Windows 10 universal application pdf viewer control sample

Written by Apitron Documentation Team

Introduction

Rendering and showing PDF documents in Windows 10 Universal applications can be a tricky task and that’s why we prepared a PDF Viewer sample based on the Apitron PDF Rasterizer .NET component that works on many platforms including Windows 10 desktop and mobile.

This viewer control is implemented as the `UserControl` and corresponding `ViewModel` that manages PDF document loading and provides properties consumed by the owner control via data binding. You can use this control as a base for you own viewer or host it as is if it satisfies your needs. Implementation details are discussed in the next sections.

The code

XAML:

```
<UserControl
    x:Class="UniversalAppSamplePDFViewerControlForWindows10.Controls.PDFViewerControl"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:converters="using:UniversalAppSamplePDFViewerControlForWindows10.Controls.Converters"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400" Name="pdfViewerControl">
    <UserControl.Resources>
        <ResourceDictionary>
            <converters:DoubleToPercentageConverter x:Name="DoubleToPercentageConverter"/>
            <converters:PageIndexToPageNumberConverter x:Name="PageIndexToPageNumberConverter"/>
            <Style TargetType="Button">
              …custom style definition for viewer buttons
            </Style>
        </ResourceDictionary>
    </UserControl.Resources>
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="60"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
        <StackPanel Orientation="Horizontal" Grid.Row="0" HorizontalAlignment="Center">
            …toolbar buttons and labels are defined here
        </StackPanel>
        <ScrollViewer Grid.Row="1" Background="{Binding ElementName=pdfViewerControl, Path=Background}"
            HorizontalScrollBarVisibility="Auto" VerticalScrollBarVisibility="Auto" Name="scrollViewer">
            <Image x:Name="myImage" Stretch="None"/>
        </ScrollViewer>
    </Grid>
</UserControl>
```

You can see that the control is pretty simple and its main parts are toolbar, and a scrollview with an image inside it representing a rendered page. The control binds to a viewmodel represented by the `PDFViewerControlViewModel` class. The model gets set in the constructor of the control like this:

```
public PDFViewerControl()
{
    this.InitializeComponent();

    viewModel = new PDFViewerControlViewModel();
    viewModel.Navigated += NavigationHandler;
    this.DataContext = viewModel;
}
```

And the main page of the app looks as follows:

```
<Page
    x:Class="UniversalAppSamplePDFViewerControlForWindows10.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:UniversalAppSamplePDFViewerControlForWindows10"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:controls="using:UniversalAppSamplePDFViewerControlForWindows10.Controls"
    mc:Ignorable="d">
    <controls:PDFViewerControl Background="Black"/>
</Page>
```

The view model internally uses `Document` object to open PDF documents and its very useful property `Navigator` to subscribe to navigation events and avoid implementing its own navigation logic. The navigator object keeps track of the currently “selected” page and current page index as well. It can also be used to move forward or backward, and all you need to do is to handle `Navigated` event. See the code excerpts below.

Opening the file:

```
        currentDocument = new Document(documentStream);

        // subscribe to navigation events and go to first page
        currentDocument.Navigator.Navigated += Navigator_Navigated;
        currentDocument.Navigator.Move(0, Origin.Begin);
```

Navigation can be implemented as follows:

```
        currentDocument.Navigator.MoveBackward();

        or

        currentDocument.Navigator.MoveForward();
```

where `currentDocument` object is a currently opened pdf document.

