How to programmatically annotate PDF documents

Written by Apitron Documentation Team

Introduction

PDF supports various kinds of annotations which can be used to markup or complement the contents of the PDF document. Many of them are interactive, so you may select, alter and reposition them as you wish.

Examples are:

- Text annotation, representing a “sticky note” attached to a point in the PDF document. When closed, the annotation appears as an icon; when open, it displays a pop-up window containing the text of the note in a font and size chosen by the conforming reader.
- Watermark annotation, used to indicate the state or other properties of the document. These annotations are discussed in this post.
- Polyline and polygon annotations which can be used to outline parts of the page content.
- Link annotations (thoroughly discussed in this post).

There are twenty annotations types described in PDF specification, see the section *12.5.6 Annotation Types.*

In this post we’ll show how to create and use some of them, leaving others as an exercise for inquisitive readers.

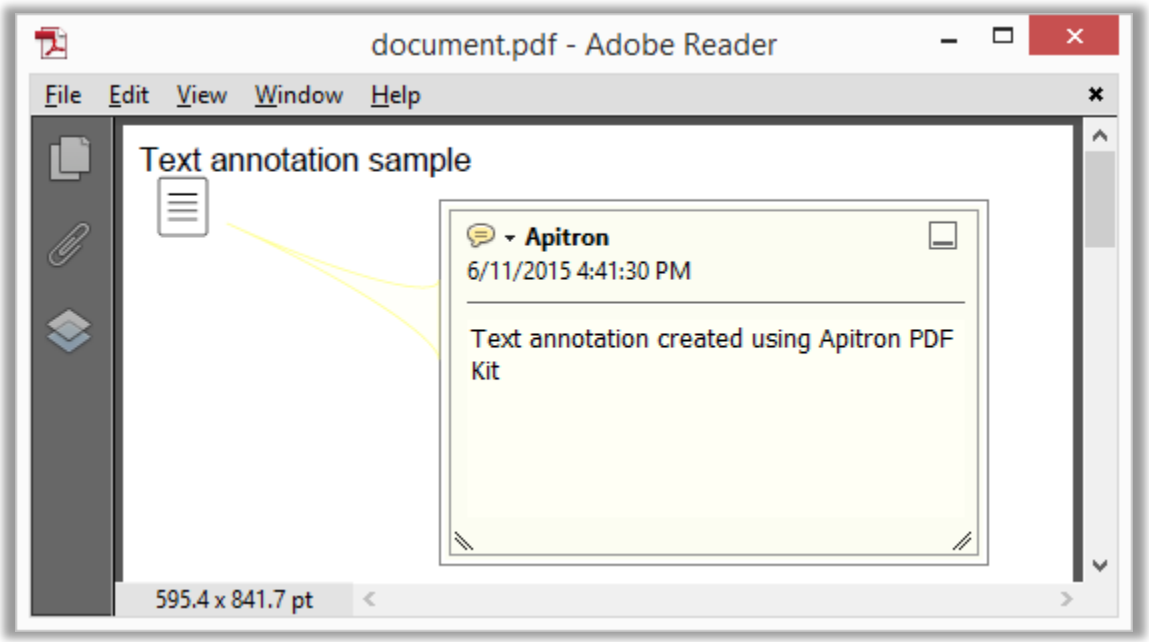
Working with PDF annotations

Text Annotations

Let’s create a simple sticky note on PDF page. See the code below:

```
public void CreateTextAnnotation()
{
    // open document
    using (Stream  stream = new FileStream("document.pdf", FileMode.Open, FileAccess.ReadWrite))
    {
        // create document object
        FixedDocument doc = new FixedDocument(stream);

        // create text annotation
        TextAnnotation textAnnotation = new TextAnnotation(20,790);
        textAnnotation.Title = "Apitron";
        textAnnotation.Contents = "Text annotation created using Apitron PDF Kit";
        textAnnotation.IsOpen = true;

        // add annotation to page's annotation collection
        doc.Pages[0].Annotations.Add(textAnnotation);

        // save as incremental update
        doc.Save();
    }
}
```

The resulting document looks as follows:



