Apitron PDF Kit and Apitron PDF Rasterizer - engine settings explained

Written by Apitron Documentation Team

Introduction

Apitron PDF Kit for .NET and Apitron PDF Rasterizer for .NET both share a common approach for fine tuning their internal engines PDF processing engines. One may use `EngineSettings` class for that and control the memory usage, as well as font substitution and fallback. In addition the Apitron PDF Rasterizer has a `RenderingSettings` class that controls the rendering. In this article we'll explain how to use `EngineSettings` and what each setting means.

The code

Example below shows how to use global engine settings for both Apitron PDF Kit and Apitron PDF Rasterizer, and also rendering settings. The piece of code related to rasterizer demonstrates the usage of *per document* engine settings as well.

```
static void Main(string[] args)
{
    /* GLOBAL SETTINGS usage, same for Apitron PDF Kit  and Apitron PDF Rasterizer */

    // controls whether we have to unload resources whenever possible based on size limit setting,
    // if the resource takes more than allowed it will be unloaded.
    EngineSettings.GlobalSettings.MemoryAllocationMode = MemoryAllocationMode.ResourcesLowMemory;
    // sets the resource size limit
    EngineSettings.GlobalSettings.ResourceSizeLimit = 1048676;

    // system font paths used to find external fonts [readonly].
    ICollection<string> systemFontPaths = EngineSettings.SystemFontPaths;

    // a collection used to specifiy additional font search paths
    ICollection<string> userFontPaths = EngineSettings.UserFontPaths;
    // example:
    userFontPaths.Add(@"c:\\myfonts");

    // set font fallbacks
    // map Arial and Calibri to Helvetica if they are not embedded in document
    // and not found in system and user font folders
    EngineSettings.UserFontMappings.Add(new KeyValuePair<string, string[]>("Helvetica",
       new string[]{"Arial","Calibri"}));
    // map all not found fonts to TimesNewRoman using special name "*"
    EngineSettings.UserFontMappings.Add(new KeyValuePair<string, string[]>("TimesNewRoman",
       new string[]{"*"}));

    // registers additional font in library's font cache if don't have a user font folder
    // or can't create one. Font name and parameters will be read from font file.
    using (Stream fontStream = File.Open("c:\\fonts\\Consolas.ttf", FileMode.Open))
    {
        EngineSettings.RegisterUserFont(fontStream);
    }

    // unregister all fonts registered via RegisterUserFonts()
    EngineSettings.UnregisterUserFonts();

    // create document
    using (Stream outputDocument = File.Create("document.pdf"))
    {
        FixedDocument document = new FixedDocument();
        document.Pages.Add(new Page());
        document.Save(outputDocument);
    }

    /* Apitron PDF Rasterizer only usage*/
    using (Stream inputDocument = File.Open("document.pdf", FileMode.Open))
    {
        // create engine settings instance and set memory usage limit
        Apitron.PDF.Rasterizer.Configuration.EngineSettings settings =
           new Apitron.PDF.Rasterizer.Configuration.EngineSettings();
        settings.MemoryAllocationMode =
           Apitron.PDF.Rasterizer.Configuration.MemoryAllocationMode.ResourcesLowMemory;
        settings.ResourceSizeLimit = 2000000;
```

```
            // open PDF document using specific engine settings,
            // these settings will be applied to this document only
            using (Document doc = new Document(inputDocument, settings))
            {
                // create rendering settings instance
                RenderingSettings renderingSettings = new RenderingSettings();
                // turn off annotations drawing for example
                renderingSettings.DrawAnotations = false;

                // render page
                Bitmap bitmap = doc.Pages[0].Render(new Resolution(96, 96), renderingSettings);
                bitmap.Save("page0.png");
            }
        }
    }
```

Using engine settings you can: control memory usage; define font substitution and fallback behavior, register own fonts. Using rendering settings you can: control which parts of content should be drawn or not, control rendering speed and quality as well as scale mode used.

The complete code sample can be found in our github repo.

Conclusion

Apitron PDF products are highly customizable and can be set up to work under many different conditions. If you have any questions or need help, just drop us an email and we'll be happy to consult you.