifiées appartiennent aux utilisateurs qui sont authentifiés par un fournisseur de connexion public (par exemple, Facebook, Amazon, Google). Les identités non authentifiées appartiennent aux utilisateurs invités.

Chaque identité est associée à un AWS Identity and Access Management rôle. À l'étape 1, vous avez créé deux rôles IAM, l'un pour les utilisateurs authentifiés et l'autre pour les utilisateurs non authentifiés. Chaque rôle IAM est associé à une ou plusieurs politiques qui spécifient les services AWS auxquels les identités attribuées à ce rôle peuvent accéder. Par exemple, l'exemple de politique suivant accorde l'accès à un compartiment Amazon S3. :

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

Pour définir des autorisations pour les services AWS que vous souhaitez utiliser dans votre application, modifiez la politique associée aux rôles.

1. Accédez à la console IAM et choisissez des rôles. Entrez le nom du pool d'identités dans la zone de recherche. Choisissez le rôle IAM que vous souhaitez configurer. Si votre application autorise les utilisateurs authentifiés et non authentifiés, vous devez accorder des autorisations aux deux rôles.
2. Cliquez sur Attacher la stratégie, sélectionnez la stratégie de votre choix, puis cliquez sur Attacher la stratégie. Les politiques par défaut pour les rôles IAM que vous avez créés permettent d'accéder à Amazon Cognito Sync et à Mobile Analytics.

Pour plus d'informations sur la création de stratégies ou pour choisir dans une liste de stratégies existantes, consultez Stratégies IAM.

Étape 3 : Créer un projet

Windows

Vous pouvez utiliser Visual Studio pour développer votre application.

OS X

Vous pouvez utiliser Visual Studio pour développer vos applications. Le développement iOS à l'aide de Xamarin nécessite l'accès à un Mac pour exécuter votre application. Pour plus d'informations, consultez la page Installation de Xamarin.iOS sous Windows.

> **Note**
>
> L'IDE Rider commercial multiplateforme JetBrains inclut le support de Xamarin sur les plateformes Windows et Mac.

Étape 4 : Installation du kit SDK AWS Mobile pour .NET and Xamarin

Windows

Option 1 : Installer à l'aide de la console du gestionnaire de package

Le SDK AWS Mobile pour .NET et Xamarin se compose d'un ensemble d'assemblages .NET. Pour installer le SDK AWS Mobile pour .NET et Xamarin, exécutez la commande install-package pour chaque package dans la console Package Manager. Par exemple, pour installer Cognito Identity, exécutez les éléments suivants :

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

Les packages AWS Core Runtime et Amazon Cognito Identity sont requis pour tous les projets. Voici une liste complète des noms de package pour chaque service.

Service	Nom du package
AWS Core Runtime	AWSSDK.Noyau
Amazon Cognito Sync	AWSSDK.CognitoSync
Amazon Cognito Identity	AWSSDK.CognitoIdentity
Amazon DynamoDB	AWSSDK.Dynamo DBv2
Amazon Mobile Analytics	AWSSDK.MobileAnalytics
Amazon S3	AWSSDKS.3
Amazon SNS	AWSSDK.SimpleNotificationService

Pour inclure un package préliminaire, incluez l'argument de ligne de commande `-Pre` lors de l'installation du package, comme suit :

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

Vous trouverez une liste complète des packages de services AWS dans les packages du SDK AWS NuGet ou dans le référentiel AWS SDK for GitHub .NET.

Option 2 : Installer à l'aide de votre IDE

Dans Visual Studio

1. Cliquez avec le bouton droit sur le projet, puis cliquez sur Gérer les NuGet packages.
2. Recherchez le nom du package que vous souhaitez ajouter à votre projet. Pour inclure les NuGet packages de prélancement, choisissez Inclure la prélocation. Vous trouverez une liste complète des packages de services AWS sur NuGet les packages du SDK AWS à l'adresse.
3. Choisissez le package, puis choisissez Install.

Mac (OS X)

Dans Visual Studio

1. Effectuez un clic droit sur le dossier des packages, puis choisissez Add Packages (Ajouter des packages).
2. Recherchez le nom du package que vous souhaitez ajouter à votre projet. Pour inclure les packages de pré-version, choisissez Afficher les NuGet packages de pré-version. Vous trouverez une liste complète des packages de services AWS sur NuGet les packages du SDK AWS à l'adresse.
3. Cochez la case en regard du package souhaité, puis choisissez Add Package (Ajouter le package).

> **Important**
>
> Si vous développez à l'aide d'une bibliothèque de classes portable, vous devez également ajouter le NuGet package AWSSDK .Core à tous les projets dérivés de la bibliothèque de classes portable.

Étape 5 : Configuration du SDK AWS Mobile pour .NET and Xamarin

Définir la connexion

Vous définissez les paramètres de journalisation à l'aide de la classe `Amazon.AWSConfigs` et la classe `Amazon.Util.LoggingConfig`. Vous pouvez les trouver dans l'assembly `AWSSdk.Core`, disponible via le gestionnaire de package Nuget dans Visual Studio. Vous pouvez placer les paramètres de journalisation du code dans la méthode `OnCreate` dans le fichier `MainActivity.cs` pour les applications Android ou le fichier `AppDelegate.cs` pour les applications iOS. Vous devez également ajouter des instructions `using Amazon` et `using Amazon.Util` aux fichiers .cs.

Configurez les paramètres de journalisation comme suit :

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

Lorsque vous vous connectez à SystemDiagnostics, le framework imprime en interne la sortie sur la System.Console. Si vous souhaitez consigner les réponses HTTP, définissez l'indicateur `LogResponses`. Les valeurs peuvent être Toujours, Jamais ou OnError.

Vous pouvez également consigner les métriques de performance des demandes HTTP à l'aide de la propriété `LogMetrics`. Le format de journal peut être spécifié à l'aide de la propriété `LogMetricsFormat`. Les valeurs valides sont JSON ou standard.

Définir le point de terminaison de la région

Configurez la région par défaut pour tous les clients du service comme suit :

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

Ce paramètre définit la région par défaut pour tous les clients de service figurant dans le kit de développement logiciel. Vous pouvez remplacer ce paramètre en spécifiant explicitement la région au moment de la création d'une instance du client de service, comme suit :

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

Configurer les paramètres de proxy HTTP

Si votre réseau est derrière un proxy, vous pouvez configurer les paramètres proxy pour les demandes HTTP comme suit.

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

Corriger le décalage d'horloge

Cette propriété détermine si le kit de développement logiciel doit corriger le décalage d'horloge client en déterminant le temps de serveur correct et en réemettant la demande avec l'heure appropriée.

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

Ce champ est défini si un appel de service a généré une exception et si le kit SDK a déterminé qu'il existait une différence entre l'heure locale et celle du serveur.

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

Pour en savoir plus sur le décalage d'horloge, consultez l'article relatif à la correction du décalage d'horloge sur le blog AWS.

Étapes suivantes

Maintenant que vous avez configuré le SDK AWS Mobile pour .NET et Xamarin, vous pouvez :

- Mise en route. Consultez Getting Started with the AWS Mobile SDK for .NET and Xamarin pour obtenir des instructions de démarrage rapide sur la façon d'utiliser et de configurer les services du SDK AWS Mobile pour .NET and Xamarin.
- Exploration des rubriques de service. Découvrez chaque service et son fonctionnement dans le SDK AWS Mobile pour .NET and Xamarin.
- Exécution des démonstrations. Référez-vous à nos exemples d'applications Xamarin, qui illustrent quelques cas d'utilisation courants. Pour exécuter les exemples d'applications, configurez le SDK AWS Mobile pour .NET et Xamarin comme décrit précédemment, puis suivez les instructions contenues dans les fichiers README des échantillons individuels.
- Apprenez le APIs. Consultez le | sdk-xamarin-ref |_.
- Posez des questions : publiez des questions sur les forums du kit SDK AWS Mobile ou soulevez un problème sur GitHub.

Commencer à utiliser le kit SDK AWS Mobile pour .NET and Xamarin

Le SDK AWS Mobile pour .NET and Xamarin fournit les bibliothèques, les exemples et la documentation nécessaires pour appeler les services AWS à partir d'applications Xamarin.

Avant de commencer à utiliser les services ci-dessous, vous devez suivre toutes les instructions de la section Configurer le kit SDK AWS Mobile pour .NET et Xamarin.

Ces rubriques de mise en route vous guideront tout au long des étapes suivantes :

Rubriques

- Stocker et récupérer des fichiers avec Amazon S3
- Synchroniser les données utilisateur avec Cognito Sync
- Stocker et récupérer des données avec DynamoDB
- Suivi des données d'utilisation des applications avec Amazon Mobile Analytics
- Recevoir des notifications Push via SNS (Xamarin iOS)
- Recevoir des notifications Push via SNS (Xamarin Android)

Pour plus d'informations sur les autres appareils AWS Mobile SDKs, consultez le SDK AWS Mobile.

Stocker et récupérer des fichiers avec Amazon S3

Amazon Simple Storage Service (Amazon S3) offre aux développeurs d'applications mobiles un espace de stockage d'objets sécurisé, durable et hautement évolutif. Amazon S3 est facile à utiliser et doté d'une interface de services Web simple pour stocker et récupérer toute quantité de données depuis n'importe où sur le Web.

Le didacticiel ci-dessous explique comment intégrer le S3 TransferUtility, un utilitaire de haut niveau permettant d'utiliser S3 avec votre application. Pour plus d'informations sur l'utilisation de S3 à partir d'applications Xamarin, consultez la page Amazon Simple Storage Service (S3).

Configuration du projet

Prérequis

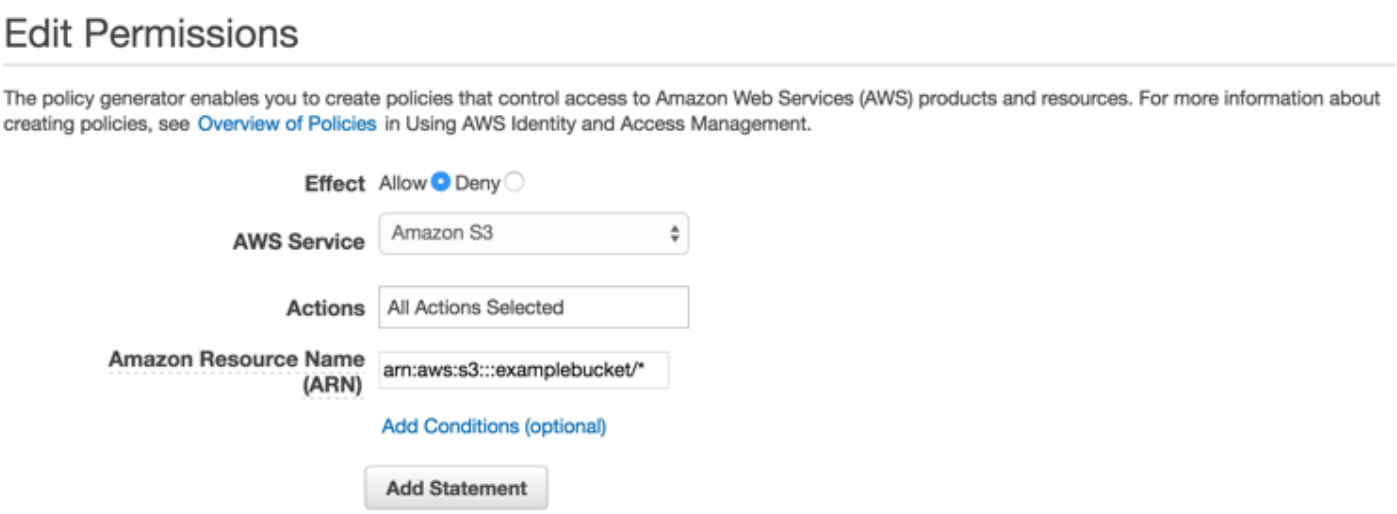
Avant de commencer ce didacticiel, vous devez suivre toutes les instructions de la section Configurer un kit SDK AWS Mobile pour .NET et Xamarin.

Ce didacticiel suppose également que vous avez déjà créé un compartiment S3. Pour créer un compartiment S3, consultez la page Console AWS S3.

Définir les autorisations pour S3

La stratégie de rôle IAM par défaut autorise votre application à accéder à Amazon Mobile Analytics et Amazon Cognito Sync. Pour que votre pool d'identités Cognito puisse accéder à Amazon S3, vous devez modifier les rôles de ce pool d'identités.

1. Accédez à la console Identity and Access Management et cliquez sur Rôles dans le volet de gauche.
2. Entrez le nom du pool d'identités dans la zone de recherche. Deux rôles seront répertoriés : un premier pour les utilisateurs authentifiés et un second pour les utilisateurs non authentifiés.
3. Cliquez sur le rôle pour les utilisateurs non authentifiés (unauth sera ajouté au nom du pool d'identités).
4. Cliquez sur Créer une stratégie de rôle, sélectionnez Générateur de stratégies, puis cliquez sur Sélectionner.
5. Sur la page Modifier les autorisations, entrez les paramètres affichés dans l'image suivante, en remplaçant le nom de ressource Amazon (ARN) par le vôtre. L'ARN d'un compartiment S3 ressemble à ceci : `arn:aws:s3:::examplebucket/*`. Il se compose de la région dans laquelle se trouve le compartiment et du nom du compartiment. Les paramètres ci-dessous accordent au pool d'identités complet l'accès à toutes les actions pour le compartiment spécifié.



1. Cliquez sur le bouton Ajouter une instruction, puis sur Étape suivante.
2. L'Assistant vous indiquera la configuration qui aura été générée. Cliquez sur Apply Policy.

Pour plus d'informations sur l'octroi d'accès à S3, consultez Octroi d'accès à un compartiment Amazon S3.

Ajoutez un NuGet package pour S3 à votre projet

Suivez l'étape 4 des instructions de la section Configuration du kit SDK AWS Mobile pour .NET et Xamarin pour NuGet ajouter le package S3 à votre projet.

(facultatif) Configurer la version de Signature pour les requêtes S3

Chaque interaction avec Amazon S3 est authentifiée ou anonyme. AWS utilise les algorithmes Signature Version 4 ou Signature Version 2 pour authentifier les appels au service.

Toutes les nouvelles régions AWS créées après janvier 2014 prennent en charge Signature Version 4 uniquement. Cependant, de nombreuses régions plus anciennes prennent encore en charge les demandes de Signature Version 4 et Signature Version 2.

