Sending Emails with AWS SES and .NET

# Overview

The Amazon Simple Email Service (SES) is an email platform that provides an easy, cost-effective way to send and receive email without managing mail servers or other infrastructure. It’s integrated with the AWS SDKs, so programmatically sending emails from your .NET applications is fast and easy. You can send email with SES using either the APIs, or SMTP.

For this walk-through, we’ll use the AWS Management Console to do the one-time verification of the email address we’ll use to send mail from. Next, we’ll create a new .NET console application (you can use either the .NET Framework or .NET Core) and add the AWS SDK for SES to it. Finally, we’ll send some email using our app!

# Prerequisites

* .NET Framework 3.5 or higher, **or** .NET Core 1.0 or higher installed
* AWS Account with credentials configured locally in Visual Studio or using the CLI
* Optional: Visual Studio 2017 (you can also use the command line for .NET Core)

# Verify an Email Address in SES

Amazon SES requires you to verify either the email addresses or the entire domain that you use to send email from, or that you use in the return address. For this walk-through, we’ll just confirm a single email address.

*Note: Email addresses are case-sensitive. If you verify* [*sender@EXAMPLE.com*](mailto:sender@EXAMPLE.com)*, you can’t send from* [*sender@example.com*](mailto:sender@example.com) *without also verifying that address.*

**Step 1: Select a Region**

Amazon SES is a regional service. Log into the AWS Management Console, and using the region-selector at the top-right, select the region you want to verify your email address in.

**Step 2: Verify an Email Address**

1. Go to SES dashboard. In the navigation pane, under **Identity Management**, select **Email Addresses**.
2. Choose **Verify a New Email Address**.
3. In the **Verify a New Email Address** dialog box, type your email address in the **Email Address** field, and then choose **Verify This Email Address**.
   1. Note: If your SES service has only “sandbox” access, you have to verify both sending and receiving email addresses for the email delivery to be successful.
4. You should receive an email with the subject line, “Amazon Web Services – Email Address Verification Request in region *<RegionName>*.” Click the link in the body of the message.
5. Back in the AWS SES console, verify that your email status is “verified”. Click the refresh icon at the top-right of the email list if necessary. See Figure 1 below.

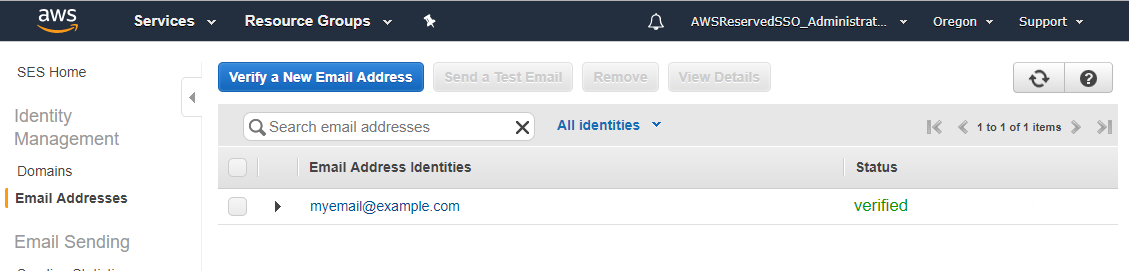


Figure 1 – Verified Email Address in SES Console

*Note: If you don’t see the email verification email in your inbox, follow the* [*troubleshooting steps in the SES Developer Guide*](https://docs.aws.amazon.com/ses/latest/DeveloperGuide/verify-email-addresses-procedure.html#verify-email-addresses-troubleshooting)*.*

# Create and Configure a Console Application

**Step 1: Create an empty console application project**

You can create, build and run the application using either Visual Studio (.NET Framework or .NET Core), or from the command line (.NET Core only). Follow the steps under the relevant heading below.

**Visual Studio:**

1. Select File 🡪 New 🡪 Project from the main menu.
2. In the New Project dialog, select Console App (.NET Framework) **or** Console App (.NET Core).  
   Note that in Figure 2 below, we have chosen .NET Core.
3. Select a name for your project, and then click the “Ok” button.

