



UWP



Choose a Page



WINDOWS > GUIDES

NavigationPageCache support and limitations

`Page` can be cached in the navigation cache for faster loading of pages by setting the `NavigationCacheMode` attribute to `Required`. This works well for pages where the content is easily cached and there are no complications regarding the acquisition or release of limited resources associated with the `Page` or any of its `UIElements`. For example, a simple dial control or a grid control showing a small amount of data will benefit from this.

The `PdfView`, however, has to manage several system-critical resources, such as significant memory allocation and file handles. Unfortunately, the XAML UI framework doesn't supply enough event information for the control to safely hold on to those resources when a `Page` is unloaded from the visual tree, because there's no event signaling that the `Page` has been ejected from the navigation cache, and we cannot let these resources remain acquired until some undetermined time in the future, possibly until the app exits. Even then, there's no way to clean up or persist changes properly.

Because of this problem with the Microsoft UI API design, **we do not recommend using the navigation page cache** with pages containing a `PdfView`. However, if you must use it for whatever reason, we do support the setting, but with the following restrictions and considerations:

- ❖ When the `Page` is removed from the visual tree, the `PdfView` receives an `Unloaded` event and will unload the document from the `PdfView`.
- ❖ When a `Page` is remounted into the visual tree, the `PdfView` needs to initialize again, and the `InitializationCompletedHandler` handler will be invoked when it's ready.
- ❖ If a PDF is loaded into the `PdfView` with the `PdfUriSource` or `PdfFileSource`, it'll be lost if the `Page` is remounted into the visual tree.



ASK AI

✧ If a PDF is loaded into the PdfView via the ShowDocumentAsync method, it won't be reloaded if the Page is remounted into the visual tree. This is because of the aforementioned issue with tracking resources. In this case, you should reload the document in your InitializationCompletedHandler handler.

As you can see, using the navigation page cache with controls of large complexity will add more complexity to your application. As such, we recommend you use a different design until Microsoft provides a method of complex controls for developers to properly manage limited resources.

We've raised this issue with Microsoft, as well as a related one with x:DeferLoadStrategy changing the order of loading events, but we haven't received any advice that solves the primary issue.

- ✧ Determining when a control can release resources when NavigationCacheMode is required
- ✧ Is it expected that x:DeferLoadStrategy changes the order of loading?

Was this helpful?

☒ YES

☐ NO

Questions? [Contact us](#)