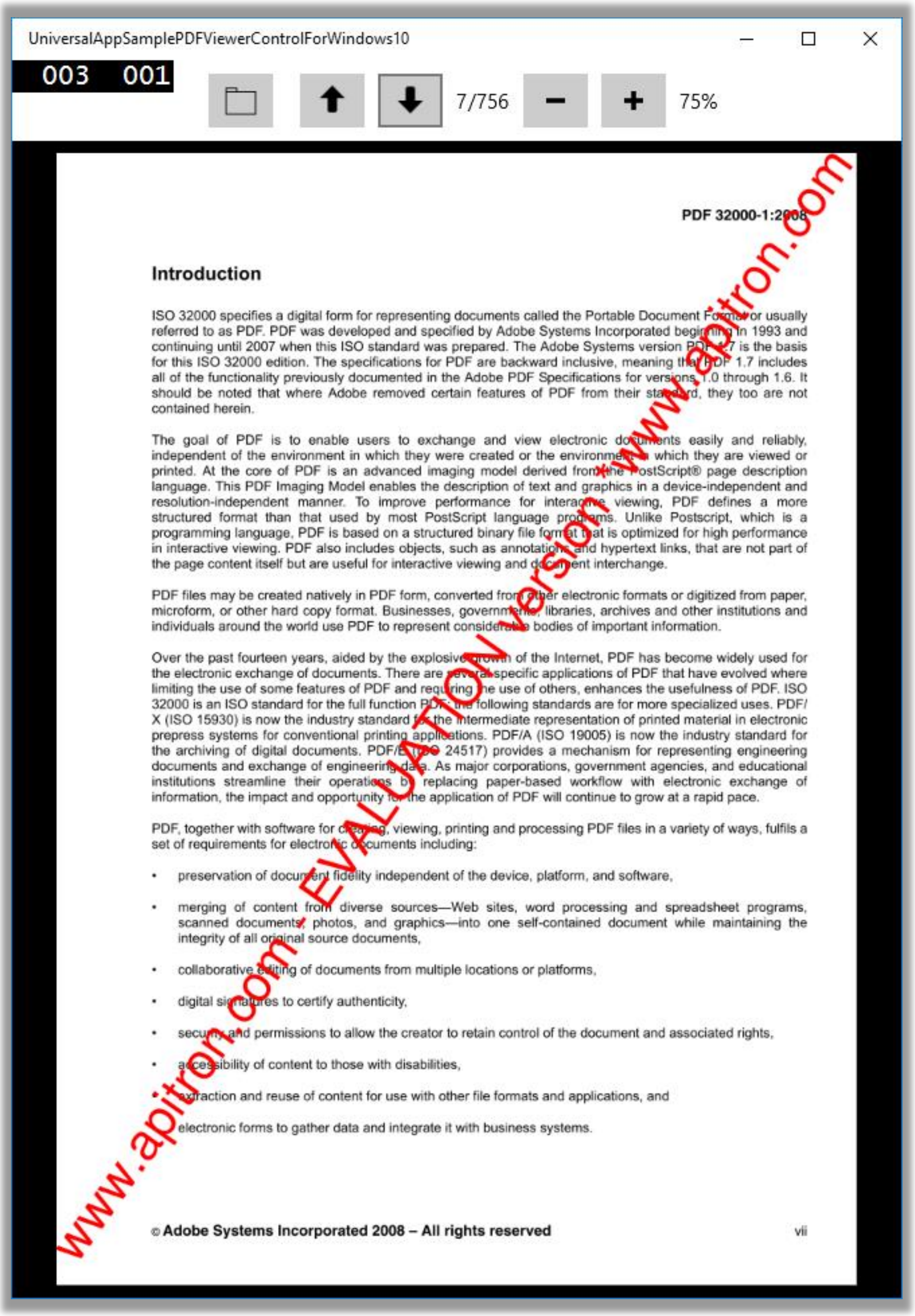
Rendering looks very simple:

```
private void RenderPage(Apitron.PDF.Rasterizer.Page page)
{
    if (page != null)
    {
        ErrorLogger logger = new ErrorLogger();

        WriteableBitmap bm = page.Render(new Resolution(96, 96), new RenderingSettings(), logger);

        myImage.Source = bm;
    }
}
```

The view model also provides many helper properties that control uses to update its UI elements.

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

Here is the screenshot of the actual windows 10 running app (desktop):



PDF 32000-1:2008

Introduction

ISO 32000 specifies a digital form for representing documents called the Portable Document Format or usually referred to as PDF. PDF was developed and specified by Adobe Systems Incorporated beginning in 1993 and continuing until 2007 when this ISO standard was prepared. The Adobe Systems version PDF 1.7 is the basis for this ISO 32000 edition. The specifications for PDF are backward inclusive, meaning that PDF 1.7 includes all of the functionality previously documented in the Adobe PDF Specifications for versions 1.0 through 1.6. It should be noted that where Adobe removed certain features of PDF from their standard, they too are not contained herein.

The goal of PDF is to enable users to exchange and view electronic documents easily and reliably, independent of the environment in which they were created or the environment in which they are viewed or printed. At the core of PDF is an advanced imaging model derived from the PostScript® page description language. This PDF Imaging Model enables the description of text and graphics in a device-independent and resolution-independent manner. To improve performance for interactive viewing, PDF defines a more structured format than that used by most PostScript language programs. Unlike Postscript, which is a programming language, PDF is based on a structured binary file format that is optimized for high performance in interactive viewing. PDF also includes objects, such as annotations and hypertext links, that are not part of the page content itself but are useful for interactive viewing and document interchange.

PDF files may be created natively in PDF form, converted from other electronic formats or digitized from paper, microform, or other hard copy format. Businesses, governments, libraries, archives and other institutions and individuals around the world use PDF to represent considerable bodies of important information.