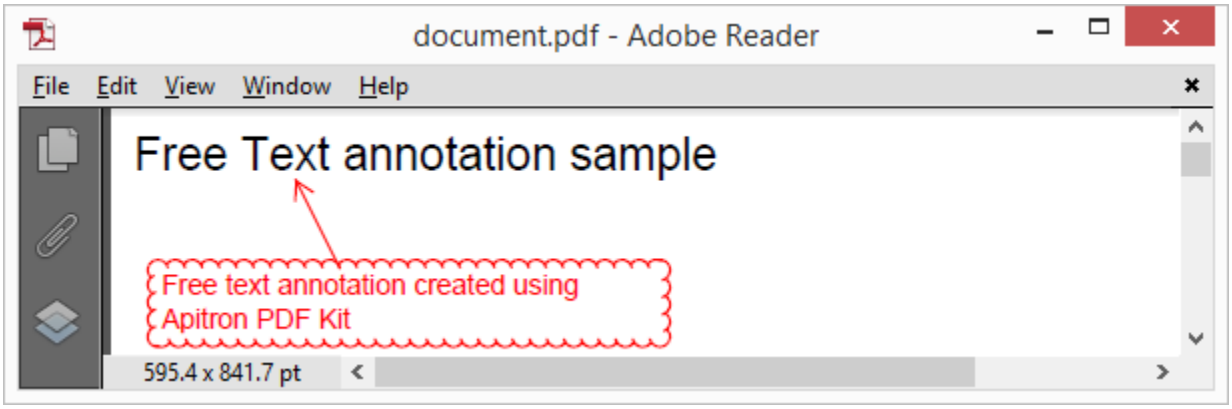
Pic. 1 Text annotation on PDF page

Free Text Annotations

This type of annotation has no open or closed state like regular text annotation - it displays the text directly on page. It can be created using three available views: FreeText, FreeTextCallout, and FreeTextTypeWriter. The first shows plain text, second works as callout, and the last one shows text and allows typing. We'll use the callout subtype in our sample, see the code below.

```
public static void CreateFreeTextAnnotation()
{
    // open document
    using (Stream stream = new FileStream("document.pdf", FileMode.Open, FileAccess.ReadWrite))
    {
        // create document object
        FixedDocument doc = new FixedDocument(stream);

        // create text annotation
        FreeTextAnnotation annotation = new FreeTextAnnotation(new Boundary(20,740,240,770));
        annotation.Title = "Apitron";
        annotation.Contents = "Free text annotation created using Apitron PDF Kit";
        // indicate the callout subtype
        annotation.Intent = IntentStyle.FreeTextCallout;
        // set callout line properties
        annotation.Callout = new double[] { 80, 810, 100, 770 };
        annotation.LineEndingStyle = AnnotationLineEndingStyle.OpenArrow;
        annotation.TextColor = RgbColors.Red.Components;
        annotation.FontSize = 12;
        // define the border effect
        annotation.BorderEffect = new AnnotationBorderEffect(AnnotationBorderEffectStyle.Cloudy, 1);

        // add annotation to page's annotation collection
        doc.Pages[0].Annotations.Add(annotation);

        // save as incremental update
        doc.Save();
    }
}
```

See the results below:



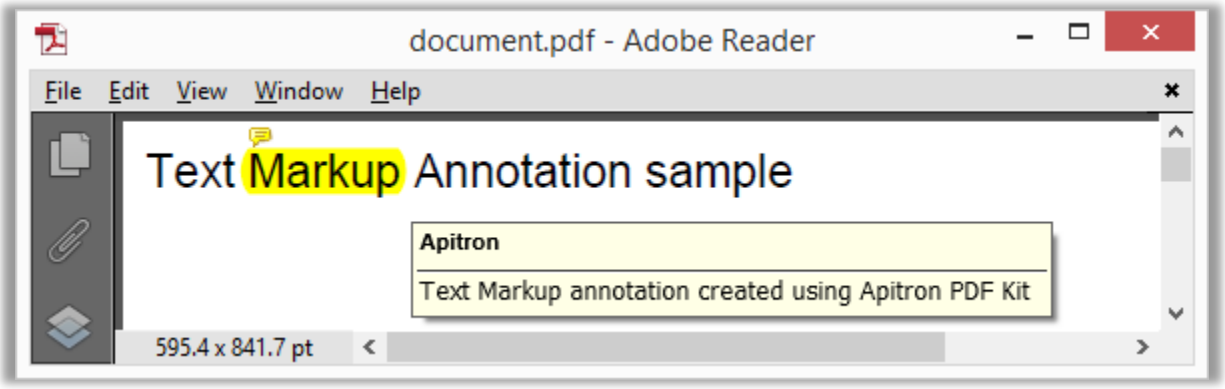
Pic. 2 Free Text PDF annotation

Text Markup Annotations

These annotations appear as highlights, underlines, strikeouts or jagged (“squiggly”) underlines in the text of a document. When opened, they display a popup window containing the text of the linked note. See the code below:

```
public void CreateTextMarkupAnnotation()
{
    // open document
    using (Stream stream = new FileStream("document.pdf", FileMode.Open, FileAccess.ReadWrite))
    {
        // create document object
        FixedDocument doc = new FixedDocument(stream);

        // create "highlight" text markup annotation
        TextMarkupAnnotation annotation = new TextMarkupAnnotation(new Boundary(55,810,120,830));
        annotation.Title = "Apitron";
        annotation.Contents = "Text Markup annotation created using Apitron PDF Kit";
        annotation.MarkupType = TextMarkupType.Highlight;
        annotation.Color = new double[]{1,1,0};

        // add annotation to page's annotation collection
        doc.Pages[0].Annotations.Add(annotation);

        // save as incremental update
        doc.Save();
    }
}
```

The resulting document looks as follows:



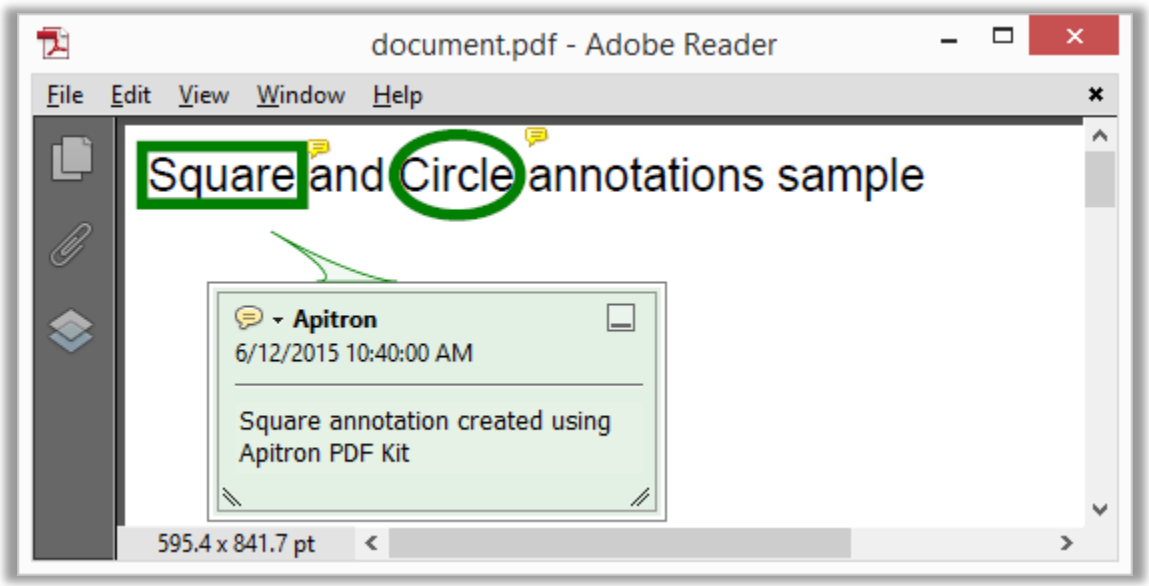
Pic. 3 Text Markup annotation added to PDF page

