



UWP



Electronic signatures



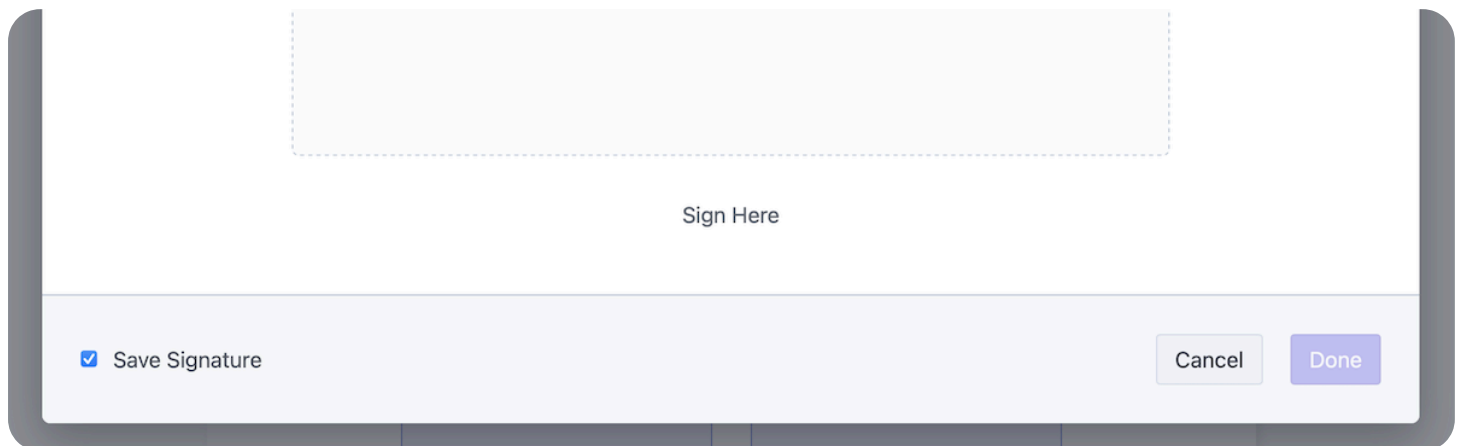
WINDOWS > GUIDES > SIGNATURES > ELECTRONIC SIGNATURES

Store and manage electronic signatures in UWP

Nutrient allows you to implement your own mechanism for storing signatures.

If you provide such a mechanism, then signatures may optionally be saved at the same time they're added to a document. The saving of signatures is based on the

`Controller.ElectronicSignatureStorage` property, which allows the user to choose whether or not to save a signature in the UI.



If you provide stored signatures to Nutrient, then when the user selects a signature form field or the signature tool, the list of stored signatures will be shown instead of the signature creation UI.





To achieve this, you can set the `ElectronicSignatureStorage` property of the `Controller` class. Once this property is set, the signatures UI is updated to display a checkbox to allow users to decide whether or not they want their signatures stored.

The `OnSignatureCreated` method is invoked with the signature annotation created by the user. There, you can serialize the signature and send it to a backend server, or add it to any other storage mechanism such as an SQL database or `LocalSettings`. The `OnSignatureDeleted` method is invoked when the annotation is deleted, and should instead remove the passed annotation from the storage.

Your implementation of the `IElectronicSignatureStorage` interface should also set the `InitialSignatures` property. This property describes the contents of the storage and will be used by the UI to display signatures to the user the first time they open the signature UI.

As an example, the code below shows how to implement the `IElectronicSignatureStorage` interface to persist signatures by serializing them into JSON and then saving them to a file. Note that this code isn't ideal for production applications, since its main purpose is to illustrate how to handle the callbacks and not to deal with performance and resilience:

```
1 public sealed class ExternalJsonFileStorage : IElectronicSignatureStorage
2 {
3     private readonly string fileName;
4
5     // Signatures that are initially loaded when a document is opened.
6     // Note that this isn't updated automatically, and, if set, is always used w
7     public IList<IAnnotation> InitialSignatures { get; set; }
8 }
```

```


9     private ExternalJsonFileStorage(string fileName, IList<IAnnotation> initialS
10     {
11         this.fileName = fileName;
12         InitialSignatures = initialSignatures;
13     }
14
15     public async void OnSignatureCreated(IAnnotation signature)
16     {
17         var signatures = await ReadSignaturesFromFile(fileName);
18         signatures.Add(signature);
19         await WriteSignaturesToFile(fileName, signatures);
20     }
21
22     public async void OnSignatureDeleted(IAnnotation signature)
23     {
24         var signatures = await ReadSignaturesFromFile(fileName);
25         var newSignatures = signatures.Where(s => s.Id != signature.Id).ToList();
26         await WriteSignaturesToFile(fileName, newSignatures);
27     }
28
29     public static async Task<ExternalJsonFileStorage> Create(string fileName)
30     {
31         // If the file already exists, we read it and pass the saved signatures as
32         var initialSignatures = await ReadSignaturesFromFile(fileName);
33         return new ExternalJsonFileStorage(fileName, initialSignatures);
34     }
35
36     private static async Task<StorageFile> TryReadingFile(string fileName)
37         => await ApplicationData.Current.LocalFolder.TryGetItemAsync(fileName) as
38
39     private static async Task<IList<IAnnotation>> ReadSignaturesFromFile(string
40     {
41         var file = await TryReadingFile(fileName);
42         return file is null
43             ? new List<IAnnotation>()
44             : JsonArray
45                 .Parse(await FileIO.ReadTextAsync(file))
46                 .Select(signatureJson => Factory.FromJson(signatureJson.GetObject()))
47                 .ToList();
48     }
49
50     private async Task WriteSignaturesToFile(string fileName, IList<IAnnotation>
51     {
52         // Convert the annotations to JSON.
53         var annotationsJson = new JsonArray();
54         foreach (var annotation in signatures)
55         {
56             annotationsJson.Add(annotation.ToJson());
57         }
58
59         var file = await ApplicationData.Current.LocalFolder
60             .CreateFileAsync(fileName, CreationCollisionOption.ReplaceExisting);
61
62         await FileIO.WriteTextAsync(file, annotationsJson.Stringify());

```

```
63     }
```

```
64 }
```

Was this helpful?

 YES

 NO

Questions? [Contact us](#)

