



UWP



Redaction



WINDOWS > GUIDES > REDACTION

Redacting PDFs programmatically in Windows

You can create redaction annotations programmatically via the `Redaction` annotation. Use the `Rects` property to set the regions that should be covered by the redaction annotation. Additionally, the `BoundingBox` needs to be set to a `Windows.Foundation.Rect` containing all the specified `Rects`.

You also have a few customization options for how a redaction should look, both while the redaction annotation is in its marked state, which is when it has been created but not yet applied, and in its redacted state, which is when the content is effectively redacted:

- ❖ `OverlayText` can be used to set the text that should be displayed in a specified region when a redaction has been applied.
- ❖ `RepeatOverlayText` defines whether the overlay text should be drawn once or repeated to fill the entire redaction area. This defaults to `false`, which means the overlay text is only drawn once. It has no effect if there's no overlay text specified.
- ❖ `Color` can be used to change the color of the overlay text. It has no effect if there's no overlay text specified. This defaults to `Windows.UI.Colors.Red`.
- ❖ `FillColor` specifies the background color of the redaction area after it has been applied. The color is drawn on all the specified `Rects`. This defaults to `Windows.UI.Colors.Black`.
- ❖ `OutlineColor` specifies the color used for the redaction's border in its marked state. This defaults to `Windows.UI.Colors.Red`.

It's not possible to change the appearance once a redaction has been applied, since the redaction annotation will be removed from the document and the redaction will be part of the content of the document. This is an action that irreversibly replaces the original content under the specified



ASK AI



```
1 var boundingBox = new Rect(25, 25, 175, 30);
2
3 var redaction = new Redaction
4 {
5     PageIndex = 0,
6     BoundingBox = boundingBox,
7     Rects = new List<Rect> { boundingBox },
8     OverlayText = "REDACTED",
9     Color = Colors.Orange
10 };
11
12 await PDFView.Document.CreateAnnotationAsync(redaction);
```

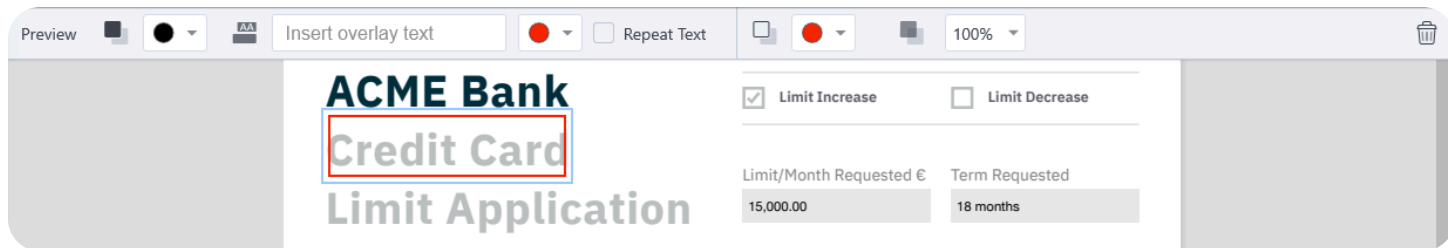
The redaction annotation created with the above code snippet would look like what's shown in the image below.

MARKED STATE



Redaction properties

Once a redaction is added to a document, its properties and appearance can be customized via the annotation toolbar.



The properties under the Preview label affect the redacted appearance (i.e. how the document will look once redactions are applied), while the outline color picker only affects the marked appearance of the annotation.

Previewing redactions

To preview redactions and see how they'd look when applied without removing any document content, you can use the `PDFView.PreviewRedactionsAsync()` method, passing in `true` as its parameter:

```
await PDFView.Document.PreviewRedactionsAsync(true);
```



Applying redactions

To redact the document after all the redaction annotations are added, you can call the `Pdf.Document.ApplyRedactionsAsync()` API method. This will overwrite the existing document, removing content irreversibly:

```
await PDFView.Document.ApplyRedactionsAsync();
```



The redaction annotations will be removed once the document has been redacted and the affected content has been removed. Any other existing annotations within the area of a redaction annotation are also removed.

Was this helpful?

✓ YES

✗ NO