Square and Circle Annotations

These annotations represent squares or circles on PDF page. When opened, they display a pop-up window containing the text of the assigned note. Consider the code below:

```
public void CreateSquareAndCircleAnnotations()
{
    // open document
    using (Stream stream = new FileStream("document.pdf", FileMode.Open, FileAccess.ReadWrite))
    {
        // create document object
        FixedDocument doc = new FixedDocument(stream);

        // create square annotation
        SquareAnnotation square = new SquareAnnotation(new Boundary(5, 805, 80, 835),
          AnnotationFlags.Default, new AnnotationBorderStyle(5));
        square.Title = "Apitron";
        square.Contents = "Square annotation created using Apitron PDF Kit";

        // create circle annotation
        CircleAnnotation circle = new CircleAnnotation(new Boundary(115, 800, 175, 840),
          AnnotationFlags.Default, new AnnotationBorderStyle(5));
        circle.Title = "Apitron";
        circle.Contents = "Circle annotation created using Apitron PDF Kit";

        square.Color = circle.Color = new double[] {0, 0.5, 0};
        // add annotations to page's annotation collection
        doc.Pages[0].Annotations.Add(square);
        doc.Pages[0].Annotations.Add(circle);

        // save as incremental update
        doc.Save();
    }
}
```

The resulting document is shown below:



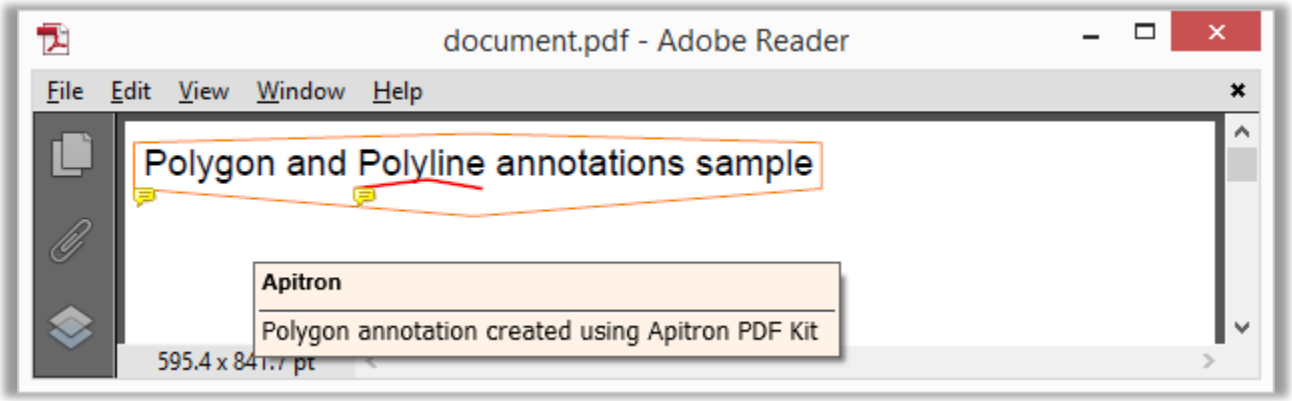
Pic. 4 Square and circle annotations added to PDF page

Polygon and Polyline Annotations

Polygon annotations display closed polygons on the PDF page. Such polygons may have any number of vertices connected by straight lines. Polyline annotations are similar to polygons, except that their first and last vertices are not implicitly connected. See the code below:

```
public void CreatePolygonAndPolylineAnnotations()
{
    // open document
    using (Stream stream = new FileStream("document.pdf", FileMode.Open, FileAccess.ReadWrite))
    {
        // create document object
        FixedDocument doc = new FixedDocument(stream);

        // create polygon annotation
        PolygonAnnotation polygon = new PolygonAnnotation(new Boundary(5, 805, 380, 835));
        polygon.Title = "Apitron";
        polygon.Contents = "Polygon annotation created using Apitron PDF Kit";
        polygon.Intent = IntentStyle.PolygonDimension;
        polygon.Vertices = new[] { 5.0, 805, 190, 790, 380, 805, 380, 830, 190, 835, 5, 830, 5, 805 };
        polygon.Color = new double[] { 1, 0.5, 0 };

        // create polyline annotation
        PolylineAnnotation polyline = new PolylineAnnotation(new Boundary(115, 800, 175, 840),
          AnnotationFlags.Default, new AnnotationBorderStyle(2));
        polyline.Title = "Apitron";
        polyline.Contents = "Polyline annotation created using Apitron PDF Kit";
        polyline.Vertices = new[] {125.0, 805, 165, 810, 195, 805};
        polyline.Color = RgbColors.Red.Components;

        // add annotations to page's annotation collection
        doc.Pages[0].Annotations.Add(polygon);
        doc.Pages[0].Annotations.Add(polyline);

        // save as incremental update
        doc.Save();
    }
}
```