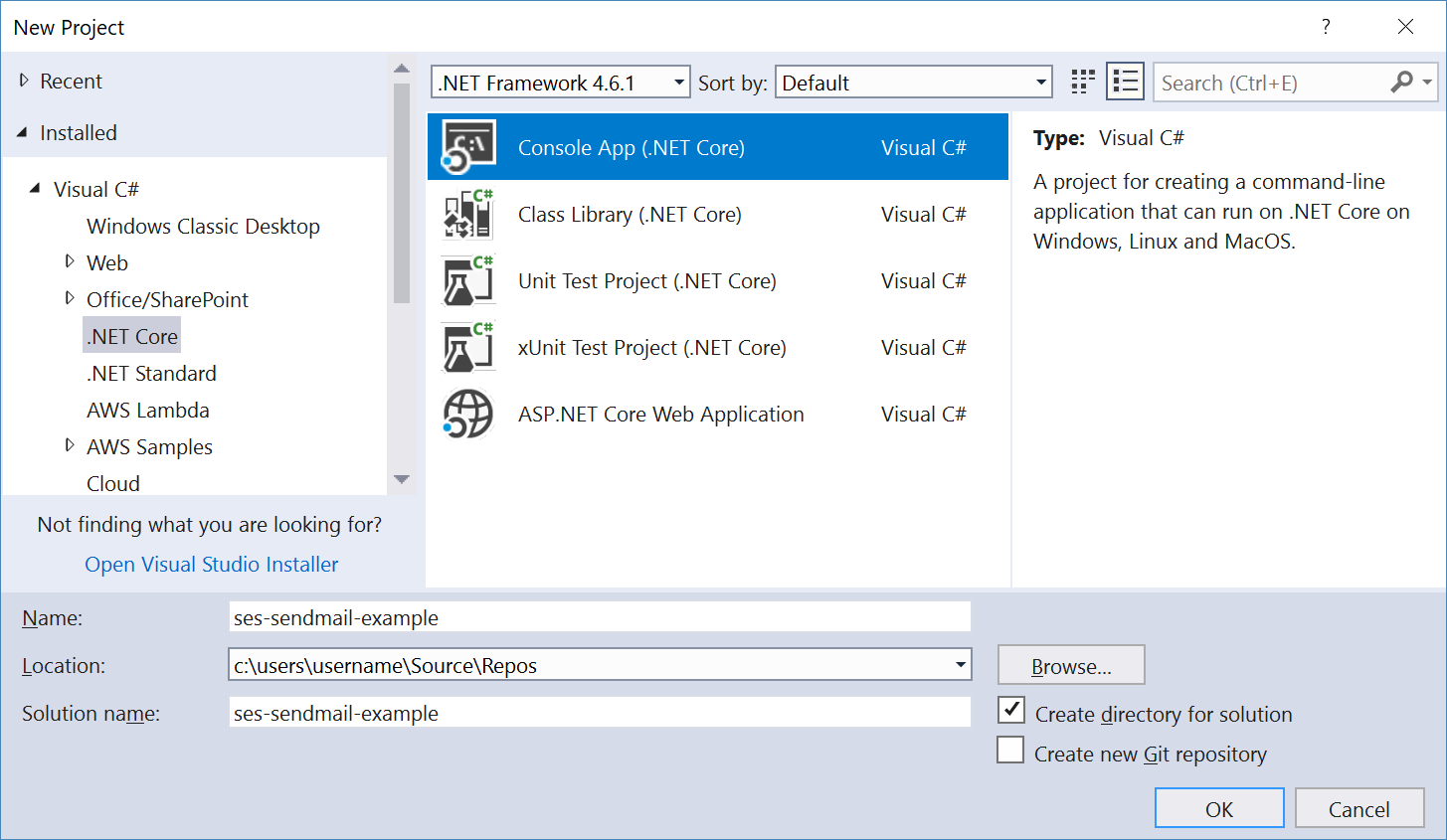


Figure 2 – New Console App in Visual Studio New Project Dialog

**Command Line:**

We’ll create a new directory, and then create the console app project in it using the following commands in either a Windows command prompt or Mac OS X or Linux bash shell:

1. mkdir ecs-sample
2. cd ecs-sample
3. dotnet new console

The *dotnet new* command will create the project files, and restore packages referenced by the template.

**Step 2: Add the SES NuGet Package**

**Visual Studio:**

1. Right-click the project node for your project in the Solution Explorer, and select, “Manage NuGet Packages…” from the context menu.
2. In the NuGet tab that appears, click “Browse” from the horizontal menu, and in the search box type “AWSSDK.SimpleEmail”.
3. When the package appears in the list area, select it, and then click the “Install” button in the details pane (see Figure 3 below).
4. If prompted to accept the license agreement, click the “I Accept” button in the dialog.

