**Enhance PDF Security with Timestamping Via External Signing in .NET Core**

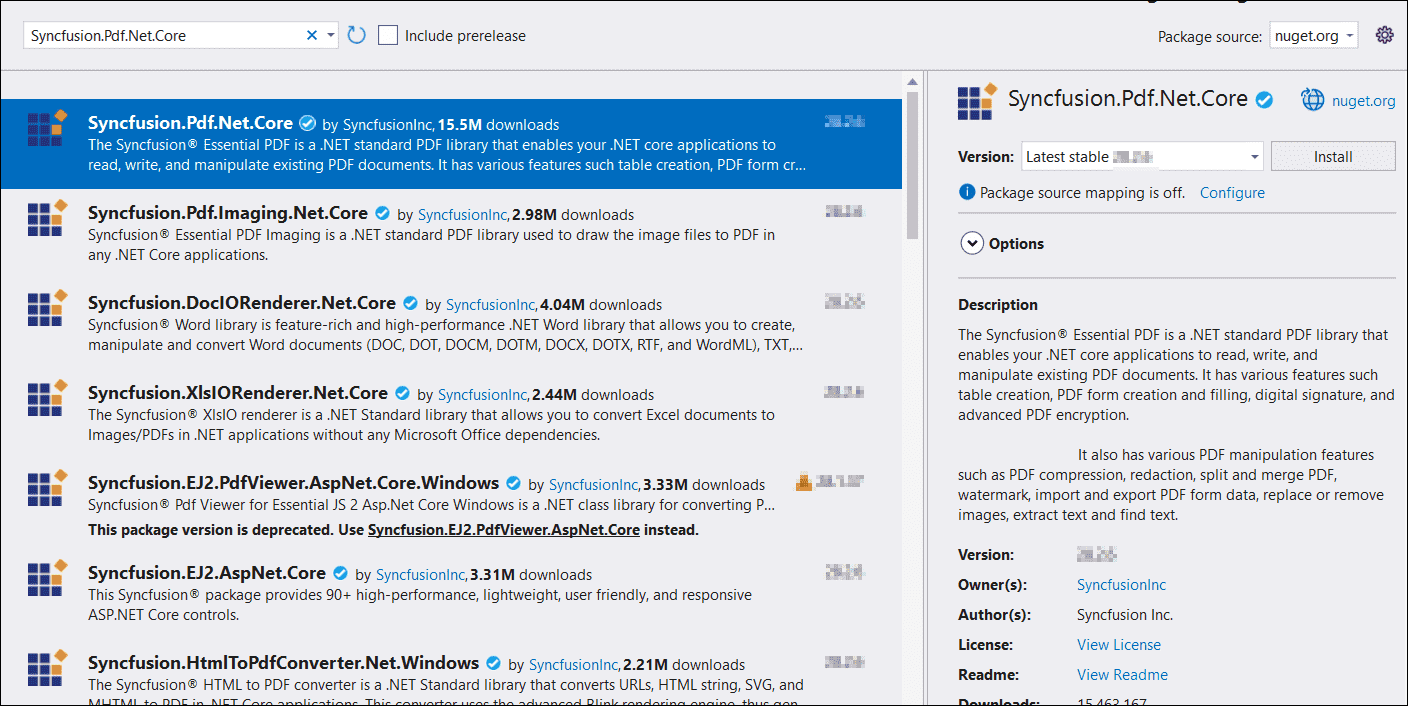
Syncfusion Essential® PDF provides a comprehensive [.NET PDF library](https://www.syncfusion.com/document-processing/pdf-framework/net-core) to create, read, and edit PDF documents programmatically without relying on Adobe dependencies. This guide illustrates how to enhance your PDF document's security by adding timestamps via external signing using C#.

