



UWP



Print



WINDOWS > GUIDES

PDF printing library in UWP

The print button from the main toolbar triggers an event you can subscribe to. You can then use

`PSPDFKit.PrintHelper` to present a print dialog to the user. Note that it isn't necessary to subscribe to the event in order to initiate printing; it's merely a convenience.

Toolbar print button

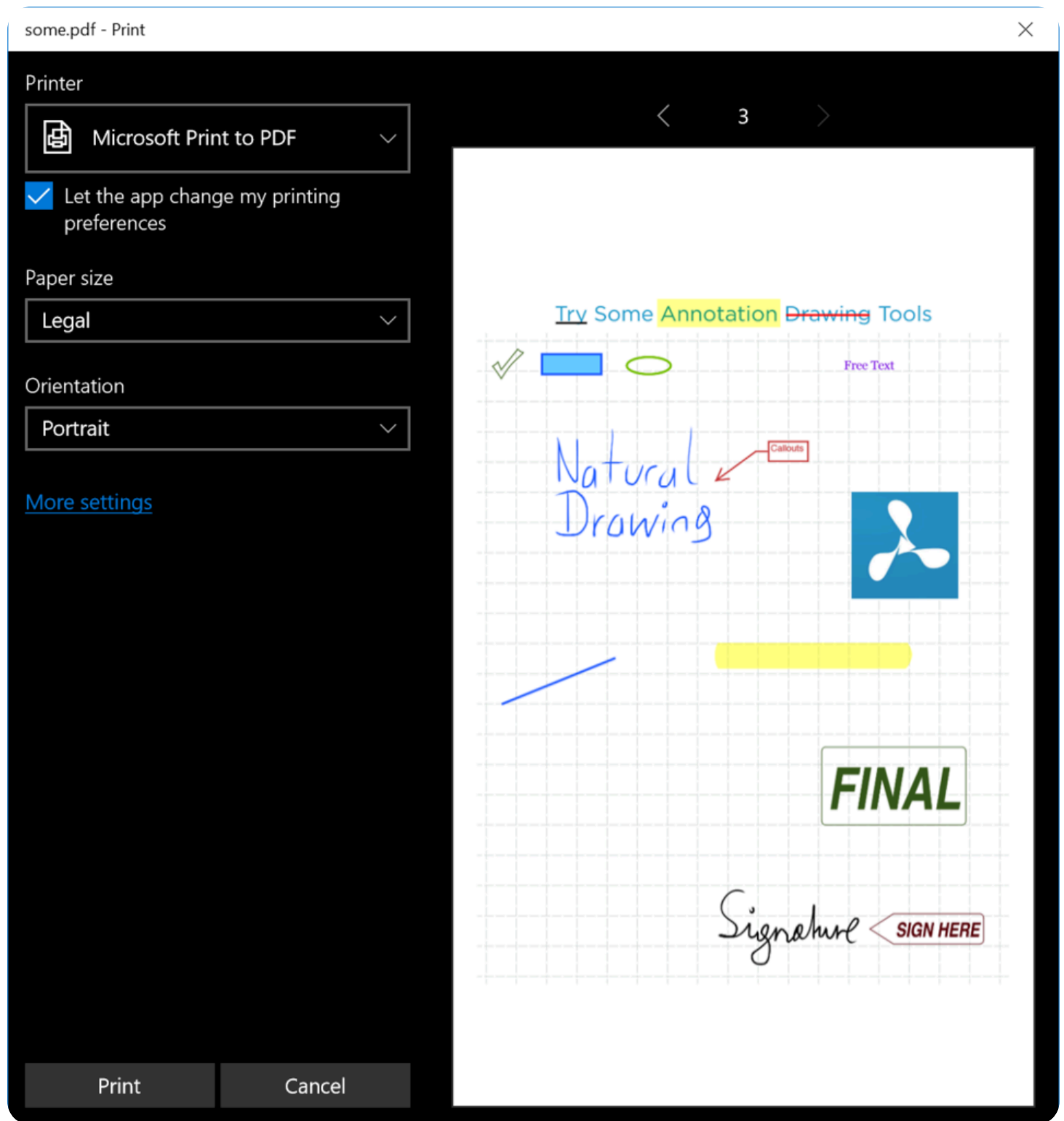
The following code from the Catalog app demonstrates how to subscribe to the print button click event in the toolbar. It's important to only subscribe once the `PSPDFKit.UI.PdfView` has completed initialization:

```
1 public PrintPage()  
2 {  
3     InitializeComponent();  
4     ViewModel.Initialize(PDFView);  
5  
6     PDFView.InitializationCompletedHandler += PDFView_InitializationCompletedHa  
7 }  
8  
9 // Once the API is initialized, we tell it we want to know when the user click  
10 private void PDFView_InitializationCompletedHandler(PSPDFKit.PDFView sender, i  
11 {  
12     sender.Controller.OnPrint += API_OnPrint;  
13 }  
14  
15 private void API_OnPrint(PSPDFKit.API sender, object args)  
16 {  
17     ViewModel.OnPrint(this);  
18 }
```



ASK AI

Windows print service dialog



To display the Windows printing service dialog, you'll need to place a named `Canvas` `UIElement` in your page's XAML:

```
<Canvas x:Name="PrintCanvas" Opacity="0"/>
```



To open the dialog, register an instance of the `PrintHelper` class with the `PSPDFKit.Pdf.Document`, the page you're printing from, and the name of the `Canvas` element.

To be informed of the status of the print job, add an event handler to

`PrintHelper.PrintingCompleteHandler`:

```
1 internal async void OnPrint(Windows.UI.Xaml.Controls.Page owningPage)
2 {
3     try
4     {
5         // For printing a `Canvas`, `UIElement` is required in the visual tree
6         var printHelper = new PrintHelper(PDFView.Document, owningPage, "Print")
7
8         printHelper.PrintingCompleteHandler += PrintHelper_PrintingCompleteHandler
9
10        await printHelper.ShowPrintUIAsync();
11    }
12    catch (Exception e)
13    {
14        var messageDialog = new MessageDialog(e.ToString());
15        await messageDialog.ShowAsync();
16    }
17 }
```

The print status tells you if the job was canceled or completed or something else:

```
1 private static async void PrintHelper_PrintingCompleteHandler(PrintHelper
2 {
3     await CoreApplication.MainView.CoreWindow.Dispatcher.RunAsync(CoreDispatch
4     {
5         var messageDialog = new MessageDialog("Print Status: " + args.Completi
6         await messageDialog.ShowAsync();
7     });
8 }
```

PrintHelper

The `PrintHelper` also exposes two notable properties: `PrintMediaSize` and `Watermark`.

Setting the default media size for the printing dialog can be done through `PrintMediaSize`, using Microsoft's `PrintMediaSize` enum.

Watermarks can be added during the printing process as follows:

```
1 var printHelper = await PrintHelper.CreatePrintHelperFromSourceAsync(docur
2
3 using (var image = await AssetLoader.GetWatermarkAsync())
4 {
5     printHelper.Watermark = new Watermark(image)
6     {
7         HorizontalAlignment = HorizontalAlignment.Center,
8         VerticalAlignment = VerticalAlignment.Center,
9         SizeOnPage = new Size(200, 200)
10    };
11 }
12
13 await printHelper.ShowPrintUIAsync();
```



For more information on watermarks, refer to our [watermarks](#) guide.

Was this helpful?

✓ YES

✗ NO

Questions? [Contact us](#)