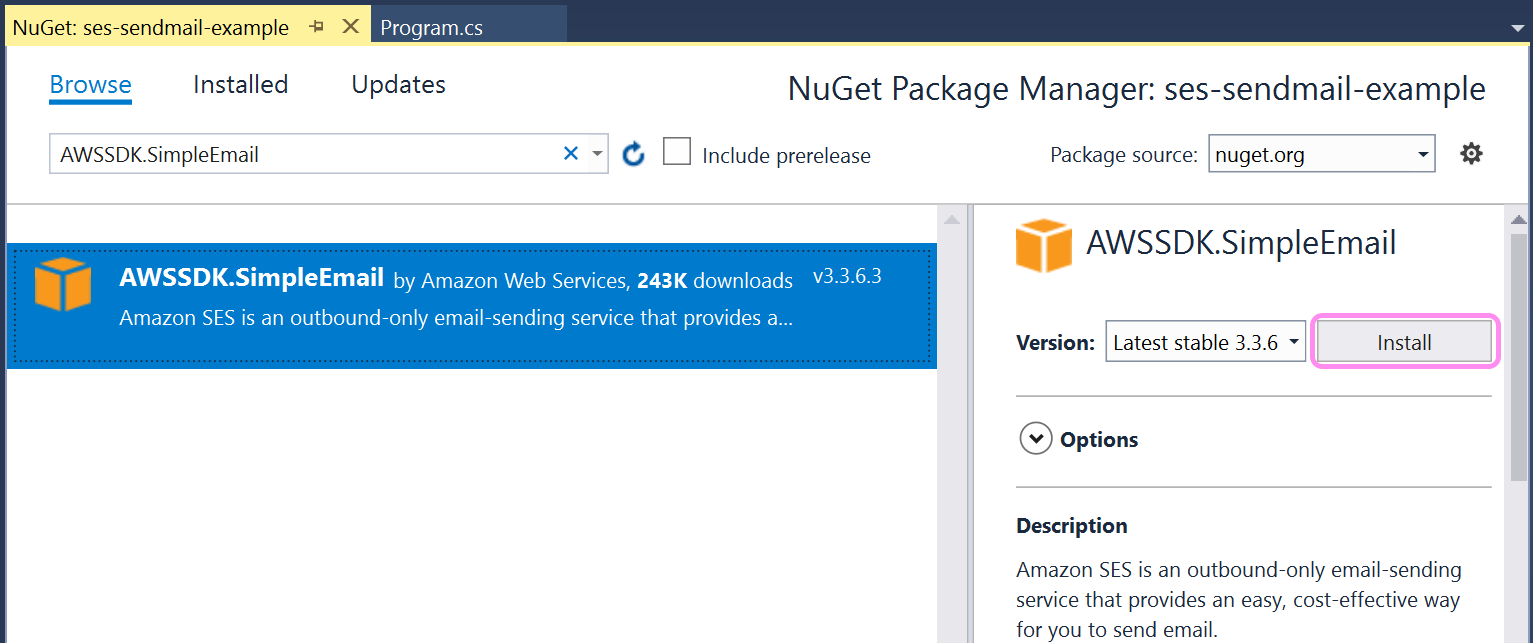


Figure 3 – AWS SES NuGet Package in Visual Studio

**Command Line:**

Add the NuGet package *AWSSDK.SimpleEmail* to the project with the following command in either a Windows command prompt or Mac OS X or Linux bash shell:

1. dotnet add package AWSSDK.SimpleEmail

**Step 3: Edit the C# Code**

You can download the complete project and then edit the *Program.cs* file, or copy and paste the code in Figure 4 below into the *Program.cs* file of your new project using either Visual Studio or a code editor of your choice.

using System;

using Amazon.SimpleEmail;

using Amazon.SimpleEmail.Model;

namespace ses\_sendmail\_example

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Sending Email...");

using (var client = new AmazonSimpleEmailServiceClient(region: Amazon.RegionEndpoint.USEast1))

{

var sendRequest = new SendEmailRequest

{

Source = "verified-email-address@example.com",

Destination = new Destination { ToAddresses = {"dest@example.com"} },

Message = new Message

{

Subject = new Content("Hello from the Amazon Simple Email Service!"),

Body = new Body

{

Html = new Content("<html><body><h2>Hello from Amazon SES</h2><ul><li>I'm a list item</li><li>So am I!</li></body></html>")

}

}

};

try

{

var response = client.SendEmailAsync(sendRequest).Result;

Console.WriteLine("Email sent! Message ID = {0}", response.MessageId);

} catch (Exception ex)

{

Console.WriteLine("Send failed with exception: {0}", ex.Message);

}

}

Console.Write("Press any key to continue...");

Console.ReadKey();

}

}

}

Figure 4 – Program.cs file

This sample code demonstrates a common pattern when developing with the AWS SDK for .NET, which is the use of a client object to represent an AWS service. That client object then exposes functionality via methods, such as the “*SendEmailAsync*” method in this example.

Ensure you make the following changes in the code:

* Replace “*verified-email-address@example.com*” with the email address you verified earlier
* Replace “*dest@example.com*” with your email address (where emails will be sent)
* Replace the region parameter in the AmazonSimpleEmailServiceClient instantiation with the region you would like to send from. In our example, we have it set to us-east-1.

*Note: For our sample app, we’re sending an HTML-formatted email. If you wish to send a plain text email, just change the email body property to be* new Content(“your text here”)

# Send Emails!

To send email using the console app, just run it using either Visual Studio or the command line.

**Visual Studio:**

Just press F5 to build and run the app.

**Command Line:**

Use the following command to build (compile) and run the app:

dotnet run

If you want to run the app again without compiling, just pass the --no-build flag like this:

dotnet run --no-build

The console application will run. It will send an email to the destination email address you specified in *Program.cs*, and then wait for you to press any key before exiting. Be sure to check your spam folder if you don’t see the email in your inbox.