Si votre compartiment se trouve dans l'une des régions qui ne prennent pas en charge les demandes Signature version 2 répertoriées sur cette page, vous devez configurer le AWSConfigs S3. UseSignatureVersion4 propriétés à « vrai » comme ça :

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

Pour plus d'informations sur les versions d'AWS Signature, consultez la page Demandes d'authentification (AWS Signature Version 4).

Initialisation du client S3 TransferUtility

Créez un client S3, passez-lui l'objet des informations d'identification AWS, puis passez le client S3 à l'utilitaire de transfert, comme suit :

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

Charger un fichier dans Amazon S3

Pour charger un fichier dans S3, appelez `Upload` sur l'objet Transfer Utility (Utilitaire de transfert) avec les paramètres suivants :

- `file` : nom de chaîne du fichier que vous souhaitez charger
- `bucketName` : nom de chaîne du compartiment S3 destiné au stockage du fichier

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

Le code ci-dessus suppose qu'il existe un fichier dans le répertoire Environment. SpecialFolder. ApplicationData. Les chargements utilisent automatiquement la fonctionnalité de téléchargement en plusieurs parties de S3 sur les fichiers volumineux pour améliorer le débit.

Télécharger un fichier à partir d'Amazon S3

Pour télécharger un fichier depuis S3, appelez `Download` sur l'objet Transfer Utility (Utilitaire de transfert) avec les paramètres suivants :

- `file` : nom de chaîne du fichier que vous souhaitez télécharger
- `bucketName` : nom de chaîne du compartiment S3 à partir duquel vous voulez télécharger le fichier
- `key` : chaîne représentant le nom de l'objet S3 (un fichier dans le cas présent) à télécharger

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

Pour plus d'informations sur l'accès à Amazon S3 à partir d'une application Xamarin, consultez la page Amazon Simple Storage Service (S3).

Synchroniser les données utilisateur avec Cognito Sync

Avec Amazon Cognito Sync, vous pouvez enregistrer facilement dans le cloud AWS les données relatives aux utilisateurs mobiles, telles que les préférences d'une application ou le statut d'un jeu, sans avoir à écrire un code en backend ni à gérer d'infrastructure. Les données peuvent être enregistrées en local sur les périphériques des utilisateurs, afin de permettre aux applications de fonctionner même lorsque ces appareils sont hors ligne. Vous pouvez aussi synchroniser ces données sur les différents périphériques d'un utilisateur pour lui offrir une expérience homogène, quel que soit l'appareil utilisé.

Le didacticiel ci-dessous explique comment intégrer le service Sync à votre application.

Configuration du projet

Prérequis

Avant de commencer ce didacticiel, vous devez suivre toutes les instructions de la section Configurer un kit SDK AWS Mobile pour .NET et Xamarin.

Accorder l'accès à vos ressources Cognito Sync

La stratégie par défaut associée aux rôles non authentifiés et authentifiés que vous avez créés au cours de l'installation accordent à votre application un droit d'accès à Cognito Sync. Aucune autre configuration n'est requise.

Ajouter un NuGet package pour Cognito Sync à votre projet

Suivez l'étape 4 des instructions de la section Configuration du kit SDK AWS Mobile pour .NET et Xamarin pour SyncManager NuGet ajouter le package Cognito à votre projet.

Initialisez le CognitoSyncManager

Communiquez votre fournisseur d'informations d'identification Amazon Cognito initialisé au constructeur `CognitoSyncManager` :

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

Synchronisation des données utilisateur

Pour synchroniser les données utilisateur non authentifiées :

1. Créez un ensemble de données.
2. Ajoutez des données utilisateur à l'ensemble de données.
3. Synchronisez l'ensemble de données avec le cloud.

Créer un ensemble de données

Créez une instance de `Dataset`. La méthode openOrCreate Dataset est utilisée pour créer un nouvel ensemble de données ou pour ouvrir une instance existante d'un ensemble de données stocké localement sur l'appareil :

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

Ajouter des données utilisateur à l'ensemble de données

Les données utilisateur sont ajoutées sous la forme de paires clé/valeur :

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

Les ensembles de données Cognito fonctionnent comme des dictionnaires, avec des valeurs accessibles par clé :

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

Synchroniser l'ensemble de données

Pour synchroniser un ensemble de données, appelez sa méthode synchronize :

```
dataset.SynchronizeAsync();

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

Toutes les données écrites dans des ensembles de données sont stockées localement jusqu'à ce que l'ensemble de données ait été synchronisé. Le code de cette section suppose que vous utilisez une identité Cognito non authentifiée. Ainsi, lorsque les données utilisateur sont synchronisées avec le cloud, elles sont stockées par appareil. L'appareil est associé à un ID. Lorsque les données utilisateur sont synchronisées sur le cloud, elles sont associées à cet ID d'appareil.

Pour plus d'informations sur Cognito Sync, consultez la page Amazon Cognito Sync.

Stocker et récupérer des données avec DynamoDB

Amazon DynamoDB est un service de base de données non relationnelle rapide, économique, très évolutif et hautement disponible. DynamoDB permet de s'affranchir des limites habituelles du dimensionnement de stockage de données, tout en conservant une faible latence et des performances prévisibles.

Le didacticiel ci-dessous explique comment intégrer le modèle de persistance des objets DynamoDB à votre application, afin de stocker les objets dans DynamoDB.

Configuration du projet

Prérequis

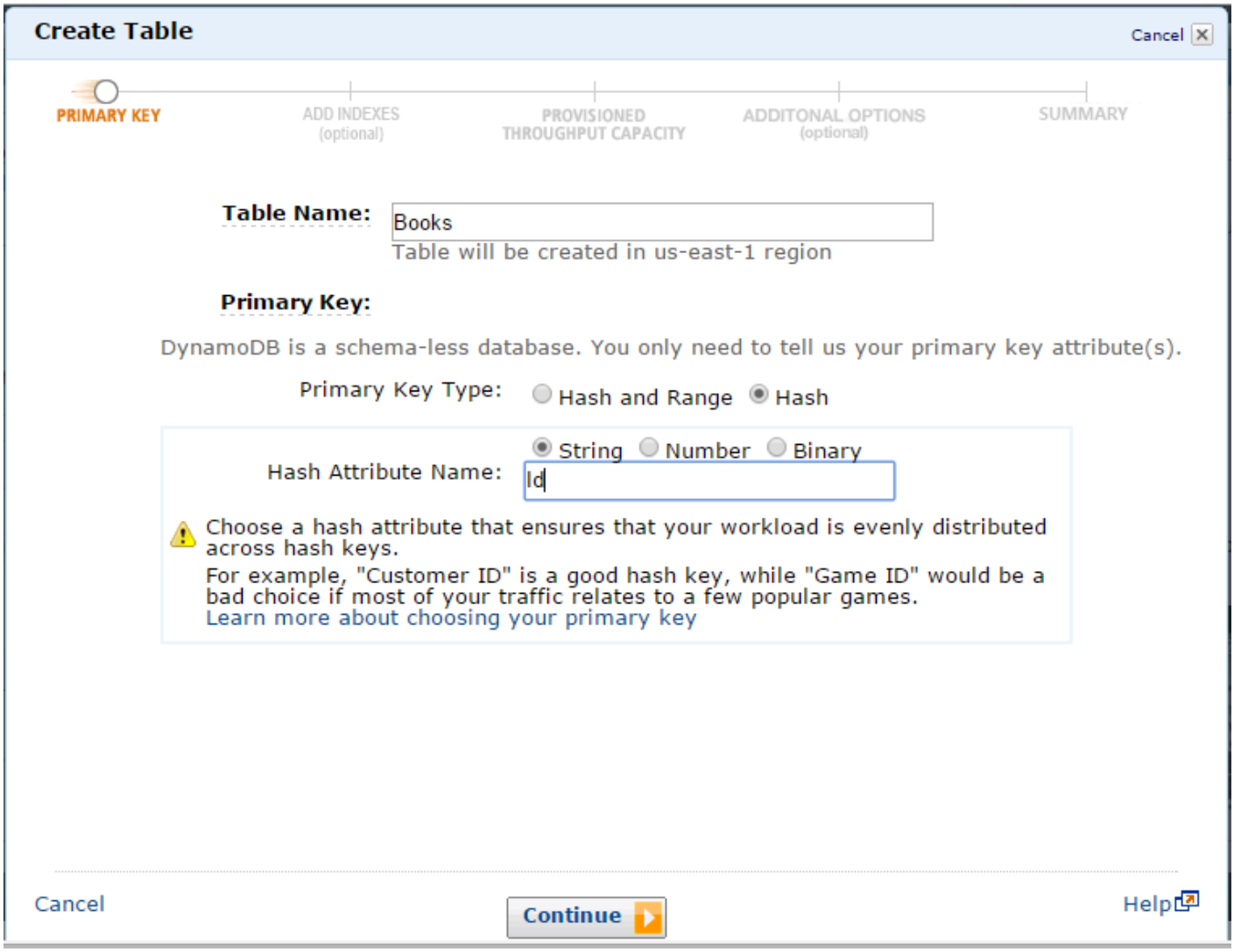
Avant de commencer ce didacticiel, vous devez suivre toutes les instructions de la section Configurer un kit SDK AWS Mobile pour .NET et Xamarin.

Créer une table DynamoDB

Pour pouvoir lire et écrire des données dans une base de données DynamoDB, vous devez d'abord créer une table. Lors de la création de la table, vous devez spécifier la clé primaire. La clé primaire se compose d'un attribut de hachage et d'un attribut de plage facultatif. Pour en savoir plus sur

l'utilisation des attributs primaires et des attributs de plage, reportez-vous à la section Utilisation de tables.

1. Accédez à la console DynamoDB et cliquez sur Créer une table. L'Assistant de création de table s'affiche.
2. Indiquez le nom de votre table, le type de clé primaire (hachage) et le nom de l'attribut de hachage (« Id »), comme indiqué ci-dessous, puis cliquez sur Continuer :

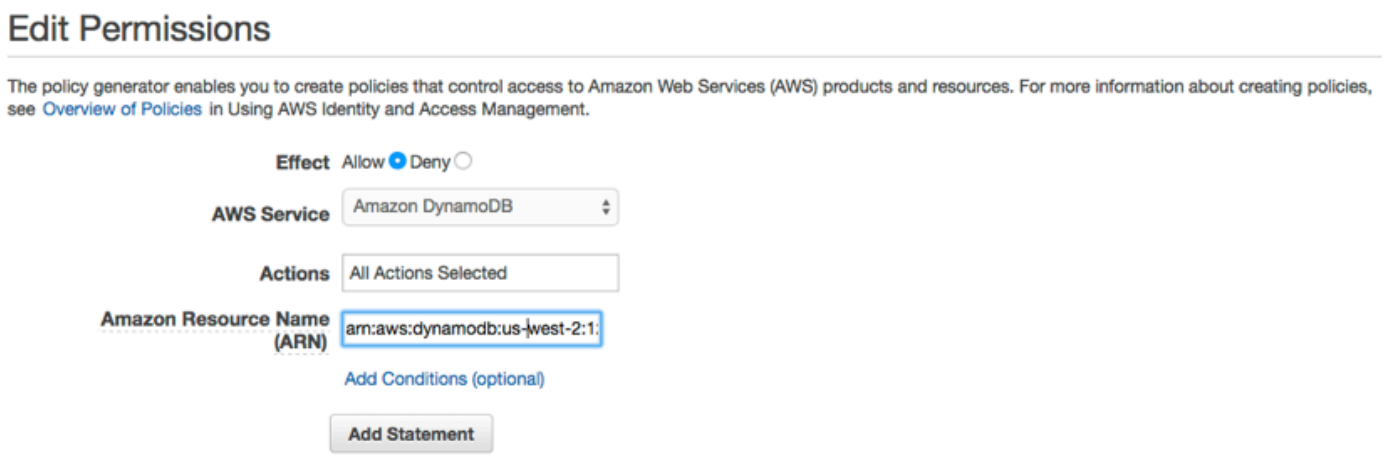


3. Laissez les champs modifiables de l'écran suivant vides et cliquez sur Continuer.
4. Acceptez les valeurs par défaut pour les champs Unités de capacité en lecture et Unités de capacité en écriture et cliquez sur Continuer.
5. Dans l'écran suivant, entrez votre adresse e-mail dans la zone de texte Envoyer les notifications à : et cliquez sur Continuer. L'écran de vérification s'affiche.
6. Cliquez sur Create. La création de votre table peut prendre quelques minutes.

Définir les autorisations pour DynamoDB

Pour que votre groupe d'identités puisse accéder à Amazon DynamoDB, vous devez modifier les rôles de ce groupe d'identités.

1. Accédez à la console Identity and Access Management et cliquez sur Rôles dans le volet de gauche. Recherchez le nom de votre groupe d'identités. Deux rôles seront indiqués : un pour les utilisateurs non authentifiés et un autre, pour les utilisateurs authentifiés.
2. Cliquez sur le rôle correspondant aux utilisateurs non authentifiés (signalé par le suffixe « unauth » à la suite du nom de votre groupe d'identités) et cliquez sur Créer une stratégie de rôle.
3. Sélectionnez Générateur de stratégies et cliquez sur Sélectionner.
4. Sur la page Modifier les autorisations, saisissez les paramètres indiqués dans l'illustration suivante. L'ARN (Amazon Resource Name) d'une table DynamoDB se présente comme suit : `arn:aws:dynamodb:us-west-2:123456789012:table/Books`. Il se compose de la région dans laquelle se trouve la table, du numéro de compte AWS du titulaire et du nom de la table au format `table/Books`. Pour plus d'informations sur la spécification ARNs, consultez Amazon Resource Names for DynamoDB.



5. Cliquez sur Ajouter une instruction, puis sur Étape suivante. L'Assistant vous indiquera la configuration générée.
6. Cliquez sur Apply Policy.

Ajouter un NuGet package pour DynamoDB à votre projet

Suivez l'étape 4 des instructions de la section Configuration du SDK AWS Mobile pour .NET and Xamarin pour NuGet ajouter le package DynamoDB à votre projet.

Initialiser AmazonDynamo DBClient

Transmettez votre fournisseur d'informations d'identification Amazon Cognito initialisé et votre région AmazonDynamoDB au constructeur, puis transmettez le client au Dynamo : DBContext

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

Créer une classe

Pour écrire une ligne dans la table, définissez une classe qui contiendra les données de votre ligne. La classe doit également contenir des propriétés qui indiquent les données des attributs de la ligne et seront mises en correspondance avec la table DynamoDB créée dans la console. La déclaration de classe suivante correspond à une classe de ce type :