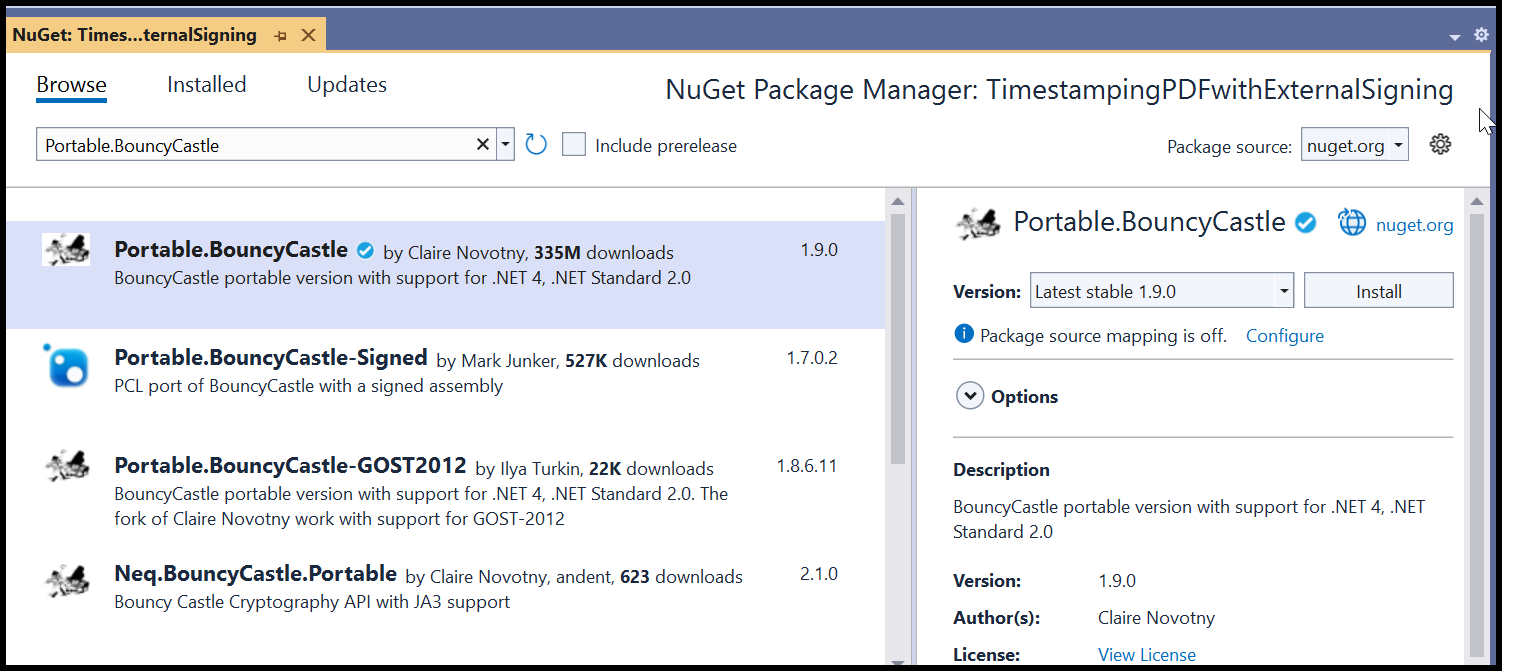
**Steps to Add Timestamps to a PDF Document via External Signing**

1. **Set Up Your Project**: Create a new console application project in your development environment.A screenshot of a computer

AI-generated content may be incorrect.

2. **Install Syncfusion Package**: Add the [Syncfusion.Pdf.Net.Core](https://www.nuget.org/packages/Syncfusion.Pdf.Net.Core/) and [Portable.BouncyCastle](https://www.nuget.org/packages/Portable.BouncyCastle) package from NuGet to your project.





In this example, the open-source [BouncyCastle](https://www.nuget.org/packages/Portable.BouncyCastle) library is used. Be sure to review its licensing terms before including it in a production environment. Alternatively, you may use a timestamp token provided by your trusted certificate authority or service provider.

3. **Include Required Namespaces**: Add these namespaces in your **Program.cs**:

**C#**

|  |
| --- |
| **using** Syncfusion.Pdf.Parsing;  **using** Syncfusion.Pdf.Security;  **using** Syncfusion.Drawing;  **using** System.Security.Cryptography.X509Certificates;  **using** System.Security.Cryptography;  **using** Org.BouncyCastle.Tsp;  **using** System.Net;  **using** System.Text;  **using** Org.BouncyCastle.Math; |

4. **Add Timestamps During External Signing**: Implement the following code to add timestamps during the external signing process

**C#**

|  |
| --- |
| // Get the stream from the input PDF document.  FileStream documentStream = new FileStream("Input.pdf", FileMode.Open, FileAccess.Read);  // Load the existing PDF document.  PdfLoadedDocument loadedDocument = new PdfLoadedDocument(documentStream);  // Create a digital signature for the first page of the document.  PdfSignature signature = new PdfSignature(loadedDocument, loadedDocument.Pages[0], null, "Signature");  signature.Bounds = new RectangleF(new PointF(0, 0), new SizeF(100, 30));  signature.Settings.CryptographicStandard = CryptographicStandard.CADES;  signature.Settings.DigestAlgorithm = DigestAlgorithm.SHA1;  IPdfExternalSigner externalSignature = new ExternalSigner("SHA1");  List<X509Certificate2> certificates = new List<X509Certificate2>  {  new X509Certificate2(Convert.FromBase64String(PublicCert))  };  signature.AddExternalSigner(externalSignature, certificates, null);  using (FileStream outputFileStream = new FileStream(Path.GetFullPath(@"Output.pdf"), FileMode.Create, FileAccess.ReadWrite))  {  loadedDocument.Save(outputFileStream);  }  loadedDocument.Close(true);  } |

