



UWP



Events and notifications



WINDOWS &gt; GUIDES

# Events and notifications

Nutrient UWP SDK exposes specific events for applications to consume. These events are mainly related to user actions. They're raised after key UI interactions and data manipulation, which allows users more control over their `PdfView` S.

After a `PdfView` is instantiated, subscribing to events is straightforward, as one would expect in C#:

```
PDFView.InitializationCompletedHandler += PDFViewOnInitializationCompletedHa
```



```
1 private void PDFViewOnInitializationCompletedHandler(PdfView sender, Docur
2 {
3     // ...
4 }
```



In addition to their specific arguments, events also include the sender object in their event handlers.

## Structure

Events are found in four major classes:



Controller



PdfView



Document



ASK AI



Library

The primary source for events is the `Controller` class, which should be where you look first. A number of events related to the view and its state are also found in the `PdfView` class — most notably, the `InitializationCompletedHandler` event shown above.

Events related to bookmarks and annotations that aren't necessarily tied to the UI are part of the `Document` class, while search events are found in the `Library` class. More details about Nutrient UWP SDK's capabilities for indexed search can be found [here](#).

Finally, custom `ToolbarItems` expose their own press events. You can read more about customizing the toolbar [here](#).

For working code samples, the Catalog example app has a section for events. Moreover, a detailed view of each event can be found in the documentation for its respective parent class.

## Annotations

[Annotations](#) have several events associated with them, mainly regarding user interaction, with the `IAnnotation` object interacted with being passed through in the event arguments. Considering annotations have unique IDs, and given your application subscribes to the relevant events, managing them and adapting interactions to specific use cases is simple.

The following annotation and annotation-related events are included in Nutrient UWP SDK:



`AnnotationsCreated`



`AnnotationsDeleted`



`AnnotationsUpdated`



`OnAnnotationPresetUpdate`



`OnAnnotationSelectionChanged`



`OnAnnotationPressed`



`IsEditableAnnotation`



`OnAnnotationsWillChange`

The `OnAnnotationPressed` event has been exposed in the `Controller` class, and it handles user presses, including information about the pointer used:

```

1  {
2      PDFView.Controller.OnAnnotationPressed += OnAnnotationPressed;
3  }
4
5  private void OnAnnotationPressed(Controller controller, AnnotationPressedEventI
6  {
7      if (args.Pointer.IsPrimary)
8      {
9          Debug.Write(args.Annotation.AnnotationType);

```

Preventing the default behavior of an annotation is also possible through the same event. Note that this isn't supported inside `async` event handlers:

```

1  private void OnAnnotationPressed(Controller controller, AnnotationPressedI
2  {
3      if (args.Annotation.AnnotationType == AnnotationType.Ink)
4      {
5          args.PreventDefault = true;
6      }
7  }

```

Detecting when annotations will change can be done with the aptly named `OnAnnotationsWillChange` event, which is also found in the `Controller`. Before user actions cause annotation changes, this event is invoked, and it states the reason and lists the affected annotations:

```

1  {
2      PDFView.Controller.OnAnnotationsWillChange += OnAnnotationsWillChange;
3  }
4
5  private void OnAnnotationsWillChange(Controller controller, AnnotationsChanging
6  {
7      if (args.Reason == AnnotationsWillChangeReason.DrawStart)
8      {
9          for (var i = 0; i < annotationsChanging.Annotations.Count; i++)
10         {
11             Debug.Write(annotationsChanging.Annotations[i].Id)
12         }
13     }
14 }

```

A more in-depth introduction to annotations and their uses can be found [here](#).

# Bookmarks

Reacting to bookmark actions can be done through the `BookmarksCreated`, `BookmarksDeleted`, and `BookmarksUpdated` events found in the `Document` class. These can be used to add custom logic and manually manage bookmarks:

```
1 document.BookmarksCreated += (view, bookmarks) =>
2 {
3     foreach (var bookmark in bookmarks)
4     {
5         _bookmarks[bookmark.Id] = bookmark;
6     }
7 };
```

Additional examples can be found on the events page of the Catalog, and [this page](#) presents an overview of bookmarks.

## User interaction

The `OnPagePressed` event found in the `Controller` class both informs page-press coordinates and holds information about the pointer used. This allows for better multi-pointer support and specific behavior depending on, for example, the pointer type:

```
1 pdfView.Controller.OnPagePressed += (sender, args) =>
2 {
3     if (args.PageIndex == 3 && args.Pointer.PointerId == 1)
4     {
5         // ...
6     }
7 };
```

Another useful event regarding user interaction is `OnTextSelectionChanged`. More information about it can be found [here](#).

## Indexed full-text search

The full-text search library exposes a few different events, which gives your application a lot of control over the process:

- ❖ OnFinishedIndexingDocument
- ❖ OnPageIndexed
- ❖ OnPasswordRequested
- ❖ OnSearchComplete
- ❖ OnSearchPreviewComplete
- ❖ OnSearchTimedOut
- ❖ OnStartIndexingDocument

Examples of these events are detailed on the Full-Text-Search page of the Catalog project and in the [indexed full-text search guide](#).

---

Was this helpful?

☒ YES

☐ NO

Questions? [Contact us](#)