```
[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; }
}
```

Enregistrer un élément

Pour enregistrer un élément, commencez par créer un objet :

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

Puis enregistrez-le :

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

Pour mettre à jour une ligne, modifiez l'instance de la classe `DDTableRow` et appelez `AWSDynamoObjectMapper.save()` comme indiqué ci-dessus.

Récupérer un élément

Récupérer un élément à l'aide d'une clé primaire :

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

Mettre à jour un élément

Pour mettre à jour un élément :

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

Supprimer un élément

Pour supprimer un élément :

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

Pour en savoir plus sur l'accès à DynamoDB depuis une application Xamarin, consultez la page Amazon DynamoDB.

Suivi des données d'utilisation des applications avec Amazon Mobile Analytics

Amazon Mobile Analytics permet de mesurer l'utilisation d'une application et le chiffre d'affaires qu'elle génère. En suivant les tendances principales, comme celles concernant les nouveaux utilisateurs par rapport à ceux déjà inscrits, les revenus de l'application, la fidélisation des utilisateurs,

et les événements de comportements personnalisés inhérents à l'application, vous pouvez prendre des décisions guidées par les données afin d'améliorer l'implication des utilisateurs et la monétisation de votre app.

Le didacticiel ci-dessous explique comment intégrer Mobile Analytics à votre application.

Configuration du projet

Prérequis

Avant de commencer ce didacticiel, vous devez suivre toutes les instructions de la section Configurer un kit SDK AWS Mobile pour .NET et Xamarin.

Créer une application dans la console Mobile Analytics

Accédez à la console Amazon Mobile Analytics et créez une application. Notez la valeur `appId` car vous en aurez besoin par la suite. Lorsque vous créez une application dans la console Mobile Analytics, vous devez spécifier l'ID de votre groupe d'identités. Pour savoir comment créer un pool d'identités, consultez la section Configurer le kit SDK AWS Mobile pour .NET et Xamarin.

Pour en savoir plus sur le fonctionnement de la console, consultez le manuel de l'utilisateur Amazon Mobile Analytics.

Définir les autorisations pour Mobile Analytics

La stratégie par défaut, associée aux rôles créés au moment de la configuration, accorde à votre application des droits d'accès à Mobile Analytics. Aucune autre configuration n'est requise.

Ajoutez un NuGet Package pour Mobile Analytics à votre projet

Suivez l'étape 4 des instructions de la section Configuration du kit SDK AWS Mobile pour .NET et Xamarin pour ajouter le package Mobile NuGet Analytics à votre projet.

Configurer les paramètres Mobile Analytics

Mobile Analytics définit certains paramètres qui peuvent être configurés dans le fichier awsconfig.xml :

```
var config = new MobileAnalyticsManagerConfig();
config.AllowUseDataNetwork = true;
config.DBWarningThreshold = 0.9f;
```

```
config.MaxDBSize = 5242880;
config.MaxRequestSize = 102400;
config.SessionTimeout = 5;
```

- AllowUseDataNetwork - Un booléen qui indique si les événements de session sont envoyés sur le réseau de données.
- DBWarningSeuil - Il s'agit de la limite de taille de la base de données qui, une fois atteinte, générera des journaux d'avertissement.
- Max DBSize : il s'agit de la taille de la SQLIte base de données. Lorsque la base de données atteint sa taille maximale, tous les nouveaux événements sont ignorés.
- MaxRequestSize - Il s'agit de la taille maximale de la demande en octets qui doit être transmise dans une requête HTTP au service d'analyse mobile.
- SessionTimeout - Il s'agit de l'intervalle de temps entre le passage d'une application en arrière-plan et le moment où la session peut être interrompue.

Les paramètres ci-dessus constituent les valeurs par défaut pour chaque élément de configuration.

Initialiser MobileAnalyticsManager

Pour initialiser votre MobileAnalyticsManager, faites appel GetOrCreateInstance à votre`MobileAnalyticsManager`, en transmettant vos informations d'identification AWS, votre région, l'ID de votre application Mobile Analytics et votre objet de configuration facultatif :

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

Suivre les événements de session

Xamarin Android

Remplacez les méthodes `OnPause()` et `OnResume()` de l'activité pour enregistrer les événements de session.

```
protected override void OnResume()
```

```
{
    manager.ResumeSession();
    base.OnResume();
}

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

Cela doit être fait pour chaque activité dans votre application.

Xamarin iOS

Dans votre AppDelegate .cs :

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

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

Pour de plus amples informations sur Mobile Analytics, consultez la page Amazon Mobile Analytics.

Recevoir des notifications Push via SNS (Xamarin iOS)

Ce document explique comment envoyer des notifications push à une application Xamarin iOS à l'aide d'Amazon Simple Notification Service (SNS) et du SDK AWS Mobile pour .NET and Xamarin.

Configuration du projet

Prérequis

Avant de commencer ce didacticiel, vous devez suivre toutes les instructions de la section Configurer un kit SDK AWS Mobile pour .NET et Xamarin.

Définir les autorisations pour SNS

Suivez l'étape 2 de la section Configurer le kit SDK AWS Mobile pour .NET et Xamarin afin d'attacher la stratégie susmentionnée aux rôles de votre application. Votre application bénéficiera ainsi des autorisations appropriées pour accéder à SNS :

1. Accédez à la console IAM et sélectionnez le rôle IAM que vous souhaitez configurer.
2. Cliquez sur Attacher la politique, sélectionnez la politique Amazon SNSFull Access et cliquez sur Attacher la politique.

> **Warning**
>
> L'utilisation SNSFull d'Amazon Access n'est pas recommandée dans un environnement de production. Nous l'utilisons ici pour vous permettre d'être opérationnel rapidement. Pour en savoir plus sur la définition d'autorisations pour un rôle IAM, consultez Présentation des autorisations des rôles IAM.

Obtenir l'adhésion au programme Apple iOS pour les développeurs

Pour recevoir des notifications Push, vous devez exécuter votre application sur un appareil physique. Pour exécuter votre application sur un appareil, vous devez adhérer au programme Apple iOS pour les développeurs. Une fois que vous disposez d'un abonnement, vous pouvez utiliser Xcode pour générer une identité de signature. Pour en savoir plus, consultez la documentation App Distribution Quick Start publiée par Apple.

Créer un certificat iOS

Tout d'abord, vous devez créer un certificat iOS. Ensuite, vous devez créer un profil de mise en service configuré pour recevoir des notifications Push. Pour ce faire :

1. Accédez au centre Apple Developer et cliquez sur Certificates, Identifiers & Profiles (Certificats, identifiants et Profils).
2. Cliquez sur Identifiers sous iOS Apps, cliquez sur le bouton Plus situé dans l'angle supérieur droit de la page pour ajouter un nouvel identifiant iOS App ID, puis saisissez une description de l'ID d'application.
3. Faites défiler la page jusqu'à la section Add ID Suffix, puis choisissez Explicit App ID et saisissez votre identifiant de groupe.

4. Faites défiler jusqu'à la section App Services et sélectionnez Push Notifications.
5. Cliquez sur Continuer.
6. Cliquez sur Soumettre.
7. Cliquez sur Done.
8. Sélectionnez l'ID d'application que vous venez de créer, puis cliquez sur Edit (Modifier).
9. Faites défiler jusqu'à la section Push Notifications. Cliquez sur Create Certificate (Créer certificat) sous Development SSL Certificate (Certificat SSL de développement).
10. Suivez les instructions pour créer une demande de signature de certificat (CSR, Certificate Signing Request), chargez la demande et téléchargez un certificat SSL qui sera utilisé pour communiquer avec Apple Notification Service (APNS).
11. Revenez à la page Certificates, Identifiers & Profiles (Certificats, Identifiants et Profils). Cliquez sur All (Tous) sous Provisioning Profiles (Profils de mise en service).
12. Cliquez sur le bouton Plus dans l'angle supérieur droit pour ajouter un nouveau profil de mise en service.
13. Sélectionnez iOS App Development (Développement d'application iOS), puis cliquez sur Continue (Continuer).
14. Sélectionnez l'ID de votre application, puis cliquez sur Continue (Continuer).
15. Sélectionnez votre certificat de développeur, puis cliquez sur Continue (Continuer).
16. Sélectionnez votre appareil, puis cliquez sur Continue (Continuer).
17. Entrez un nom de profil, puis cliquez sur Generate (Générer).
18. Téléchargez le fichier de mise en service et double-cliquez dessus pour installer le profil de mise en service.

Pour en savoir plus sur la mise en service d'un profil configuré pour les notifications push, reportez-vous à la documentation Configuring Push Notifications publiée par Apple.

Utiliser le certificat pour créer un ARN de plate-forme dans la console SNS

1. Exécutez l'application KeyChain d'accès, sélectionnez Mes certificats dans le coin inférieur gauche de l'écran, puis cliquez avec le bouton droit sur le certificat SSL que vous avez généré pour vous connecter à APNS et sélectionnez Exporter. Vous serez invité à indiquer un nom pour le fichier et un mot de passe pour protéger le certificat. Le certificat sera enregistré dans un fichier P12.
2. Accédez à la console SNS et cliquez sur Applications sur la gauche de l'écran.

3. Cliquez sur Créer une application de plate-forme pour créer une nouvelle application de plateforme SNS.
4. Saisissez le nom de l'application.
5. Sélectionnez Apple Development comme plate-forme de notifications Push.
6. Cliquez sur Choisir un fichier et sélectionnez le fichier P12 que vous avez créé lorsque vous avez exporté votre certificat SSL.
7. Entrez le mot de passe vous avez spécifié lorsque vous avez exporté le certificat SSL et cliquez sur Charger à partir du fichier.
8. Cliquez sur Créer une application de plate-forme.
9. Sélectionnez l'application de plateforme que vous venez de créer et copiez l'ARN de l'application. Vous aurez besoin de ces informations pour la suite de la procédure.

Ajoutez un NuGet package pour SNS à votre projet

Suivez l'étape 4 des instructions de la section Configuration du SDK AWS Mobile pour .NET and Xamarin pour ajouter le package NuGet Amazon Simple Notification Service à votre projet.

Créer un client SNS

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

Enregistrer votre application pour activer les notifications à distance

Pour enregistrer une application, faites appel RegisterForRemoteNotifications à votre UIApplication objet, comme indiqué ci-dessous. Placez le code suivant dans le AppDelegate fichier .cs, en insérant l'ARN de votre application de plateforme comme demandé ci-dessous :

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

Envoyer un message à partir de la console SNS vers votre point de terminaison

1. Accédez à la Console SNS > Applications.
2. Sélectionnez votre application de plateforme, choisissez un point de terminaison et cliquez sur Publier sur le point de terminaison.
3. Pour publier un message, saisissez un message texte dans la zone de texte et cliquez sur Publier le message.

Recevoir des notifications Push via SNS (Xamarin Android)

Ce didacticiel explique comment envoyer des notifications push à une application Xamarin pour Android à l'aide d'Amazon Simple Notification Service (SNS) et du SDK mobile AWS pour .NET and Xamarin.

Configuration du projet

Prérequis

Avant de commencer ce didacticiel, vous devez suivre toutes les instructions de la section Configurer un kit SDK AWS Mobile pour .NET et Xamarin.

Définir les autorisations pour SNS

Suivez l'étape 2 de la section Configurer le kit SDK AWS Mobile pour .NET et Xamarin afin d'attacher la stratégie susmentionnée aux rôles de votre application. Votre application bénéficiera ainsi des autorisations appropriées pour accéder à SNS :

1. Accédez à la console IAM et sélectionnez le rôle IAM que vous souhaitez configurer.
2. Cliquez sur Attacher la politique, sélectionnez la politique Amazon SNSFull Access et cliquez sur Attacher la politique.

> **Warning**
>
> L'utilisation SNSFull d'Amazon Access n'est pas recommandée dans un environnement de production. Nous l'utilisons ici pour vous permettre d'être opérationnel rapidement. Pour en savoir plus sur la définition d'autorisations pour un rôle IAM, consultez Présentation des autorisations des rôles IAM.

Activer les notifications Push sur Google Cloud

Commencez par ajouter un projet d'API Google :

1. Accédez à la console Google de développement.
2. Cliquez sur Create Project (Créer un projet) .
3. Dans la zone New Project (Nouveau projet), entrez un nom de projet, notez son ID (vous en aurez besoin plus tard) et cliquez sur Create (Créer).

Ensuite, activez le service de messagerie de Google Cloud (GCM) pour le projet :

1. Dans la console des développeurs Google, votre nouveau projet est normalement déjà sélectionné. Dans le cas contraire, sélectionnez-le dans le menu déroulant en haut de la page.
2. Sélectionnez APIs & auth dans la barre latérale sur le côté gauche de la page.
3. Dans la zone de recherche, tapez « Google Cloud Messaging pour Android » et cliquez sur le lien du même nom.
4. Cliquez sur Enable API.

Enfin, obtenez une clé API :

1. Dans la console Google Developers, sélectionnez APIs & auth > Identifiants.
2. Sous Public API access, cliquez sur Create new key.
3. Dans la boîte de dialogue Create new key, cliquez sur Server Key.
4. Dans la boîte de dialogue qui s'affiche, cliquez sur Create, puis copiez la clé d'API affichée. Vous utiliserez cette clé API pour effectuer l'authentification par la suite.

Utilisez un identifiant de projet pour créer un ARN de plateforme dans la console SNS

1. Accédez à la console SNS.
2. Cliquez sur Applications sur le côté gauche de l'écran.
3. Cliquez sur Créer une application de plate-forme pour créer une nouvelle application de plateforme SNS.
4. Saisissez le nom de l'application.
5. Sélectionnez Google Cloud Messaging (GCM) comme plate-forme de notifications Push.
6. Collez la clé de l'API dans la zone de texte API key.
7. Cliquez sur Créer une application de plate-forme.
8. Sélectionnez l'application de plateforme que vous venez de créer et copiez l'ARN de l'application.

Ajoutez un NuGet package pour SNS à votre projet

Suivez l'étape 4 des instructions de la section Configuration du SDK AWS Mobile pour .NET and Xamarin pour ajouter le package NuGet Amazon Simple Notification Service à votre projet.

