Setting Up the AWS Mobile SDK for .NET and Xamarin

You can set up the AWS Mobile SDK for .NET and Xamarin and start building a new project or you can integrate the SDK with an existing project. You can also clone and run the samples (https://github.com/awslabs/aws-sdk-net-samples/tree/master/XamarinSamples) to get a sense of how the SDK works. Follow these steps to set up and start using the AWS Mobile SDK for .NET and Xamarin.

Prerequisites

Before you can use the AWS Mobile SDK for .NET and Xamarin, you must do the following:

- Create an An AWS Account (https://aws.amazon.com/) .
- Install the Xamarin platform (https://xamarin.com/) .
- **Windows users:** complete Xamarin: Getting Started for Windows (http://developer.xamarin.com/guides/cross-platform/windows/)
- **Mac users:** complete Xamarin: Getting Started for Mac (http://developer.xamarin.com/guides/mac/getting_started/installation/) .

After you complete the prerequisites:

1. Obtain AWS credentials by using Amazon Cognito.
2. Set the required permissions for each AWS service that you will use in your application.
3. Create a new project in your IDE.
4. Install the AWS Mobile SDK for .NET and Xamarin.
5. Configure the AWS Mobile SDK for .NET and Xamarin.

Step 1: Obtain AWS Credentials

To make calls to AWS in your application, you must first obtain AWS credentials. You do this by using Amazon Cognito, an AWS service that allows your application to access the services in the SDK without having to embed your private AWS credentials in the application.

To get started with Amazon Cognito, you need to create an identity pool. An identity pool is a store of information that is specific to your account and is identified by a unique identity pool ID that looks like the following.:

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

1. Log in to the Amazon Cognito Console (https://console.aws.amazon.com/cognito/home), choose **Manage Federated Identities**, and then choose **Create new identity pool**.
2. Enter a name for your identity pool and select the checkbox to enable access to unauthenticated identities. Choose **Create Pool** to create your identity pool.
3. Choose **Allow** to create the two default roles associated with your identity pool, one for unauthenticated users and one for authenticated users. These default roles provide your identity pool access to Amazon Cognito Sync and Amazon Mobile Analytics.

Typically, you will only use one identity pool per application.

After you create your identity pool, you obtain AWS credentials by creating a `CognitoAWSCredentials` object (passing it your identity pool ID) and then passing it to the constructor of an AWS client as follows.:

```
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
);
```

Step 2: Set Permissions

You need to set permissions for every AWS service that you want to use in your application. First, you need to understand how AWS views the users of your application.

When someone uses your application and makes calls to AWS, AWS assigns that user an identity. The identity pool that you created in Step 1 is where AWS stores these identities. There are two types of identities: authenticated and unauthenticated. Authenticated identities belong to users who are authenticated by a public login

provider (e.g., Facebook, Amazon, Google). Unauthenticated identities belong to guest users.

Every identity is associated with an AWS Identity and Access Management role. In Step 1, you created two IAM roles, one for authenticated users and one for unauthenticated users. Every IAM role has one or more policies attached to it that specify which AWS services the identities assigned to that role can access. For example, the following sample policy grants access to an Amazon S3 bucket.:

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

To set permissions for the AWS services that you want to use in your application, simply modify the policy attached to the roles.

1. Go to the IAM Console and choose Roles (https://console.aws.amazon.com/iam/home) . Type your identity pool name into the search box. Choose the IAM role that you want to configure. If your application allows both authenticated and unauthenticated users, you need to grant permissions for both roles.

2. Click **Attach Policy**, select the policy you want, and then click **Attach Policy**. The default policies for the IAM roles that you created provide access to Amazon Cognito Sync and Mobile Analytics.

For more information about creating policies or to choose from a list of existing policies, see IAM Policies (http://docs.aws.amazon.com/IAM/latest/UserGuide/PoliciesOverview.html) .

Step 3: Create a New Project

Windows

You can use either Visual Studio or Xamarin Studio to develop your application.

OS X

You must use the Xamarin Studio IDE to develop your applications. iOS development using Xamarin requires access to a Mac to run your app. For more information, see Installing Xamarin.iOS on Windows (http://developer.xamarin.com/guides/ios/getting_started/installation/windows) .

Step 4: Install the AWS Mobile SDK for .NET and Xamarin

Windows

Option 1: Install by Using the Package Manager Console

The AWS Mobile SDK for .NET and Xamarin consists a set of .NET assemblies. To install the AWS Mobile SDK for .NET and Xamarin, run the install-package command for each package in the Package Manager console. For example, to install Cognito Identity, run the following.:

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

The AWS Core Runtime and Amazon Cognito Identity packages are required for all projects. The following is a full list of package names for each service.