To sign the document using the **X509Certificate2** API and generate an RFC3161-compliant timestamp token with the [**BouncyCastle**](https://www.nuget.org/packages/Portable.BouncyCastle) library, add the following code:

**C#**

|  |
| --- |
| // External signer class that implements IPdfExternalSigner for signing the document hash.  class ExternalSigner : IPdfExternalSigner  {  private string \_hashAlgorithm;  public string HashAlgorithm => \_hashAlgorithm;  public ExternalSigner(string hashAlgorithm) => \_hashAlgorithm = hashAlgorithm;  public byte[] Sign(byte[] message, out byte[] timeStampResponse)  {  byte[] signedBytes = null;  X509Certificate2 digitalID = new X509Certificate2(Path.GetFullPath(@"Data/PDF.pfx"), "password123");  if (digitalID.PrivateKey is RSACryptoServiceProvider rsa)  {  signedBytes = rsa.SignData(message, HashAlgorithm);  }  else if (digitalID.PrivateKey is RSACng rsaCng)  {  signedBytes = rsaCng.SignData(message, HashAlgorithmName.SHA1, RSASignaturePadding.Pkcs1);  }  timeStampResponse = GetRFC3161TimeStampToken(signedBytes);  return signedBytes;  }  public byte[] GetRFC3161TimeStampToken(byte[] bytes)  {  SHA1 sha1 = SHA1CryptoServiceProvider.Create();  byte[] hash = sha1.ComputeHash(bytes);  TimeStampRequestGenerator reqGen = new TimeStampRequestGenerator();  reqGen.SetCertReq(true);  TimeStampRequest tsReq = reqGen.Generate(TspAlgorithms.Sha1, hash, BigInteger.ValueOf(100));  byte[] tsData = tsReq.GetEncoded();  HttpWebRequest req = (HttpWebRequest)WebRequest.Create("https://rfc3161.ai.moda");  req.Method = "POST";  req.ContentType = "application/timestamp-query";  req.Headers.Add("Authorization", "Basic " + Convert.ToBase64String(Encoding.ASCII.GetBytes("9024:yourPass")));  req.ContentLength = tsData.Length;  using (Stream reqStream = req.GetRequestStream())  reqStream.Write(tsData, 0, tsData.Length);    using (HttpWebResponse res = (HttpWebResponse)req.GetResponse())  {  using (Stream resStream = new BufferedStream(res.GetResponseStream()))  {  TimeStampResponse tsRes = new TimeStampResponse(resStream);  return tsRes.TimeStampToken.GetEncoded();  }  }  } |

A complete working sample can be downloaded from [**GitHub**](https://github.com/SyncfusionExamples/PDF-Examples/tree/master/Digital%20Signature/TimestampingPDFwithExternalSigning/).  
By executing the program, the output PDF document will be generated as shown below.

A screenshot of a computer

AI-generated content may be incorrect.

Take a moment to explore the [documentation](https://help.syncfusion.com/document-processing/pdf/pdf-library/net/working-with-digitalsignature#adding-timestamps-to-a-pdf-document-using-external-signing) for adding timestamps to a PDF document using external signing.

**Conclusion**  
I hope you enjoyed learning on how to add timestamps to a PDF document via external signing

You can refer to our [**ASP.NET Core PDF feature tour**](https://www.syncfusion.com/document-processing/pdf-framework/net-core) page to know about its other groundbreaking feature representations and [**documentation**](https://help.syncfusion.com/file-formats/pdf/create-pdf-file-in-asp-net-core), and how to quickly get started for configuration specifications. You can also explore our [**ASP.NET Core PDF example**](https://ej2.syncfusion.com/aspnetcore/PDF/Default#/bootstrap5) to understand how to create and manipulate data.

For current customers, you can check out our components from the [**License and Downloads**](https://www.syncfusion.com/sales/teamlicense) page. If you are new to Syncfusion®, you can try our 30-day [**free trial**](https://www.syncfusion.com/downloads/aspnetcore-js2) to check out our other controls.

If you have any queries or require clarifications, please let us know in the comments section below. You can also contact us through our [**support forums**](https://www.syncfusion.com/forums), [**Direct-Trac**](https://support.syncfusion.com/create), or [**feedback portal**](https://www.syncfusion.com/feedback/aspnet-core?control=pdf). We are always happy to assist you!