Créer un client SNS

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

Enregistrer votre application pour activer les notifications à distance

Pour vous inscrire aux notifications à distance sur Android, vous devez créer un BroadcastReceiver système capable de recevoir des messages Google Cloud. Modifiez le nom de package ci-dessous lorsque vous êtes invité à le faire :

```
[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);
      }
}
```

Vous trouverez ci-dessous le service qui reçoit la notification push de la part du BroadcastReceiver et qui affiche la notification dans la barre de notification de l'appareil :

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

Envoyer un message à partir de la console SNS vers votre point de terminaison

1. Accédez à la Console SNS > Applications.

2. Sélectionnez votre application de plateforme, choisissez un point de terminaison et cliquez sur Publier sur le point de terminaison.
3. Pour publier un message, saisissez un message texte dans la zone de texte et cliquez sur Publier le message.

Amazon Cognito Identity

Qu'est-ce qu'Amazon Cognito Identity ?

Amazon Cognito Identity vous permet de créer des identités uniques pour des utilisateurs et de les authentifier avec les fournisseurs d'identités. Une identité vous permet d'obtenir des informations d'identification AWS temporaires avec des privilèges limités pour synchroniser les données avec Amazon Cognito Sync ou accéder directement aux autres services AWS. Amazon Cognito Identity prend en charge les fournisseurs d'identité publics, tels qu'Amazon, Facebook et Google, ainsi que les identités non authentifiées. Il prend également en charge les identités authentifiées par les développeurs, qui vous permettent d'inscrire et d'authentifier les utilisateurs par l'intermédiaire de votre processus d'authentification backend.

Pour en savoir plus sur Cognito Identity, consultez le Manuel du développeur Amazon Cognito.

Pour en savoir plus sur la disponibilité des régions pour l'authentification Cognito, consultez Disponibilité des régions pour les services AWS.

Utilisation d'un fournisseur public pour authentifier les utilisateurs

A l'aide d'Amazon Cognito Identity, vous pouvez créer des identités uniques pour vos utilisateurs et les authentifier pour qu'ils bénéficient d'un accès sécurisé à vos ressources AWS, comme Amazon S3 ou Amazon DynamoDB. Amazon Cognito Identity prend en charge les fournisseurs d'identité publics (Amazon, Facebook, Twitter/Digits, Google ou tout autre fournisseur compatible avec OpenID Connect) ainsi que les identités non authentifiées.

Pour plus d'informations sur l'utilisation de fournisseurs d'identité publics comme Amazon, Facebook, Twitter/Digits ou Google pour l'authentification des utilisateurs, consultez la page Fournisseurs d'identité externes dans le manuel du développeur Amazon Cognito.

Utilisation d'identités authentifiées par le développeur

Amazon Cognito prend en charge les identités authentifiées par le développeur, en plus de la fédération d'identité Web via Facebook, Google et Amazon. Les identités authentifiées par le développeur vous permettent d'inscrire et d'authentifier les utilisateurs via votre processus d'authentification existant, tout en utilisant Amazon Cognito Sync pour synchroniser les données les concernant et l'accès aux ressources AWS. Leur utilisation implique l'interaction entre le dispositif de l'utilisateur, votre système backend pour l'authentification et Amazon Cognito.

Pour plus d'informations sur les identités authentifiées par les développeurs, consultez Identités authentifiées par le développeur dans le manuel du développeur Amazon Cognito.

Amazon Cognito Sync

Qu'est-ce qu'Amazon Cognito Sync ?

Amazon Cognito Sync est un service AWS et une bibliothèque client qui permet une synchronisation des données utilisateur (p. ex. scores de jeu, préférences utilisateur, état du jeu) sur différents appareils. Vous pouvez utiliser l'API Cognito Sync pour synchroniser les données utilisateur sur tous les appareils. Pour utiliser Cognito Sync dans votre application, vous devez inclure les | dans votre projet.

Pour savoir comment intégrer Amazon Cognito Sync dans votre application, consultez le manuel du développeur Amazon Cognito Sync.

Amazon Mobile Analytics

Amazon Mobile Analytics est un service qui vous permet de recueillir, de visualiser, de comprendre et d'extraire les données d'utilisation de votre application à grande échelle. Mobile Analytics capture facilement les données des appareils standard et les événements personnalisés, et génère automatiquement des rapports à votre place. Outre les rapports consolidés mentionnés ci-dessous, vous pouvez également configurer vos données de manière à les exporter automatiquement dans Redshift et S3 pour des analyses complémentaires.

Avec Amazon Mobile Analytics, vous pouvez suivre les comportements des clients, regrouper des mesures, générer des visualisations de données et identifier des tendances pertinentes.

Concepts clés

Types de rapport

Par défaut, Mobile Analytics fournit les rapports suivants dans la console Mobile Analytics :

- Daily Active Users (DAU, utilisateurs actifs quotidiens), Monthly Active Users (MAU, utilisateurs actifs mensuels) et New Users (nouveaux utilisateurs)
- Sticky factor (taux de fidélisation) (DAU divisé par MAU)
- Session Count (nombre de sessions) et Average Sessions per Daily Active User (nombre moyen de sessions par utilisateur actif quotidien)
- Average Revenue per Daily Active User (ARPDAU, revenus moyens par utilisateur actif quotidien) et Average Revenue per Paying Daily Active User (ARPPDAU, revenus moyens par utilisateur actif quotidien payant)
- Day 1, 3, and 7 Retention (rétention à 1, 3 et 7 jours) et Week 1, 2, and 3 Retention (rétention à 1, 2 et 3 semaines)
- Événements personnalisés

Ces rapports sont fournis par l'intermédiaire de six onglets dans la console :

- Vue d'ensemble — Suivez neuf rapports présélectionnés dans un simple-to-review tableau de bord pour avoir une idée rapide de l'engagement : MAU, DAU, nouveaux utilisateurs, sessions quotidiennes, Sticky Factor, rétention en un jour, ARPDAU, utilisateurs payants quotidiens, ARPPDAU.

- Active Users (Utilisateurs actifs) : permet de connaître, par jour et par mois, le nombre d'utilisateurs se servant de votre application et de surveiller le taux de fidélisation pour mesurer l'engagement, l'intérêt et la monétisation.
- Sessions : permet de suivre la fréquence d'utilisation de votre application sur une journée donnée et de savoir à quelle fréquence chaque utilisateur ouvre votre application au cours d'une journée.
- Retention (Conservation) : permet de suivre la fréquence à laquelle les clients réutilisent votre application sur une base quotidienne et hebdomadaire.
- Revenue (Revenus) : permet de suivre les tendances pour les revenus intégrés à l'application, afin d'identifier les domaines d'amélioration de la monétisation.
- Custom events (Evénements personnalisés) : permet de suivre les actions utilisateur personnalisées propres à votre application.

Pour en savoir plus sur les rapports Mobile Analytics et sur l'utilisation de la console correspondante, consultez la page Vue d'ensemble des rapports de la console Mobile Analytics dans le Manuel du développeur Mobile Analytics.

Configuration du projet

Prérequis

Pour utiliser Mobile Analytics dans votre application, vous devez ajouter le kit SDK à votre projet. Pour ce faire, suivez les instructions de la section Configurer le kit SDK AWS Mobile pour .NET et Xamarin.

Configurer les paramètres Mobile Analytics

Mobile Analytics définit certains paramètres qui peuvent être configurés dans le fichier awsconfig.xml :

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

- SessionTimeout- Si l'application reste en arrière-plan pendant une durée supérieure à celle du client Mobile Analytics SessionTimeout , celui-ci met fin à la session en cours et une nouvelle session est créée lorsque l'application revient au premier plan. Nous vous conseillons d'utiliser des valeurs comprises entre 5 et 10. La valeur par défaut est 5.
- Max DBSize : taille maximale de la base de données (en octets) utilisée pour le stockage local des événements. Si la taille de la base de données dépasse cette valeur, les événements suivants sont ignorés. Nous vous conseillons d'utiliser des valeurs comprises entre 1 Mo et 10 Mo. La valeur par défaut est 5242880 (5 Mo).
- DBWarningSeuil : seuil d'avertissement. Les valeurs valides sont comprises entre 0 et 1. Si les valeurs dépassent ce seuil, des journaux d'avertissement sont générés. La valeur par défaut est 0.9.
- MaxRequestSize- La taille maximale d'une requête HTTP envoyée au service Mobile Analytics. La valeur est spécifiée en octets et peut être comprise entre 1 et 512 Ko. La valeur par défaut est 102400 (100 Ko). N'utilisez pas de valeurs supérieures à 512 Ko, faute de quoi le service risque de refuser la demande HTTP.
- AllowUseDataNetwork- Une valeur indiquant si l'appel de service est autorisé sur un réseau de données cellulaires. Utilisez cette option avec précaution, car cela peut accroître l'utilisation des données du client.

Les paramètres ci-dessus constituent les valeurs par défaut pour chaque élément de configuration.

Intégration de Mobile Analytics avec votre application

Les sections ci-dessous expliquent comment intégrer Mobile Analytics à votre application.

Créer une application dans la console Mobile Analytics

Accédez à la console Amazon Mobile Analytics et créez une application. Notez la valeur `appId` car vous en aurez besoin par la suite. Lorsque vous créez une application dans la console Mobile Analytics, vous devez spécifier l'ID de votre groupe d'identités. Pour savoir comment créer un pool d'identités, consultez la section Configurer le kit SDK AWS Mobile pour .NET et Xamarin.

Pour en savoir plus sur l'utilisation de la console Mobile Analytics, consultez la page Vue d'ensemble des rapports de la console Mobile Analytics dans le Manuel du développeur Mobile Analytics.

Création d'un MobileAnalyticsManager client

Pour initialiser votre MobileAnalyticsManager, faites appel GetOrCreateInstance à votre`MobileAnalyticsManager`, en transmettant vos informations d'identification AWS, votre région, l'ID de votre application Mobile Analytics et votre objet de configuration facultatif :

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

L'`APP_ID` est généré pour vous par l'Assistant de création d'applications. Ces deux valeurs doivent correspondre à celles de la console Mobile Analytics. L'`APP_ID` permet de grouper vos données dans la console Mobile Analytics. Après avoir créé une application dans la console Mobile Analytics, vous pouvez trouver son ID en accédant à la console Mobile Analytics, puis en cliquant sur l'icône en forme de roue dentée dans le coin supérieur droit de l'écran. Cela affichera la page de gestion des applications qui répertorie toutes les applications enregistrées et leur application IDs.

Enregistrer les événements de monétisation

Le SDK AWS Mobile pour .NET et `MonetizationEvent` Xamarin fournit cette classe, qui vous permet de générer des événements de monétisation afin de suivre les achats effectués dans les applications mobiles. L'extrait de code suivant montre comment créer un événement de monétisation :

```
// 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);
```

Enregistrer les événements personnalisés

Mobile Analytics vous permet de définir des événements personnalisés. Vous définissez entièrement les événements personnalisés. Ils vous aident à suivre les actions utilisateur propres à votre application ou à votre jeu. Pour plus d'informations sur les événements personnalisés, consultez la page Evénements personnalisés.

Pour cet exemple, nous supposons que notre application est un jeu, et que nous voulons enregistrer un événement lorsqu'un utilisateur termine un niveau. Créez un événement « LevelComplete » en créant une nouvelle `AmazonMobileAnalyticsEvent` instance :

```
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);
```

Sessions d'enregistrement

Xamarin iOS

Lorsque l'application passe à l'arrière-plan, vous pouvez mettre la session en pause. Pour les applications iOS, dans le fichier AppDelegate .cs, remplacez `DidEnterBackground` `MobileAnalyticsManager.PauseSesion` et `WillEnterForeground` appelez, `MobileAnalyticsManager.ResumeSession` comme indiqué dans l'extrait de code suivant :

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

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

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

Xamarin Android

Pour les applications Android, appelez `MobileAnalyticsManager.PauseSesion` la méthode OnPause () et `MobileAnalyticsManager.ResumeSession` la méthode OnResume () comme indiqué dans l'extrait de code suivant :

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

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

Par défaut, si l'utilisateur sort de l'application pendant moins de 5 secondes, puis y revient, sa session reprend. Si l'utilisateur quitte l'application pendant 5 secondes ou plus, une nouvelle session est créée. Ce paramètre est configurable dans le fichier de configuration aws_mobile_analytics.json en définissant la propriété « SESSION_DELTA » sur le délai (en secondes) devant précéder la création d'une nouvelle session.

Amazon Simple Storage Service (S3)

Qu'est-ce qu'S3 ?

Amazon Simple Storage Service (Amazon S3) offre aux développeurs un espace de stockage d'objets sécurisé, durable et hautement évolutif. Amazon S3 est facile à utiliser et doté d'une interface de services Web simple pour stocker et récupérer toute quantité de données depuis n'importe où sur le Web. Avec Amazon S3, vous ne payez que le stockage que vous utilisez réellement. Il n'y a aucun frais minimum ni frais d'installation.

Amazon S3 fournit un espace de stockage d'objets économique pour de nombreux et divers cas d'utilisation, notamment les applications cloud, la diffusion de contenu, la sauvegarde et l'archivage, la reprise après sinistre et les analyses Big Data.

Pour en savoir plus sur la disponibilité des régions pour AWS S3, consultez la page Disponibilité des régions pour les services AWS.

Concepts clés

Compartiment

Chaque objet que vous stockez dans Amazon S3 réside dans un compartiment. Vous pouvez utiliser des compartiments pour regrouper des objets connexes, comme lorsque vous utilisez un répertoire pour regrouper des fichiers dans un système de fichiers. Les compartiments ont des propriétés, telles que les autorisations d'accès et l'état de la gestion des versions, et vous pouvez spécifier où vous souhaitez qu'ils résident.

Pour en savoir plus sur les compartiments S3, consultez la page Utilisation des compartiments dans le Manuel du développeur S3.

Objets

Les objets correspondent aux données que vous stockez dans Amazon S3. Chaque objet réside dans un compartiment que vous créez dans une région AWS spécifique.

Les objets stockés dans une région ne quittent jamais la région, à moins que vous ne les transfériez explicitement vers une autre région. Par exemple, les objets stockés dans la région UE (Irlande)

ne la quittent jamais. Les objets stockés dans une région Amazon S3 restent physiquement dans cette région. Amazon S3 ne conserve pas de copies et ne les déplace pas dans une autre région. Toutefois, vous pouvez accéder aux objets depuis n'importe où, tant que vous disposez des autorisations nécessaires.