Service	Package name
AWS Core Runtime	AWSSDK.Core
Amazon Cognito Sync	AWSSDK.CognitoSync
Amazon Cognito Identity	AWSSDK.CognitoIdentity
Amazon DynamoDB	AWSSDK.DynamoDBv2
Amazon Mobile Analytics	AWSSDK.MobileAnalytics
Amazon S3	AWSSDK.S3
Amazon SNS	AWSSDK.SimpleNotificationService

To include a prerelease package, include the `-Pre` command line argument while installing the package as follows.:

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

You can find a complete list of AWS service packages at AWS SDK packages on NuGe (https://www.nuget.org/packages?q=+aws-sdk-v3) or at the AWS SDK for .NET Github Repository (https://github.com/aws/aws-sdk-net#nuget-packages) .

Option 2: Install by Using Your IDE

In Visual Studio

1. Right-click the project, and then click **Manage NuGet Packages**.
2. Search for the package name that you want to add to your project. To include the prelease NuGet packages, choose **Include Prelease**. You can find a complete list of AWS service packages at AWS SDK packages on NuGet (https://www.nuget.org/packages?q=+aws-sdk-v3) .
3. Choose the package, and then choose **Install**.

In Xamarin Studio

1. Right-click the packages folder, and then choose **Add Packages**.
2. Search for the package name that you want to add to your project. To include the prelease NuGet packages, choose **Show pre-release packages**. You can find a complete list of AWS service packages at AWS SDK packages on NuGet (https://www.nuget.org/packages?q=+aws-sdk-v3) .
3. Select the checkbox next to the package you want, and then choose **Add Package**.

Mac (OS X)

In Xamarin Studio

1. Right-click the packages folder, and then choose **Add Packages**.
2. Search for the package name that you want to add to your project. To include the prelease NuGet packages, choose **Show pre-release packages**. You can find a complete list of AWS service packages at AWS SDK packages on NuGet (https://www.nuget.org/packages?q=+aws-sdk-v3) .
3. Select the checkbox next to the package you want, and then choose **Add Package**.

> **Important**
>
> If you are developing using a Portable Class Library, you must also add the AWSSDK.Core NuGet package to all projects deriving from the Portable Class Library.

Step 5: Configure the AWS Mobile SDK for .NET and Xamarin

Set Logging

You set logging settings by using the `Amazon.AWSConfigs` class and the `Amazon.Util.LoggingConfig` class. You can find these in the `AWSSdk.Core` assembly, available through the Nuget Package Manager in Visual Studio. You can place the logging settings code in the `OnCreate` method in the `MainActivity.cs` file for Android apps or the `AppDelegate.cs` file for iOS apps. You should also add `using Amazon` and `using Amazon.Util` statements to the .cs files.

Configure logging settings as follows.:

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

When you log to SystemDiagnostics, the framework internally prints the output to the System.Console. If you want to log HTTP responses, set the `LogResponses` flag. The values can be Always, Never, or OnError.

You can also log performance metrics for HTTP requests by using the `LogMetrics` property. The log format can be specified by using `LogMetricsFormat` property. Valid values are JSON or standard.

Set the Region Endpoint

Configure the default region for all service clients as follows.:

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

This sets the default region for all the service clients in the SDK. You can override this setting by explicitly specifying the region at the time of creating an instance of the service client, as follows.:

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

Configure the HTTP Proxy Settings

If your network is behind a proxy, you can configure the proxy settings for the HTTP requests as follows.

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

Correct for Clock Skew

This property determines if the SDK should correct for client clock skew by determining the correct server time and reissuing the request with the correct time.

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

This field is set if a service call resulted in an exception and the SDK has determined that there is a difference between local and server times.

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

To learn more about clock skew, see Clock-skew Correction (https://blogs.aws.amazon.com/net/post/Tx2HM54KL5LMTGI/Clock-skew-correction) on the AWS Blog.

Next Steps

Now that you have set up the AWS Mobile SDK for .NET and Xamarin, you can:

- **Get started.** Read Getting Started with the AWS Mobile SDK for .NET and Xamarin (getting-started-xamarin.html) for quick-start instructions about how to use and configure the services in the AWS Mobile SDK for .NET and Xamarin.
- **Explore the service topics.** Learn about each service and how it works in the AWS Mobile SDK for .NET and Xamarin.
- **Run the demos.** View our sample Xamarin applications (https://github.com/awslabs/aws-sdk-net-samples/tree/master/XamarinSamples) that demonstrate common use cases. To run the sample apps, set up the AWS Mobile SDK for .NET and Xamarin as described previously, and then follow the instructions contained in the README files of the individual samples.
- **Learn the APIs.** View the |sdk-xamarin-ref|_ (http://docs.aws.amazon.com/sdkfornet/v3/apidocs/) .
- **Ask questions**: Post questions on the AWS Mobile SDK Forums (https://forums.aws.amazon.com/forum.jspa?forumID=88) or open an issue on Github (https://github.com/awslabs/aws-sdk-xamarin/issues) .