This code produces the following results:



Pic. 5 Polygon and polyline annotations added to PDF page

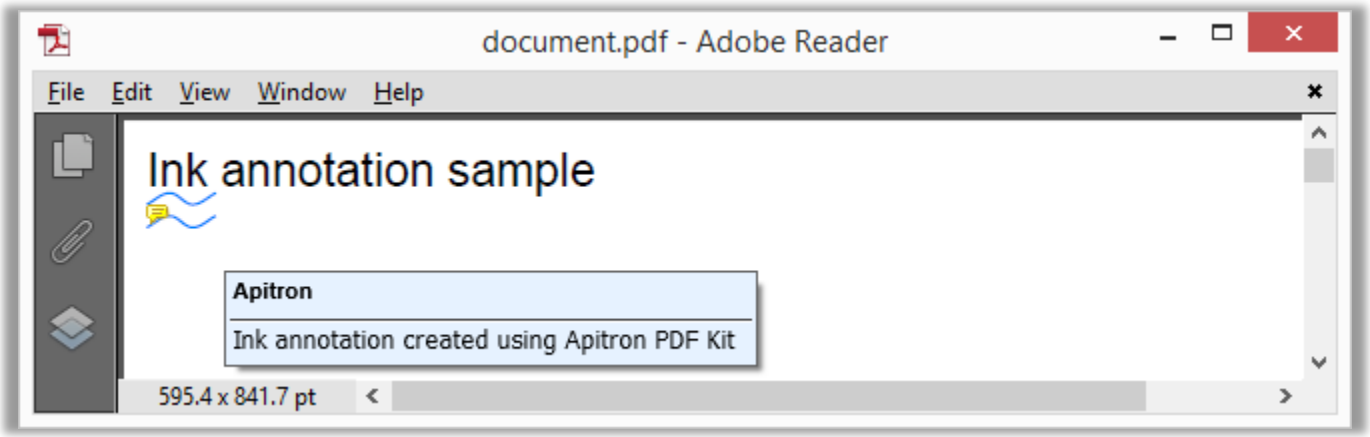
Ink Annotations

Ink annotation represents a freehand drawing composed of one or more, possibly disconnected, graphic paths. These paths are stored in `InkList` paths collection, and when drawn, the points become connected by straight lines or curves in an implementation-specific way. When opened, the annotation displays a pop-up window containing the text of the linked note.

See the code below:

```
public void CreateInkAnotation()
{
    // open document
    using (Stream stream = new FileStream("document.pdf", FileMode.Open, FileAccess.ReadWrite))
    {
        // create document object
        FixedDocument doc = new FixedDocument(stream);

        // create ink annotation
        InkAnnotation annotation = new InkAnnotation(new Boundary(5, 805, 380, 835));
        annotation.Title = "Apitron";
        annotation.Contents = "Ink annotation created using Apitron PDF Kit";
        // add paths
        annotation.InkList.Add(new []{10.0,805,20,810,30,805,40,810});
        annotation.InkList.Add(new []{10.0,795,20,800,30,795,40,800 });
        annotation.Color = new [] { 0, 0.5, 1 };

        // add annotations to page's annotation collection
        doc.Pages[0].Annotations.Add(annotation);

        // save as incremental update
        doc.Save();
    }
}
```

Resulting document looks as follows:



Pic. 5 Ink annotation added to PDF page

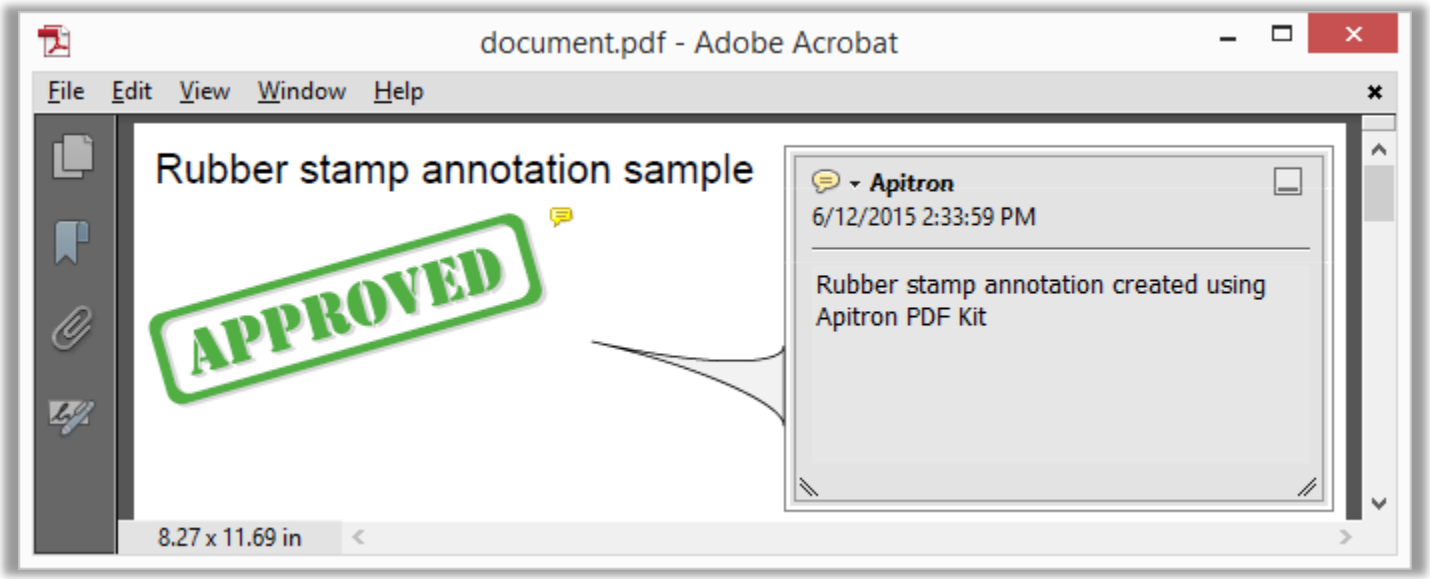
Rubber Stamp Annotations

A rubber stamp annotation displays text or graphics intended to look as if it was stamped on the page with a rubber stamp. When opened, it displays a pop-up window containing the text of the assigned note.

Let's create the stamp based on a standard icon:

```
public void CreateRubberStampAnnotation()
{
    // open document
    using (Stream stream = new FileStream("document.pdf", FileMode.Open, FileAccess.ReadWrite))
    {
        // create document object
        FixedDocument doc = new FixedDocument(stream);

        // create rubber stamp annotation
        RubberStampAnnotation stamp = new RubberStampAnnotation(new Boundary(5, 700, 205, 800));
        stamp.Title = "Apitron";
        stamp.Contents = "Rubber stamp annotation created using Apitron PDF Kit";
        stamp.Icon = AnnotationIconNames.Approved;
        stamp.Rotate = 15;

        // add annotations to page's annotation collection
        doc.Pages[0].Annotations.Add(stamp);

        // save as incremental update
        doc.Save();
    }
}
```

Resulting stamped document looks as follows:

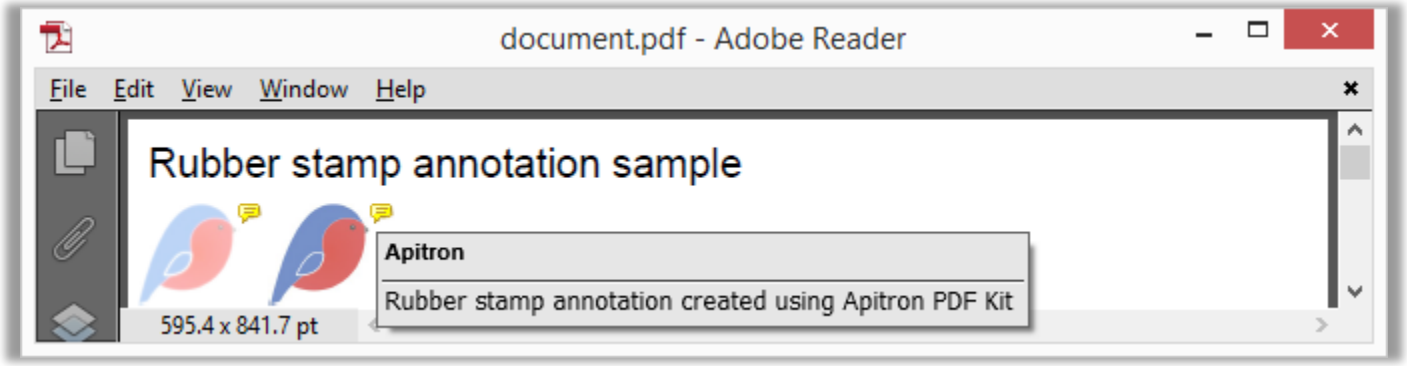


Pic. 6 Standard Rubber Stamp annotation added on PDF page

If you'd like to use a custom stamp instead, you could do it using the code below:

```
public void CreateCustomRubberStampAnnotation()
{
    // open document
    using (Stream stream = new FileStream("document.pdf", FileMode.Open, FileAccess.ReadWrite))
    {
        // create document object
        FixedDocument doc = new FixedDocument(stream);
        // register image resources
        doc.ResourceManager.RegisterResource(new Image("stamp","myimage.jpg"));
        doc.ResourceManager.RegisterResource(new Image("stamp2","myimage2.jpg"));

        // create rubber stamp annotation with normal and rollover states
        RubberStampAnnotation annotation = new RubberStampAnnotation(new Boundary(5, 750, 55, 800));
        annotation.Title = "Apitron";
        annotation.Contents = "Rubber stamp annotation created using Apitron PDF Kit";
        annotation.Appearance.Normal = new FixedContent("stampNormal",new Boundary(5, 750, 55, 800));
        annotation.Appearance.Normal.Content.AppendImage("stamp2",5,750,50,50);
        annotation.Appearance.Rollover = new FixedContent("stampRoll", new Boundary(5, 750, 55, 800));
        annotation.Appearance.Rollover.Content.AppendImage("stamp", 5, 750, 50, 50);

        // create rubber stamp annotation with normal and rollover states
        RubberStampAnnotation annotation2 = new RubberStampAnnotation(new Boundary(70, 750, 120, 800));
        annotation2.Title = "Apitron";
        annotation2.Contents = "Rubber stamp annotation created using Apitron PDF Kit";
        annotation2.Appearance.Normal = new FixedContent("stampNormal2",
          new Boundary(70, 750, 120, 800));
        annotation2.Appearance.Normal.Content.AppendImage("stamp2", 70, 750, 50, 50);
        annotation2.Appearance.Rollover = new FixedContent("stampRoll2",
          new Boundary(70, 750, 120, 800));
        annotation2.Appearance.Rollover.Content.AppendImage("stamp", 70, 750, 50, 50);

        // add annotations to page's annotation collection
        doc.Pages[0].Annotations.Add(annotation);
        doc.Pages[0].Annotations.Add(annotation2);

        // save as incremental update
        doc.Save();
    }
}
```

It adds two custom rubber stamps with normal and rollover states using different images. The washy image is used for normal state, the bright for rollover. See the image below:



Pic. 7 Rubber stamp annotations added to PDF page

Conclusion

In this post, we have shown how to work with most commonly used PDF annotations using Apitron PDF Kit .NET component. It offers the easy to use and clear API which you can employ to create PDF processing applications serving your needs. Available for many modern desktop, web, cloud, and mobile platforms: Windows, Windows Store and Windows Phone, Android and iOS (via Xamarin), OS X (MONO) – it makes any PDF manipulation scenario possible. Read our free book (link) or contact us if you have any questions related to PDF and our products.