Les objets peuvent correspondre à tout type de fichier : images, données de sauvegarde, films, etc. Un objet peut être aussi volumineux que 5 To. Vous pouvez disposer d'un nombre illimité d'objets dans un compartiment.

Avant de pouvoir charger un objet dans Amazon S3, vous devez disposer d'autorisations en écriture pour un compartiment. Pour plus d'informations sur la définition des autorisations de compartiment, consultez la page Gérer les autorisations d'un compartiment dans le Manuel du développeur S3.

Pour en savoir plus sur les compartiments S3, consultez la page Utilisation des objets dans le Manuel du développeur S3.

Métadonnées d'objet

Chaque objet dans Amazon S3 possède un ensemble de paires clé-valeur qui représente ses métadonnées. Il existe deux types de métadonnées :

- Métadonnées système : parfois traitées par Amazon S3, par exemple, Content-Type et Content-Length.
- Métadonnées utilisateur : jamais traitées par Amazon S3. Les métadonnées utilisateur sont stockées avec l'objet et renvoyées avec lui. La taille maximum des métadonnées utilisateur est de 2 Ko, et leurs clés et valeurs doivent respecter les normes US-ASCII.

Pour en savoir plus sur les métadonnées d'un objet S3, consultez la page Modification des métadonnées d'un objet.

Configuration du projet

Prérequis

Pour utiliser Amazon S3 dans votre application, vous devez ajouter le kit SDK à votre projet. Pour ce faire, suivez les instructions de la section Configurer le kit SDK AWS Mobile pour .NET et Xamarin.

Création d'un compartiment S3

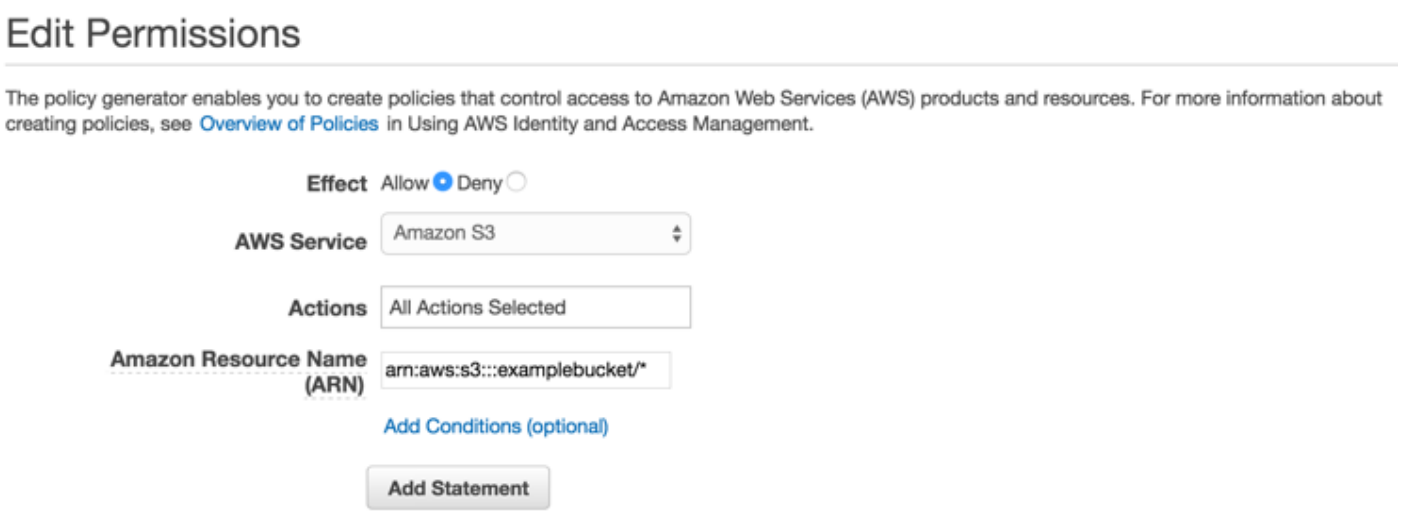
Amazon S3 stocke les ressources de votre application dans des compartiments Amazon S3. Ces conteneurs de stockage cloud résident dans une région spécifique. Chaque compartiment Amazon S3 doit avoir un nom unique dans le monde entier. Vous pouvez utiliser la console Amazon S3 pour créer un compartiment.

1. Connectez-vous à la console Amazon S3, puis cliquez sur Créer un compartiment.
2. Saisissez le nom du compartiment, sélectionnez une région, puis cliquez sur Créer.

Définir les autorisations pour S3

La stratégie de rôle IAM par défaut autorise votre application à accéder à Amazon Mobile Analytics et Amazon Cognito Sync. Pour que votre pool d'identités Cognito puisse accéder à Amazon S3, vous devez modifier les rôles de ce pool d'identités.

1. Accédez à la console Identity and Access Management et cliquez sur Rôles dans le volet de gauche.
2. Entrez le nom du pool d'identités dans la zone de recherche. Deux rôles seront répertoriés : un premier pour les utilisateurs authentifiés et un second pour les utilisateurs non authentifiés.
3. Cliquez sur le rôle pour les utilisateurs non authentifiés (unauth sera ajouté au nom du pool d'identités).
4. Cliquez sur Créer une stratégie de rôle, sélectionnez Générateur de stratégies, puis cliquez sur Sélectionner.
5. Sur la page Modifier les autorisations, entrez les paramètres affichés dans l'image suivante, en remplaçant le nom de ressource Amazon (ARN) par le vôtre. L'ARN d'un compartiment S3 ressemble à ceci : `arn:aws:s3:::examplebucket/*`. Il se compose de la région dans laquelle se trouve le compartiment et du nom du compartiment. Les paramètres ci-dessous accordent au pool d'identités complet l'accès à toutes les actions pour le compartiment spécifié.



1. Cliquez sur le bouton Ajouter une instruction, puis sur Étape suivante.
2. L'Assistant vous indiquera la configuration qui aura été générée. Cliquez sur Apply Policy.

Pour plus d'informations sur l'octroi d'accès à S3, consultez Octroi d'accès à un compartiment Amazon S3.

(facultatif) Configurer la version de Signature pour les requêtes S3

Chaque interaction avec Amazon S3 est authentifiée ou anonyme. AWS utilise les algorithmes Signature Version 4 ou Signature Version 2 pour authentifier les appels au service.

Toutes les nouvelles régions AWS créées après janvier 2014 prennent en charge Signature Version 4 uniquement. Cependant, de nombreuses régions plus anciennes prennent encore en charge les demandes de Signature Version 4 et Signature Version 2.

Si votre compartiment se trouve dans l'une des régions qui ne prennent pas en charge les demandes Signature version 2 répertoriées sur cette page, vous devez configurer le AWSConfigs S3. UseSignatureVersion4 propriétés à « vrai » comme ça :

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

Pour plus d'informations sur les versions d'AWS Signature, consultez la page Demandes d'authentification (AWS Signature Version 4).

Intégration de S3 à votre application

Il existe deux façons d'interagir avec S3 dans votre application Xamarin. Les deux méthodes sont étudiées en détail dans les rubriques suivantes :

Utilisation de l'utilitaire de transfert de S3

L'utilitaire de transfert S3 facilite le téléchargement et le chargement des fichiers vers S3 à partir de votre application Xamarin.

Initialisez le TransferUtility

Créez un client S3, passez-lui l'objet des informations d'identification AWS, puis passez le client S3 à l'utilitaire de transfert, comme suit :

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

(facultatif) Configurez le TransferUtility

Vous pouvez configurer trois propriétés facultatives :

- ConcurrentServiceRequests- Détermine le nombre de threads actifs ou le nombre de requêtes Web asynchrones simultanées qui seront utilisés pour télécharger/télécharger le fichier. La valeur par défaut est 10.
- MinSizeBeforePartUpload- Obtient ou définit la taille minimale des parties à télécharger en octets. La valeur par défaut est 16 Mo. La réduction de la taille partielle minimum entraîne des chargements en plusieurs parties qui seront fractionnés en un nombre plus grand de petites parties. Une valeur trop faible a des répercussions négatives sur les vitesses de transfert, ce qui entraîne une latence et une communication réseau supplémentaires pour chaque partie.
- NumberOfUploadThreads- Obtient ou définit le nombre de threads en cours d'exécution. Cette propriété détermine le nombre de threads actifs qui seront utilisés pour charger le fichier. La valeur par défaut est de 10 threads.

Pour configurer le TransferUtility client S3, créez un objet de configuration, définissez vos propriétés et transmettez l'objet à votre TransferUtility constructeur comme suit :

```
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);
```

Télécharger un fichier

Pour télécharger un fichier depuis S3, appelez `Download` sur l'objet Transfer Utility (Utilitaire de transfert) avec les paramètres suivants :

- `file` : nom de chaîne du fichier que vous souhaitez télécharger
- `bucketName` : nom de chaîne du compartiment S3 à partir duquel vous voulez télécharger le fichier
- `key` : chaîne représentant le nom de l'objet S3 (un fichier dans le cas présent) à télécharger

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

Charger un fichier

Pour charger un fichier dans S3, appelez `Upload` sur l'objet Transfer Utility (Utilitaire de transfert) avec les paramètres suivants :

- `file` : nom de chaîne du fichier que vous souhaitez charger
- `bucketName` : nom de chaîne du compartiment S3 destiné au stockage du fichier

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

Le code ci-dessus suppose qu'il existe un fichier dans le répertoire Environment. SpecialFolder. ApplicationData. Les chargements utilisent automatiquement la fonctionnalité de téléchargement en plusieurs parties de S3 sur les fichiers volumineux pour améliorer le débit.

Utilisation du niveau de service S3 APIs

En plus d'utiliser le S3 TransferUtility, vous pouvez également interagir avec le S3 à l'aide du S3 de bas niveau. APIs

Initialiser le client Amazon S3

Pour utiliser Amazon S3, nous devons d'abord créer une instance AmazonS3Client qui fait référence à l'instance Cognito AWSCredentials que vous avez créée précédemment et à votre région :

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

Télécharger un fichier

Pour télécharger un fichier de S3 :

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

Charger un fichier

Pour charger un fichier dans 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);
```

Supprimer un élément

Pour supprimer un élément de S3 :

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

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

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

Supprimer plusieurs éléments

Pour supprimer plusieurs objets d'un compartiment à l'aide d'une requête HTTP unique :

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

Vous pouvez spécifier jusqu'à 1 000 clés.

Etablir une liste des compartiments

Pour renvoyer une liste de tous les compartiments détenus par l'expéditeur authentifié de la demande :

```
// 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);
}
```

Affichage de la liste des objets

Vous pouvez retourner tout ou une partie (jusqu'à 1 000) des objets stockés dans votre compartiment S3. Pour cela, vous devez disposer d'un accès en lecture au compartiment.

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

Obtenir la région d'un compartiment

Pour obtenir la région dans laquelle réside un compartiment :

```
// 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);
```

Obtenir une stratégie de compartiment

Pour obtenir une stratégie de compartiment :

```
// 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

Qu'est-ce qu'Amazon DynamoDB ?

Amazon DynamoDB est un service de bases de données non relationnelles rapide et hautement évolutif. DynamoDB permet de s'affranchir des limites habituelles du dimensionnement de stockage de données, tout en conservant une faible latence et des performances prévisibles.

Concepts clés

Les concepts de modèle de données DynamoDB incluent des tables, des éléments et des attributs.

Tables

Dans Amazon DynamoDB, une base de données est un ensemble de tables. Une table est un ensemble d'éléments et chaque élément est un ensemble d'attributs.

Dans une base de données relationnelle, une table a un schéma prédéfini, tel que le nom de la table, la clé primaire, la liste des noms de colonnes et leurs types de données. Tous les enregistrements stockés dans la table doivent avoir le même ensemble de colonnes. En revanche, DynamoDB exige uniquement qu'une table ait une clé primaire, mais ne requiert pas que vous définissiez tous les noms d'attribut et les types de données à l'avance.

Pour en savoir plus sur l'utilisation des tables, consultez Utilisation de tables dans DynamoDB.

Eléments et attributs

Des éléments individuels dans une table DynamoDB peuvent avoir n'importe quel nombre d'attributs, bien qu'il y ait une limite de 400 Ko sur la taille d'élément. La taille d'un élément est la somme des longueurs de ses noms et valeurs d'attribut (longueurs binaire et UTF-8).

Chaque attribut dans un élément est une paire nom-valeur. Un attribut peut être un ensemble à valeur unique ou à plusieurs valeurs. Par exemple, un élément de livre peut avoir des attributs de titre et d'auteurs. Chaque livre a un titre mais peut avoir plusieurs auteurs. L'attribut à plusieurs valeurs est un ensemble ; les valeurs en double ne sont pas autorisées.

Par exemple, envisagez de stocker un catalogue de produits dans DynamoDB. Vous pouvez créer une table avec l'attribut Id comme clé primaire. ProductCatalog La clé primaire identifie de manière

unique chaque élément de la table, de sorte que deux produits de la table ne peuvent pas avoir le même ID.

Pour en savoir plus sur l'utilisation des éléments, consultez Utilisation des éléments dans DynamoDB.

Les types de données

Amazon DynamoDB prend en charge les types de données suivants :

- Types scalar : Number, String, Binary, Boolean et Null.
- Types à valeurs multiples : String Set, Number Set et Binary Set.
- Types document : List et Map.

Pour plus d'informations sur les types de données scalar, à valeurs multiples et document, consultez la page Types de données DynamoDB.

Clé primaire

Lorsque vous créez une table, en plus du nom de la table, vous devez spécifier la clé primaire de la table. La clé primaire identifie de manière unique chaque élément de la table, afin qu'aucun deux éléments n'ait la même clé. DynamoDB prend en charge les deux types de clés primaires suivants :

- Clé de hachage : la clé primaire est constituée d'un seul attribut, un attribut de hachage. DynamoDB crée un index de hachage non ordonné sur cet attribut de clé primaire. Chaque élément de la table est identifié de manière unique par sa valeur clé de hachage.
- Clé de hachage et de plage : la clé primaire est constituée de deux attributs. Le premier est l'attribut de hachage et le second est l'attribut de plage. DynamoDB crée un index de hachage non ordonné sur l'attribut de clé primaire de hachage et un index de plage trié sur l'attribut de clé primaire de plage. Chaque élément de la table est identifié de manière unique par la combinaison de ses valeurs de clé de hachage et de plage. Il est possible pour deux éléments d'avoir la même valeur de clé de hachage, mais ces deux éléments doivent avoir des valeurs de clés de plage différentes.