Over the past fourteen years, aided by the explosive growth of the Internet, PDF has become widely used for the electronic exchange of documents. There are several specific applications of PDF that have evolved where limiting the use of some features of PDF and requiring the use of others, enhances the usefulness of PDF. ISO 32000 is an ISO standard for the full function PDF; the following standards are for more specialized uses. PDF/X (ISO 15930) is now the industry standard for the intermediate representation of printed material in electronic prepress systems for conventional printing applications. PDF/A (ISO 19005) is now the industry standard for the archiving of digital documents. PDF/E (ISO 24517) provides a mechanism for representing engineering documents and exchange of engineering data. As major corporations, government agencies, and educational institutions streamline their operations by replacing paper-based workflow with electronic exchange of information, the impact and opportunity for the application of PDF will continue to grow at a rapid pace.

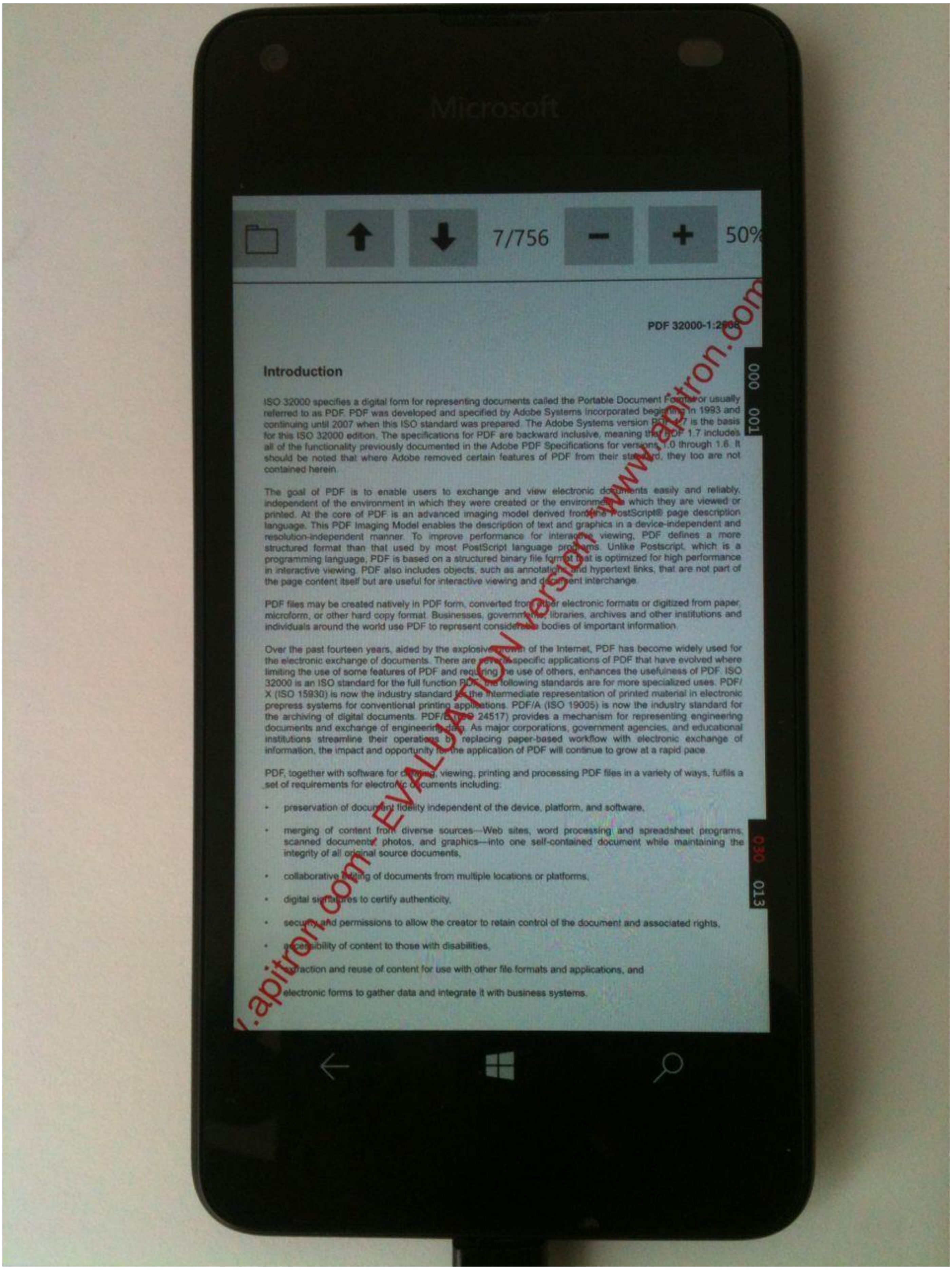
PDF, together with software for creating, viewing, printing and processing PDF files in a variety of ways, fulfils a set of requirements for electronic documents including:

- preservation of document fidelity independent of the device, platform, and software,
- merging of content from diverse sources—Web sites, word processing and spreadsheet programs, scanned documents, photos, and graphics—into one self-contained document while maintaining the integrity of all original source documents,
- collaborative editing of documents from multiple locations or platforms,
- digital signatures to certify authenticity,
- security and permissions to allow the creator to retain control of the document and associated rights,
- accessibility of content to those with disabilities,
- extraction and reuse of content for use with other file formats and applications, and
- electronic forms to gather data and integrate it with business systems.

© Adobe Systems Incorporated 2008 – All rights reserved vii

Pic. 1 PDF Viewer control sample for Universal Windows Application (Windows 10)

And photo of the application running on the windows 10 phone:



PDF 32000-1:2008

Introduction

ISO 32000 specifies a digital form for representing documents called the Portable Document Format or usually referred to as PDF. PDF was developed and specified by Adobe Systems Incorporated beginning in 1993 and continuing until 2007 when this ISO standard was prepared. The Adobe Systems version PDF 1.7 is the basis for this ISO 32000 edition. The specifications for PDF are backward inclusive, meaning that PDF 1.7 includes all of the functionality previously documented in the Adobe PDF Specifications for versions 1.0 through 1.6. It should be noted that where Adobe removed certain features of PDF from their standard, they too are not contained herein.

The goal of PDF is to enable users to exchange and view electronic documents easily and reliably, independent of the environment in which they were created or the environment in which they are viewed or printed. At the core of PDF is an advanced imaging model derived from the PostScript® page description language. This PDF Imaging Model enables the description of text and graphics in a device-independent and resolution-independent manner. To improve performance for interactive viewing, PDF defines a more structured format than that used by most PostScript language programs. Unlike Postscript, which is a programming language, PDF is based on a structured binary file format that is optimized for high performance in interactive viewing. PDF also includes objects, such as annotations and hypertext links, that are not part of the page content itself but are useful for interactive viewing and document interchange.

PDF files may be created natively in PDF form, converted from other electronic formats or digitized from paper, microform, or other hard copy format. Businesses, governments, libraries, archives and other institutions and individuals around the world use PDF to represent considerable bodies of important information.

Over the past fourteen years, aided by the explosive growth of the Internet, PDF has become widely used for the electronic exchange of documents. There are several specific applications of PDF that have evolved where limiting the use of some features of PDF and requiring the use of others, enhances the usefulness of PDF. ISO 32000 is an ISO standard for the full function PDF; the following standards are for more specialized uses. PDF/X (ISO 15930) is now the industry standard for the intermediate representation of printed material in electronic prepress systems for conventional printing applications. PDF/A (ISO 19005) is now the industry standard for the archiving of digital documents. PDF/E (ISO 24517) provides a mechanism for representing engineering documents and exchange of engineering data. As major corporations, government agencies, and educational institutions streamline their operations by replacing paper-based workflow with electronic exchange of information, the impact and opportunity for the application of PDF will continue to grow at a rapid pace.

PDF, together with software for creating, viewing, printing and processing PDF files in a variety of ways, fulfils a set of requirements for electronic documents including:

- preservation of document fidelity independent of the device, platform, and software,
- merging of content from diverse sources—Web sites, word processing and spreadsheet programs, scanned documents, photos, and graphics—into one self-contained document while maintaining the integrity of all original source documents,
- collaborative editing of documents from multiple locations or platforms,
- digital signatures to certify authenticity,
- security and permissions to allow the creator to retain control of the document and associated rights,
- accessibility of content to those with disabilities,
- extraction and reuse of content for use with other file formats and applications, and
- electronic forms to gather data and integrate it with business systems.

Pic. 2 Apitron PDF Rasterizer based PDF Viewer control running on Lumia 550 (Windows 10)

Summary

One of the possible applications of the Apitron PDF Rasterizer for .NET component is implementing PDF viewers for supported platforms. In this article we've demonstrated how to create a PDF viewer control for Windows 10 Universal application. The code runs smoothly on your desktop and mobile devices and can be used as a base for further development.

Contact us if you have any questions regarding this code sample or PDF processing in general, and we'll be happy assist you.