



UWP

Upgrade

WINDOWS > GUIDES

Nutrient UWP SDK 1.1 migration guide

This article provides a set of guidelines for migrating from version 1.0 to version 1.1 of Nutrient UWP SDK.

Class names and namespaces

To follow Microsoft guidelines, we renamed many of the classes, methods, and namespaces. Additionally, the organization of the namespaces has been improved to better reflect the purpose and relationship of the various parts of the SDK.

Here are the namespace and class name changes:

1.0	1.1
PSPDFKit.PDF	PSPDFKit.Pdf
PSPDFKit.PDFView	PSPDFKit.UI.PdfView
PSPDFKit.PDFViewMode	PSPDFKit.UI.PdfViewMode
PSPDFKit.PDFViewModeChange	PSPDFKit.UI.PdfViewModeChange
PSPDFKit.PDF.Actions.URI	PSPDFKit.Pdf.Actions.Uri
PSPDFKitSearch	PSPDFKit.Search
PSPDFKitSearch.Range	PSPDFKitFoundation.Search.Range



PdfView

The `PDFView`, now named `PdfView`, is located in the `PSPDFKit.UI` namespace. You must rename any references to the old name and namespace in your app's XAML.

Here's some typical XAML from `1.0`:

```
1 <Page xmlns:pspdfkit="using:PSPDFKit">  
2  
3     <pspdfkit:PDFView Name="myPdfView" License="{StaticResource license}" />  
4  
5 </Page>
```



And here is the equivalent for `1.1`:

```
1 <Page xmlns:ui="using:PSPDFKit.UI">  
2  
3     <ui:PdfView Name="myPdfView" License="{StaticResource license}" />  
4  
5 </Page>
```



API

The `API` class has been deprecated, and in its place, there are now several new classes.

Pdf.Document

`PSPDFKit.Pdf.Document` represents a PDF document. You can get the currently open document from the property `PdfView.Document`. The `Document` class contains all the methods for working directly with a PDF document. Here are just some of the facilities provided by this class:

- ❖ Creating, updating, and modifying annotations
- ❖ Event handlers for annotation events
- ❖ Information about the PDF, such as the number of pages
- ❖ Exporting the document to a `StorageFile` or `DataWriter`

DocumentSource

`Document` has a `DocumentSource` property, which is the type `PSPDFKit.Document.DocumentSource`. This class represents the source of the document.

A `DocumentSource` can be created with its static methods `CreateFromStorageFile` or `CreateFromBuffer`:

```
1 var documentSource = DocumentSource.CreateFromStorageFile(aStorageFile);  
2 await myPdfView.Controller.ShowDocumentAsync(documentSource);
```



UI controller

Methods and properties for querying and changing the UI can now be found in the `PSPDFKit.UI.Controller` class. The `PdfView` class has a property, `Controller`, exposing this class.

With the `Controller`, you can perform actions such as opening a PDF (as shown in the previous code example), changing the currently displayed page, or subscribing to UI events:

```
1 // Get notified if the currently displayed page changes to another page.  
2 myPdfView.Controller.OnCurrentPageChanged += OnHandlePageChanged;  
3  
4 // Change to page 2.  
5 await myPdfView.Controller.SetCurrentPageIndexAsync(2);
```



Search

Search facilities are no longer contained in a separate UWP component and now reside in the `PSPDFKit` component under the `PSPDFKit.Search` namespace. The search classes and methods have also been redesigned.

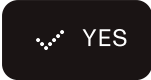
To search, use the method `TextSearcher.SearchDocument`, passing it a `Document` and a `Query`:

```
1 var myTextSearcher = new TextSearcher();  
2 myTextSearcher += OnSearchResult;  
3 myTextSearcher.SearchDocument(myPdfView, new Query("waldo"));
```



See the source code of the Catalog app included with the SDK for a complete example of searching and many other facilities.

Was this helpful?



Questions? [Contact us](#)