Index secondaires

Lorsque vous créez une table avec une clé de hachage et de plage, vous pouvez définir, le cas échéant, un ou plusieurs index secondaires sur cette table. Un index secondaire vous permet

d'interroger les données de la table à l'aide d'une clé alternative, en plus des requêtes sur la clé primaire.

DynamoDB prend en charge deux types d'index secondaires : les index secondaires locaux et index secondaires globaux.

- Index secondaire local : index ayant la même clé de hachage que la table, mais une clé de plage différente.
- Index secondaire global : index avec une clé de hachage et une clé de plage qui peuvent être différentes de celles de la table.

Vous pouvez définir jusqu'à 5 index secondaires globaux et 5 index secondaires locaux par table. Pour plus d'informations sur ces index, consultez Amélioration de l'accès aux données avec les index secondaires dans DynamoDB dans le Manuel du développeur DynamoDB.

Query and Scan

Outre l'utilisation de clés primaires pour accéder aux éléments, Amazon DynamoDB en fournit APIs deux pour rechercher les données : Query et Scan. Nous vous recommandons de lire Bonnes pratiques concernant l'interrogation et l'analyse des données dans le Manuel du développeur DynamoDB pour vous familiariser avec quelques bonnes pratiques.

Requête

Une opération Query recherche les éléments d'une table ou d'un index secondaire uniquement à l'aide des valeurs d'attribut de la clé primaire. Vous devez fournir un attribut de clé de hachage et une valeur distincte à rechercher. Vous pouvez, le cas échéant, fournir un attribut de clé de plage et une valeur, et utiliser un opérateur de comparaison pour affiner les résultats de recherche.

Pour des exemples de requêtes, consultez :

- Utilisation du modèle de document
- Utilisation du modèle de persistance des objets
- Utilisation du niveau de service DynamoDB APIs

Pour plus d'informations sur l'interrogation des données, consultez la partie Interrogation dans le Manuel du développeur DynamoDB.

Analyser

Une opération Scan lit tous les éléments d'une table ou d'un index secondaire. Par défaut, une opération Scan retourne tous les attributs de données pour chaque élément de la table ou de l'index. Vous pouvez utiliser le ProjectionExpression paramètre pour que Scan ne renvoie que certains attributs, plutôt que tous.

Pour des exemples d'analyses, consultez :

- Utilisation du modèle de document
- Utilisation du modèle de persistance des objets
- Utilisation du niveau de service DynamoDB APIs

Pour plus d'informations sur l'analyse des données, consultez la partie Analyse dans le Manuel du développeur DynamoDB.

Configuration du projet

Prérequis

Pour utiliser DynamoDB dans votre application, vous devez ajouter le kit SDK à votre projet. Pour ce faire, suivez les instructions de la section Configurer le kit SDK AWS Mobile pour .NET et Xamarin.

Créer une table DynamoDB

Pour créer une table, accédez à la console DynamoDB et procédez comme suit :

1. Cliquez sur Créer une table.
2. Saisissez le nom de la table.
3. Sélectionnez Hachage comme type de clé primaire.
4. Sélectionnez un type et entrez une valeur pour le nom d'attribut de hachage. Cliquez sur Continuer.
5. Sur la page Add Indexes (Ajouter des index), si vous prévoyez d'utiliser des index secondaires globaux, spécifiez Global Secondary Index pour Index Type (Type d'index) et sous Index Hash Key (Clé de hâchage d'index), saisissez une valeur pour l'index secondaire. Cela vous permettra

d'effectuer une interrogation et une analyse à l'aide de l'index principal et de l'index secondaire. Cliquez sur Add Index To Table (Ajouter l'index à la table), puis sur Continue (Continuer). Pour ne pas utiliser d'index secondaires globaux, cliquez sur Continue (Continuer).

6. Définissez la capacité de lecture et d'écriture aux niveaux souhaités. Pour plus d'informations sur la configuration de la capacité, consultez Débit alloué dans Amazon DynamoDB. Cliquez sur Continuer.
7. Dans l'écran suivant, entrez un e-mail de notification afin de créer des alarmes relatives au débit, si vous le souhaitez. Cliquez sur Continuer.
8. Sur la page récapitulative, cliquez sur Create (Créer). DynamoDB crée alors votre base de données.

Définir les autorisations pour DynamoDB

Pour utiliser DynamoDB dans une application, vous devez définir les autorisations appropriées. La stratégie IAM suivante permet à l'utilisateur de supprimer, d'obtenir, de placer, d'interroger, d'analyser et de mettre à jour des éléments dans une table DynamoDB qui est identifiée par son ARN :

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

Vous pouvez modifier les stratégies dans la console IAM. Ajoutez ou supprimez ensuite des actions autorisées selon les besoins de votre application.

Pour plus d'informations sur les stratégies IAM, consultez Utilisation d'IAM.

Pour en savoir plus sur les stratégies spécifiques à DynamoDB, consultez Utilisation d'IAM pour contrôler l'accès aux ressources DynamoDB dans le Manuel du développeur DynamoDB.

Intégration de DynamoDB avec votre application

Le kit SDK AWS Mobile pour .NET and Xamarin fournit une bibliothèque de haut niveau permettant de travailler avec DynamoDB. Vous pouvez également envoyer des requêtes directement à l'API DynamoDB de bas niveau, mais pour la plupart des cas d'utilisation, il est recommandée d'utiliser la bibliothèque de haut niveau. AmazonDynamoDBClient Il s'agit d'une partie particulièrement utile de la bibliothèque de haut niveau. Cette classe vous permet d'effectuer diverses opérations de création, de lecture, de mise à jour et de suppression, et d'exécuter des requêtes.

Le SDK AWS Mobile pour .NET and Xamarin vous permet de passer des appels en utilisant le SDK AWS pour .NET afin APIs de fonctionner avec DynamoDB. Ils APIs sont tous disponibles dans le AWSSDK fichier .dll. Pour plus d'informations sur le téléchargement du kit SDK AWS pour .NET, consultez Kit AWS SDK pour .NET.

Il existe trois façons d'interagir avec DynamoDB dans votre application Xamarin :

- Modèle de document : cette API fournit des classes wrapper autour de l'API de bas niveau DyanmoDB pour simplifier vos tâches de programmation. Table et Document sont les classes d'enveloppe principales. Vous pouvez utiliser le modèle de document pour les opérations de données telles que créer, extraire, mettre à jour et supprimer des éléments. L'API est disponible dans le fichier Amazon.DynamoDB. DocumentModel espace de noms.
- Modèle de persistance des objets : l'API Object Persistence vous permet de mapper vos classes côté client avec les tables DynamoDB. Ensuite, chaque instance d'objet est mappée à un élément des tables correspondantes. La DBContext classe Dynamo de cette API fournit des méthodes vous permettant d'enregistrer des objets côté client dans une table, de récupérer des éléments sous forme d'objets et d'effectuer des requêtes et des scans. Vous pouvez utiliser le modèle de persistance des objets pour les opérations de données telles que créer, extraire, mettre à jour et supprimer des éléments. Vous devez d'abord créer vos tables à l'aide de l'API Service Client, puis utiliser le modèle de persistance des objets pour mapper vos classes avec les tables. L'API est disponible dans le fichier Amazon.DynamoDB. DataModel espace de noms.
- API Service Client : il s'agit de l'API au niveau du protocole qui assure un mappage étroit avec l'API DynamoDB. Vous pouvez utiliser cette API de bas niveau pour toutes les opérations de table et d'élément, telles que créer, mettre à jour, supprimer la table et les éléments. Vous pouvez également interroger et analyser vos tables. Cette API est disponible dans le namespace Amazon.DynamoDB.

Ces trois modèles sont abordés de manière approfondie dans les rubriques suivantes :

Utilisation du modèle de document

Le modèle de document fournit des classes d'enveloppe autour de l'API .NET de bas niveau. Table et Document sont les classes d'enveloppe principales. Vous pouvez utiliser le modèle de document pour créer, récupérer, mettre à jour et supprimer des éléments. Pour créer, mettre à jour et supprimer des tables, vous devez utiliser l'API de bas niveau. Pour obtenir des instructions sur l'utilisation de l'API de bas niveau, consultez la section Utilisation du niveau de service DynamoDB. APIs L'API de bas niveau est disponible dans Amazon.DynamoDB. DocumentModel espace de noms.

Pour en savoir plus sur le modèle de document, consultez Modèle de document .NET.

Créer un client DynamoDB

Pour créer un client DynamoDB :

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

Opérations CRUD

Enregistrer un élément

Créer un élément :

```
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";
```

Enregistrer un élément dans une table DynamoDB :

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

Récupérer un élément

Pour récupérer un élément :

```
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);
}
```

Mettre à jour un élément

Pour mettre à jour un élément :

```
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);
}
```

Pour mettre à jour un élément sous certaines conditions :

```
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);
}
```

Supprimer un élément

Pour supprimer un élément :

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

Query and Scan

Pour interroger et récupérer tous les livres dont l'auteur est « Mark Twain » :

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

L'exemple de code d'analyse ci-dessous retourne tous les livres de notre table :

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

Utilisation du modèle de persistance des objets

Le SDK AWS Mobile pour .NET and Xamarin fournit un modèle de persistance des objets qui vous permet de mapper vos classes côté client à une table DynamoDB. Ensuite, chaque instance d'objet est mappée à un élément de la table correspondante. Pour enregistrer vos objets côté client dans une table, le modèle de persistance des objets fournit la DBContext classe Dynamo, un point d'entrée vers DynamoDB. Cette classe permet une connexion à DynamoDB et vous donne la possibilité d'accéder aux tables, d'effectuer diverses opérations CRUD et d'exécuter des requêtes.

Le modèle de persistance des objets n'autorise pas une API à créer, mettre à jour ou supprimer des tables. Il fournit uniquement des opérations sur les données. Pour créer, mettre à jour et supprimer des tables, vous devez utiliser l'API de bas niveau. Pour obtenir des instructions sur l'utilisation de l'API de bas niveau, consultez la section Utilisation du niveau de service DynamoDB. APIs

Présentation

Le modèle de persistance des objets fournit un ensemble d'attributs pour mapper les classes côté client aux tables, et les propriétés/champs aux attributs de table. Le modèle de persistance des objets prend en charge le mappage explicite par défaut entre les propriétés de classe et les attributs de table.

- Mappage explicite : pour associer une propriété à une clé primaire, vous devez utiliser les attributs du modèle Dynamo DBHash Key et Dynamo DBRange Key Object Persistence. En outre, pour les attributs de clé non primaires, si le nom d'une propriété de votre classe et l'attribut de table correspondant auquel vous souhaitez la mapper ne sont pas identiques, vous devez définir le mappage en ajoutant explicitement l'attribut DynamoDBProperty.
- Mappage par défaut :par défaut, le modèle de persistance des objets mappe les propriétés de la classe aux attributs avec le même nom dans la table.

Vous n'avez pas à mapper chaque propriété de classe individuelle. Vous identifiez ces propriétés en ajoutant l'DBIgnore attribut Dynamo. La sauvegarde et la récupération d'une instance d'objet ignorent toutes les propriétés marquées avec cet attribut.

Types de données pris en charge

Le modèle de persistance des objets prend en charge un ensemble de types de données .NET primitifs, de collections et de types de données arbitraires. Le modèle prend en charge les types de données primitifs suivants.

- bool
- octet
- char
- DateTime
- decimal, double, float
- Int16, Int32, Int64
- SByte
- chaîne
- UInt16, UInt32, UInt64

Le modèle de persistance des objets prend également en charge les types de collection .NET avec les limitations suivantes :

- Le type de collection doit implémenter ICollection l'interface.
- Le type de collection doit être composé des types primitifs pris en charge. Par exemple ICollection<string>, ICollection<bool>.
- Le type de collection doit fournir un constructeur sans paramètre.

Pour plus d'informations sur le modèle de persistance des objets, consultez Modèle de persistance des objets .NET.

Créer un client DynamoDB

Pour créer un client DynamoDB :

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

Opérations CRUD

Enregistrer un objet

Créer un objet :

```
[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
};
```

Enregistrer un objet dans une table DynamoDB :

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

Récupérer un objet

Pour récupérer un objet :

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

Mettre à jour un objet

Pour mettre à jour un objet :

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

Supprimer un objet

Pour supprimer un objet:

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

Query and Scan

Pour interroger et récupérer tous les livres dont l'auteur est « Charles Dickens » :

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

L'exemple de code d'analyse ci-dessous retourne tous les livres de notre table :

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

Utilisation du niveau de service DynamoDB APIs

Le niveau de service Dynamo vous APIs permet de créer, de mettre à jour et de supprimer des tables. Vous pouvez également effectuer des opérations de création, lecture, mise à jour et suppression (CRUD) classiques sur des éléments de table à l'aide de cette API.

Créer un client DynamoDB

Pour créer un client DynamoDB :

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

Opérations CRUD

Enregistrer un élément

Pour enregistrer un élément dans une table DynamoDB :

```
// 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);
```

Récupérer un élément

Pour récupérer un élément :

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

Mettre à jour un élément

Pour mettre à jour un élément :

```
// 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);
```

Supprimer un élément

Pour supprimer un élément :

```
// 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);
```

Query and Scan

Pour interroger et récupérer tous les livres dont l'auteur est « Mark Twain » :

```
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);
    });

  }
}
```

L'exemple de code d'analyse ci-dessous retourne tous les livres de notre table :

```
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)

À l'aide de SNS et du SDK mobile AWS pour .NET et Xamarin, vous pouvez créer des applications capables de recevoir des notifications push mobiles. Pour en savoir plus sur SNS, consultez la section Amazon Simple Notification Service.

Concepts clés

Amazon SNS permet aux applications et aux utilisateurs finaux de différents appareils de recevoir des notifications par le biais de notifications mobiles push (appareils Apple, Google et Kindle Fire), de HTTP/HTTPS, Email/Email -JSON, de SMS, de files d'attente Amazon Simple Queue Service (SQS) ou de fonctions AWS Lambda. SNS permet d'envoyer des messages individuels ou de diffuser des messages à un grand nombre de destinataires abonnés à une même rubrique.

Rubriques

Une rubrique est un « point d'accès » destiné à permettre aux destinataires de s'abonner de manière dynamique aux copies identiques de la même notification. Une rubrique peut remettre des messages à plusieurs types de points de terminaison. Par exemple, vous pouvez regrouper les destinataires iOS, Android et SMS.

Abonnements

Pour recevoir les messages publiés dans une rubrique, vous devez abonner un point de terminaison à cette rubrique. Un point de terminaison est une application mobile, un serveur Web, une adresse e-mail ou une file d'attente Amazon SQS qui peut recevoir des messages de notification de la part d'Amazon SNS. Une fois que vous avez abonné un point de terminaison à une rubrique et que l'abonnement est confirmé, le point de terminaison reçoit tous les messages publiés dans cette rubrique.

Publication

Lorsque vous publiez un message sur une rubrique, SNS remet une copie du message au format approprié à chaque utilisateur abonné à cette rubrique. Pour les notifications Push mobiles, vous pouvez publier directement sur le point de terminaison ou abonner le point de terminaison à une rubrique.

Configuration du projet

Prérequis

Pour utiliser SNS dans votre application, vous devez ajouter le kit SDK à votre projet. Pour ce faire, suivez les instructions de la section Configurer le kit SDK AWS Mobile pour .NET et Xamarin.

Définir les autorisations pour SNS

Pour en savoir plus sur la définition des autorisations pour SNS, reportez-vous à la section Gestion de l'accès à vos rubriques Amazon SNS.

Ajoutez un NuGet package pour SNS à votre projet

Suivez l'étape 4 des instructions de la section Configuration du SDK AWS Mobile pour .NET and Xamarin pour ajouter le package NuGet Amazon Simple Notification Service à votre projet.

Intégration de SNS à votre application

Il existe diverses façons d'interagir avec SNS dans votre application Xamarin :

Envoyer des notifications Push (Xamarin Android)

Ce document explique comment envoyer des notifications push à une application Xamarin pour Android à l'aide d'Amazon Simple Notification Service (SNS) et du SDK mobile AWS pour .NET and Xamarin.

Configuration du projet

Prérequis

Avant de commencer ce didacticiel, vous devez suivre toutes les instructions de la section Configurer un kit SDK AWS Mobile pour .NET et Xamarin.

Définir les autorisations pour SNS

Suivez l'étape 2 de la section Configurer le kit SDK AWS Mobile pour .NET et Xamarin afin d'attacher la stratégie susmentionnée aux rôles de votre application. Votre application bénéficiera ainsi des autorisations appropriées pour accéder à SNS :

1. Accédez à la console IAM et sélectionnez le rôle IAM que vous souhaitez configurer.
2. Cliquez sur Attacher la politique, sélectionnez la politique Amazon SNSFull Access et cliquez sur Attacher la politique.

> **Warning**
>
> L'utilisation SNSFull d'Amazon Access n'est pas recommandée dans un environnement de production. Nous l'utilisons ici pour vous permettre d'être opérationnel rapidement. Pour en savoir plus sur la définition d'autorisations pour un rôle IAM, consultez Présentation des autorisations des rôles IAM.

Activer les notifications Push sur Google Cloud

Commencez par ajouter un projet d'API Google :

1. Accédez à la console Google de développement.
2. Cliquez sur Create Project (Créer un projet) .
3. Dans la zone New Project (Nouveau projet), entrez un nom de projet, notez son ID (vous en aurez besoin plus tard) et cliquez sur Create (Créer).

Ensuite, activez le service de messagerie de Google Cloud (GCM) pour le projet :

1. Dans la console des développeurs Google, votre nouveau projet est normalement déjà sélectionné. Dans le cas contraire, sélectionnez-le dans le menu déroulant en haut de la page.
2. Sélectionnez APIs & auth dans la barre latérale sur le côté gauche de la page.
3. Dans la zone de recherche, tapez « Google Cloud Messaging pour Android » et cliquez sur le lien du même nom.
4. Cliquez sur Enable API.

Enfin, obtenez une clé API :

1. Dans la console Google Developers, sélectionnez APIs & auth > Identifiants.
2. Sous Public API access, cliquez sur Create new key.
3. Dans la boîte de dialogue Create new key, cliquez sur Server Key.

4. Dans la boîte de dialogue qui s'affiche, cliquez sur Create, puis copiez la clé d'API affichée. Vous utiliserez cette clé API pour effectuer l'authentification par la suite.

Utilisez un identifiant de projet pour créer un ARN de plateforme dans la console SNS

1. Accédez à la console SNS.
2. Cliquez sur Applications sur le côté gauche de l'écran.
3. Cliquez sur Créer une application de plate-forme pour créer une nouvelle application de plateforme SNS.
4. Saisissez le nom de l'application.
5. Sélectionnez Google Cloud Messaging (GCM) comme plate-forme de notifications Push.
6. Collez la clé de l'API dans la zone de texte API key.
7. Cliquez sur Créer une application de plate-forme.
8. Sélectionnez l'application de plateforme que vous venez de créer et copiez l'ARN de l'application.

Ajoutez un NuGet package pour SNS à votre projet

Suivez l'étape 4 des instructions de la section Configuration du SDK AWS Mobile pour .NET and Xamarin pour ajouter le package NuGet Amazon Simple Notification Service à votre projet.

Créer un client SNS

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

Enregistrer votre application pour activer les notifications à distance

Pour vous inscrire aux notifications à distance sur Android, vous devez créer un BroadcastReceiver système capable de recevoir des messages Google Cloud. Modifiez le nom de package ci-dessous lorsque vous êtes invité à le faire :

```
[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);
      }
}
```

Vous trouverez ci-dessous le service qui reçoit la notification push du BroadcastReceiver et qui affiche la notification dans la barre de notification de l'appareil :

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

Envoyer un message à partir de la console SNS vers votre point de terminaison

1. Accédez à la Console SNS > Applications.
2. Sélectionnez votre application de plateforme, choisissez un point de terminaison et cliquez sur Publier sur le point de terminaison.
3. Pour publier un message, saisissez un message texte dans la zone de texte et cliquez sur Publier le message.

Envoyer des notifications Push (Xamarin iOS)

Ce document explique comment envoyer des notifications push à une application Xamarin iOS à l'aide d'Amazon Simple Notification Service (SNS) et du SDK AWS Mobile pour .NET and Xamarin.

Configuration du projet

Prérequis

Avant de commencer ce didacticiel, vous devez suivre toutes les instructions de la section Configurer un kit SDK AWS Mobile pour .NET et Xamarin.

Définir les autorisations pour SNS

Suivez l'étape 2 de la section Configurer le kit SDK AWS Mobile pour .NET et Xamarin afin d'attacher la stratégie susmentionnée aux rôles de votre application. Votre application bénéficiera ainsi des autorisations appropriées pour accéder à SNS :

1. Accédez à la console IAM et sélectionnez le rôle IAM que vous souhaitez configurer.
2. Cliquez sur Attacher la politique, sélectionnez la politique Amazon SNSFull Access et cliquez sur Attacher la politique.

> **Warning**
>
> L'utilisation SNSFull d'Amazon Access n'est pas recommandée dans un environnement de production. Nous l'utilisons ici pour vous permettre d'être opérationnel rapidement. Pour en savoir plus sur la définition d'autorisations pour un rôle IAM, consultez Présentation des autorisations des rôles IAM.

Obtenir l'adhésion au programme Apple iOS pour les développeurs

Pour recevoir des notifications Push, vous devez exécuter votre application sur un appareil physique. Pour exécuter votre application sur un appareil, vous devez adhérer au programme Apple iOS pour les développeurs. Une fois que vous disposez d'un abonnement, vous pouvez utiliser Xcode pour générer une identité de signature. Pour en savoir plus, consultez la documentation App Distribution Quick Start publiée par Apple.

Créer un certificat iOS

Tout d'abord, vous devez créer un certificat iOS. Ensuite, vous devez créer un profil de mise en service configuré pour recevoir des notifications Push. Pour ce faire :

1. Accédez au centre Apple Developer et cliquez sur Certificates, Identifiers & Profiles (Certificats, identifiants et Profils).
2. Cliquez sur Identifiers sous iOS Apps, cliquez sur le bouton Plus situé dans l'angle supérieur droit de la page pour ajouter un nouvel identifiant iOS App ID, puis saisissez une description de l'ID d'application.
3. Faites défiler la page jusqu'à la section Add ID Suffix, puis choisissez Explicit App ID et saisissez votre identifiant de groupe.
4. Faites défiler jusqu'à la section App Services et sélectionnez Push Notifications.
5. Cliquez sur Continuer.
6. Cliquez sur Soumettre.
7. Cliquez sur Done.
8. Sélectionnez l'ID d'application que vous venez de créer, puis cliquez sur Edit (Modifier).
9. Faites défiler jusqu'à la section Push Notifications. Cliquez sur Create Certificate (Créer certificat) sous Development SSL Certificate (Certificat SSL de développement).
10. Suivez les instructions pour créer une demande de signature de certificat (CSR, Certificate Signing Request), chargez la demande et téléchargez un certificat SSL qui sera utilisé pour communiquer avec Apple Notification Service (APNS).
11. Revenez à la page Certificates, Identifiers & Profiles (Certificats, Identifiants et Profils). Cliquez sur All (Tous) sous Provisioning Profiles (Profils de mise en service).
12. Cliquez sur le bouton Plus dans l'angle supérieur droit pour ajouter un nouveau profil de mise en service.
13. Sélectionnez iOS App Development (Développement d'application iOS), puis cliquez sur Continue (Continuer).
14. Sélectionnez l'ID de votre application, puis cliquez sur Continue (Continuer).
15. Sélectionnez votre certificat de développeur, puis cliquez sur Continue (Continuer).
16. Sélectionnez votre appareil, puis cliquez sur Continue (Continuer).
17. Entrez un nom de profil, puis cliquez sur Generate (Générer).
18. Téléchargez le fichier de mise en service et double-cliquez dessus pour installer le profil de mise en service.

Pour en savoir plus sur la mise en service d'un profil configuré pour les notifications Push, reportez-vous à la documentation Configuring Push Notifications sur le site d'Apple.

Utiliser le certificat pour créer un ARN de plate-forme dans la console SNS

1. Exécutez l'application KeyChain d'accès, sélectionnez Mes certificats dans le coin inférieur gauche de l'écran, puis cliquez avec le bouton droit sur le certificat SSL que vous avez généré pour vous connecter à APNS et sélectionnez Exporter. Vous serez invité à indiquer un nom pour le fichier et un mot de passe pour protéger le certificat. Le certificat sera enregistré dans un fichier P12.
2. Accédez à la console SNS et cliquez sur Applications sur la gauche de l'écran.
3. Cliquez sur Créer une application de plate-forme pour créer une nouvelle application de plateforme SNS.
4. Saisissez le nom de l'application.
5. Sélectionnez Apple Development comme plate-forme de notifications Push.
6. Cliquez sur Choisir un fichier et sélectionnez le fichier P12 que vous avez créé lorsque vous avez exporté votre certificat SSL.
7. Entrez le mot de passe vous avez spécifié lorsque vous avez exporté le certificat SSL et cliquez sur Charger à partir du fichier.
8. Cliquez sur Créer une application de plate-forme.
9. Sélectionnez l'application de plateforme que vous venez de créer et copiez l'ARN de l'application. Vous aurez besoin de ces informations pour la suite de la procédure.

Ajoutez un NuGet package pour SNS à votre projet

Suivez l'étape 4 des instructions de la section Configuration du SDK AWS Mobile pour .NET and Xamarin pour ajouter le package NuGet Amazon Simple Notification Service à votre projet.

Créer un client SNS

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

Enregistrer votre application pour activer les notifications à distance

Pour enregistrer une application, faites appel RegisterForRemoteNotifications à votre UIApplication objet, comme indiqué ci-dessous. Placez le code suivant dans le AppDelegate fichier .cs, en insérant l'ARN de votre application de plateforme comme demandé ci-dessous :

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

Envoyer un message à partir de la console SNS vers votre point de terminaison

1. Accédez à la Console SNS > Applications.
2. Sélectionnez votre application de plateforme, choisissez un point de terminaison et cliquez sur Publier sur le point de terminaison.
3. Pour publier un message, saisissez un message texte dans la zone de texte et cliquez sur Publier le message.

Envoyer et recevoir des notifications par SMS

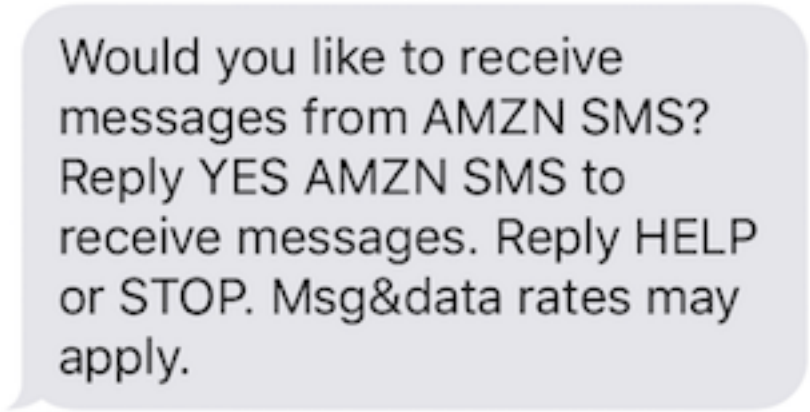
Vous pouvez utiliser Amazon Simple Notification Service (Amazon SNS) pour envoyer et recevoir des notifications par SMS sur les téléphones mobiles et les smartphones les prenant en charge.

> **Note**
>
> Les notifications par SMS sont actuellement prises en charge pour les numéros de téléphone aux États-Unis. Les messages SMS peuvent être envoyés uniquement à partir des rubriques créées dans la région USA Est (Virginie du Nord). Toutefois, vous pouvez publier des messages dans les rubriques que vous créez dans la région USA Est (Virginie du Nord) à partir de n'importe quelle autre région.

Création d'une rubrique

Pour créer une rubrique:

1. Dans la console Amazon SNS, cliquez sur Créer une rubrique. La boîte de dialogue Créer une rubrique s'ouvre.
2. Dans la zone Topic name, saisissez un nom de rubrique.
3. Entrez un nom dans la zone Nom d'affichage. La rubrique doit être associée à un nom d'affichage, car les 10 premiers caractères de ce nom sont utilisés dans la première partie du préfixe du message. Le nom d'affichage que vous entrez s'affiche dans le message de confirmation que SNS envoie à l'utilisateur (le nom d'affichage ci-dessous est « AMZN SMS »).



1. Cliquez sur Créer une rubrique. La nouvelle rubrique s'affiche dans la page Topics.
2. Sélectionnez la nouvelle rubrique et cliquez sur son ARN. La page Topic Details s'affiche.
3. Copiez l'ARN de rubrique, car vous en aurez besoin pour vous abonner à une rubrique à l'étape suivante.

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

S'abonner à une rubrique à l'aide du protocole SMS

Créez un client SNS en indiquant l'objet de vos informations d'identification et la région de votre groupe d'identités :

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

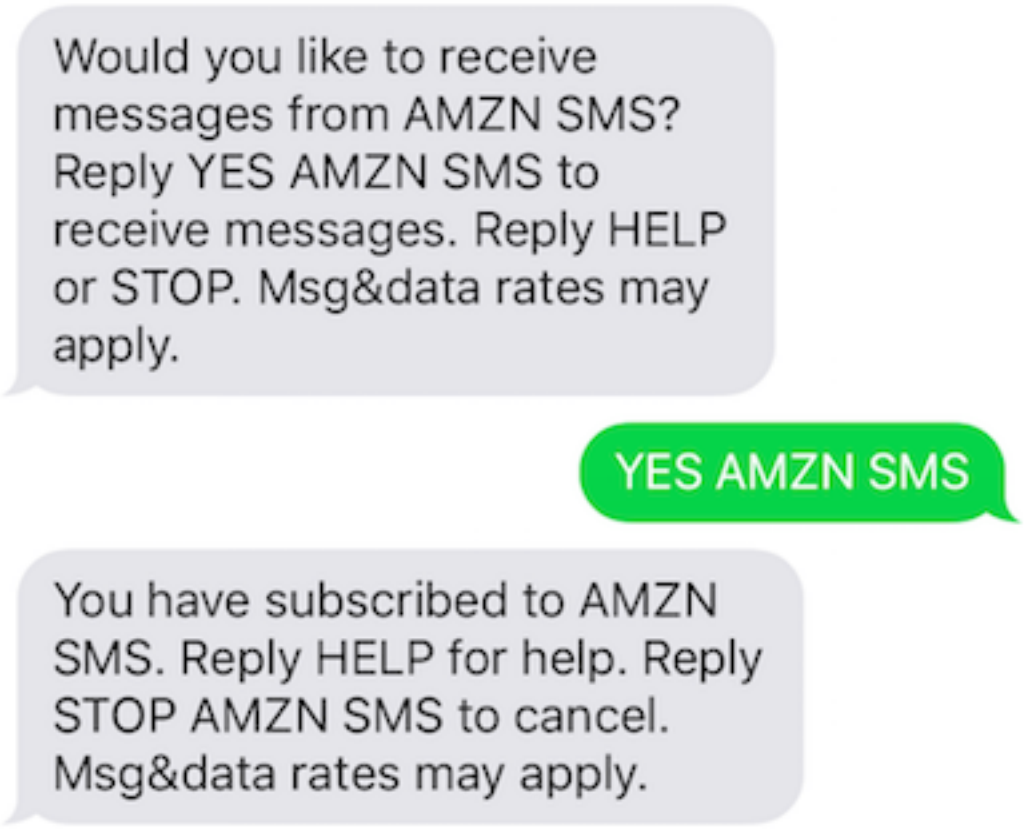
Pour vous abonner à une rubrique, appelez le code `SubscribeAsync` et indiquez l'ARN de la rubrique à laquelle vous souhaitez vous abonner, le protocole (« sms ») et le numéro de téléphone :

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

Vous recevrez un ARN d'abonnement dans l'objet de réponse d'abonnement. Votre ARN d'abonnement ressemble à ceci :

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

Lorsqu'un appareil s'abonne à une rubrique, SNS lui envoie un message de confirmation ; l'utilisateur doit alors confirmer qu'il veut recevoir des notifications, comme indiqué ci-dessous :



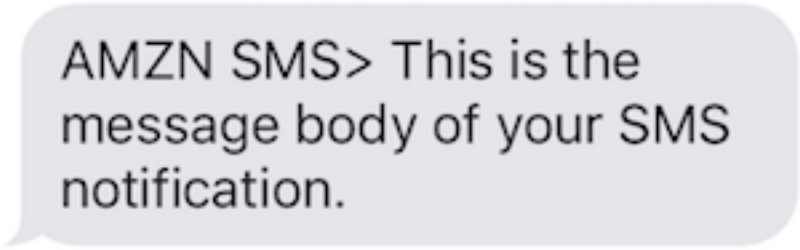
Une fois que l'utilisateur est abonné à la rubrique, il reçoit des messages SMS lorsque vous les publiez dans la rubrique en question.

Publier un message

Pour publier un message dans une rubrique :

1. Connectez-vous à AWS Management Console et ouvrez la console Amazon SNS.
2. Dans le volet de navigation de gauche, cliquez sur Rubriques, puis sélectionnez la rubrique dans laquelle vous voulez effectuer la publication.
3. Cliquez sur Publier dans la rubrique.
4. Dans la zone Subject (Objet), tapez un objet.
5. Dans la zone Message, saisissez un message. Amazon SNS envoie aux abonnés SMS le texte que vous avez saisi dans la zone Message, sauf si vous saisissez également du texte dans la zone d'objet. Etant donné qu'Amazon SNS inclut un préfixe de nom d'affichage pour tous les messages SMS que vous envoyez, la somme du préfixe et de la charge du message ne peut pas dépasser 140 caractères ASCII ou 70 caractères Unicode. Amazon SNS tronque les messages qui dépassent ces limites.

6. Cliquez sur Publier le message. Amazon SNS affiche une boîte de dialogue de confirmation. Le message SMS s'affiche sur votre appareil compatible SMS, comme indiqué ci-dessous.



Envoyer des messages à des points de terminaison HTTP/HTTPS

Vous pouvez utiliser Amazon SNS pour envoyer des messages de notification à un ou plusieurs points de terminaison HTTP ou HTTPS. Procédez comme suit :

1. Configurez votre point de terminaison de façon à recevoir des messages Amazon SNS.
2. Abonnez un point de terminaison HTTP/HTTPS à une rubrique.
3. Confirmez votre abonnement.
4. Publiez une notification dans la rubrique. Amazon SNS envoie ensuite une requête HTTP POST fournissant le contenu de la notification au point de terminaison abonné.

Configurer votre point de terminaison HTTP/HTTPS pour recevoir des messages Amazon SNS

Suivez les instructions indiquées à l'étape 1 de la procédure Envoi de messages Amazon SNS à des points de terminaison HTTP/HTTPS pour configurer votre point de terminaison.

Abonner votre point de terminaison HTTP/HTTPS à votre rubrique Amazon SNS

Créez un client SNS en indiquant l'objet de vos informations d'identification et la région de votre groupe d'identités :

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

Pour envoyer des messages à un point de terminaison HTTP ou HTTPS via une rubrique, vous devez abonner le point de terminaison à la rubrique Amazon SNS. Spécifiez le point de terminaison à l'aide de son URL:

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

Confirmer votre abonnement

Une fois que vous êtes abonné à un point de terminaison, Amazon SNS lui envoie un message de confirmation d'abonnement. Le code sur le point de terminaison doit récupérer la valeur `SubscribeURL` dans le message de confirmation d'abonnement, puis accéder à l'emplacement spécifié par l'URL `SubscribeURL` elle-même, ou le mettre à votre disposition afin que vous puissiez accéder manuellement à l'URL `SubscribeURL`, par exemple à l'aide d'un navigateur Web.

Amazon SNS n'envoie pas de messages au point de terminaison tant que l'abonnement n'a pas été confirmé. Lorsque vous accédez à `SubscribeURL`, la réponse contient un document XML comprenant un élément `SubscriptionArn` qui spécifie l'ARN de l'abonnement.

Envoyer des messages au point de terminaison HTTP/HTTPS

Vous pouvez envoyer un message aux abonnements d'une rubrique en effectuant une publication dans la rubrique. Appelez `PublishAsync` et indiquez l'ARN de la rubrique et votre message.

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

Dépannage de SNS

Utilisation de l'état de diffusion dans la console Amazon SNS

La console Amazon SNS contient une fonctionnalité d'état de diffusion qui vous permet de collecter des retours sur les tentatives de diffusion des messages réussies et avortées vers des plates-formes de notifications Push mobiles (Apple (APNS), Google (GCM), Amazon (ADM), Windows (WNS et MPNS) et Baidu).

Elle offre aussi d'autres informations importantes, telles que les temps de pause dans Amazon SNS. Ces informations sont capturées dans un groupe Amazon CloudWatch Log créé automatiquement par Amazon SNS lorsque cette fonctionnalité est activée via la console Amazon SNS ou via Amazon SNS. APIs

Pour plus d'informations sur l'utilisation de la fonctionnalité d'état de diffusion, consultez l'article Using the Delivery Status feature of Amazon SNS sur le blog d'AWS Mobile.

Bonnes pratiques d'utilisation du kit de développement logiciel AWS Mobile pour .NET and Xamarin

Seuls quelques principes fondamentaux et bonnes pratiques sont utiles à connaître lors de l'utilisation du SDK AWS Mobile pour .NET and Xamarin.

- Utilisez Amazon Cognito pour obtenir des informations d'identification AWS plutôt que de coder en dur vos informations d'identification dans votre application. Si vous codez en dur vos informations d'identification dans votre application, vous risquez de les exposer au public, et de laisser d'autres personnes émettre des appels vers AWS avec vos informations d'identification. Pour savoir comment utiliser Amazon Cognito afin d'obtenir des informations d'identification AWS, consultez la section Configurer le kit SDK AWS Mobile pour .NET et Xamarin.
- Pour connaître les bonnes pratiques relatives à l'utilisation de S3, consultez cet article sur le blog AWS.
- Pour connaître les bonnes pratiques relatives à l'utilisation de DynamoDB, reportez-vous à la section Bonnes pratiques pour DynamoDB dans le Manuel du développeur DynamoDB.

Nous cherchons toujours à aider nos clients à réussir et nous accueillons volontiers leurs commentaires. N'hésitez donc pas à publier sur les forums AWS ou à signaler un problème sur GitHub.

Bibliothèque de documentation sur les services AWS

Chaque service du SDK AWS Mobile pour .NET and Xamarin dispose d'un guide du développeur et d'une référence d'API de service distincts qui fournissent des informations supplémentaires susceptibles de vous être utiles.

Amazon Cognito Identity

- Manuel du développeur Cognito
- Référence d'API du service Amazon Cognito Identity

Amazon Cognito Sync

- Manuel du développeur Cognito

- Référence d'API du service Amazon Cognito Sync

Amazon Mobile Analytics

- Manuel du développeur Amazon Mobile Analytics
- Référence d'API du service Amazon Mobile Analytics

Amazon S3

- Manuel du développeur S3
- Manuel de mise en route S3
- Référence d'API du service S3

Amazon DynamoDB

- Manuel du développeur DynamoDB
- Manuel de mise en route DynamoDB
- Référence d'API du service DynamoDB

Amazon Simple Notification Service (SNS)

- Manuel du développeur SNS
- Référence d'API du service SNS

Autres liens utiles

- Glossaire AWS
- À propos des informations d'identification AWS

Dépannage

Cette rubrique décrit quelques idées pour résoudre les problèmes que vous pourriez rencontrer lors de l'utilisation du SDK AWS Mobile pour .NET and Xamarin.

Vérifier que le rôle IAM dispose des autorisations requises

Lorsqu'elle appelle les services AWS, votre application doit utiliser une identité issue d'un groupe d'identités Cognito. Chaque identité du groupe est associée à un rôle IAM (Identity and Access Management).

Un rôle est associé à un ou plusieurs fichiers de stratégie qui désignent les ressources AWS auxquelles ont accès les utilisateurs assignés à ce rôle. Par défaut, deux rôles sont créés par groupe d'identités : un pour les utilisateurs authentifiés, un autre pour les utilisateurs non authentifiés.

Vous aurez besoin de modifier le fichier de stratégie existant ou d'associer un nouveau fichier de stratégie aux autorisations requises par votre application. Si votre application autorise les utilisateurs authentifiés et non authentifiés, les deux rôles doivent disposer des autorisations pour l'accès aux ressources AWS dont votre application a besoin.

Le fichier de stratégie suivant montre comment accorder l'accès à un compartiment S3 :

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

Le fichier de stratégie suivant montre comment accorder l'accès à une base de données DynamoDB :

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

Pour plus d'informations sur la spécification de stratégies, consultez Stratégies IAM.

Utilisation d'un débogueur proxy HTTP

Si le service AWS que votre application appelle dispose d'un point de terminaison HTTP ou HTTPS, vous pouvez utiliser un débogueur de proxy HTTP/HTTPS pour afficher les requêtes et les réponses afin de mieux comprendre ce qui se passe. Il existe un certain nombre de débogueurs proxy HTTP tels que :

- Charles : proxy de débogage Web pour Windows et OSX
- Fiddler : proxy de débogage web pour Windows

Charles et Fiddler nécessitent tous les deux un peu de travail de configuration pour pouvoir afficher le trafic SSL chiffré. C'est pourquoi il est préférable de lire la documentation sur ces outils pour plus d'informations. Si vous utilisez un proxy de débogage Web qui ne peut pas être configuré de manière à afficher le trafic chiffré, ouvrez le fichier aws_endpoints_json et attribuez la valeur true à la balise HTTP pour le service AWS que vous devez déboguer.

Historique du document

Le tableau suivant décrit les modifications importantes apportées à la documentation depuis la dernière version du SDK AWS Mobile pour .NET and Xamarin.

- Version de l'API : 27/08/2015
- Dernière mise à jour de la documentation : 2021-02-23

Modification	Version de l’API	Description	Date de publication
Archivé	27/08/2015	Le SDK AWS mobile pour Xamarin est inclus dans le. AWS SDK pour .NET Ce guide fait référence à la version archivée du SDK mobile pour Xamarin.	23/02/2021
Version GA	27/08/2015	Version GA	27/08/2015
Version bêta	27/08/2015	Version bêta	rn 28/07/2015