AMYUNI Technologies

Amyuni PDF Creator for ActiveX

For PDF and XPS

Version 6.5 Professional

Quick Start Guide for Developers

Updated 17 June 2021

AMYUNI Consultants – AMYUNI Technologies
www.amyuni.com

Contents

Legal Information

PDFCreactiveDoc.exe is provided as a sample application for Developers, it cannot be distributed with the Developers' application. The source-code for this executable can be requested by contacting support@amyuni.com

Acknowledgments

This software uses the deflate algorithm developed by Jean-loup Gailly (jloup@gzip.org) and Mark Adler (madler@alumni.caltech.edu). This software is also based in part on the work of the Independent JPEG Group and on parts of the FreeType library.

Description of the Modules

The PDF Creator product is composed of five modules that can be integrated and distributed under the developer license agreement.

PDFCreactiveX.dll

This is the main PDF ActiveX control that provides viewing, editing, and printing of PDF documents. This DLL should be registered on the client system.

acPdfCrDb.dll

This DLL handles the interface with the database and should be registered on the client system. It is an optional DLL required only when using the default database interface provided with the PDF control.

acPDFCrExt.dll

This is the "extensions" DLL control and should be registered on the client system. It contains additional controls such as the Bookmarks tree control, data formatting control, and the default properties dialog box. This DLL is not required when not using these additional controls.

xmllite.dll

This module contains the XML parser needed to process XPS documents. This module is needed only if XPS support is required.

PDFCreactiveDoc.exe

This is a complete PDF viewing/editing application built using the previous modules. This application can be used for testing purposes by the developers or as a basis for providing a custom user interface around the PDF ActiveX control. The source-code for this executable is provided, but only the executable and not the source-code can be distributed.

Note that if all the files are copied to the same directory, **PDFCreactiveDoc.exe** will register all DLLs every time it is launched, so there is no need to manually register each DLL separately.

Administrative privileges are required under Windows Vista, Windows 7, Windows 8, Windows 8.1 and Windows 10 in order to register ActiveX components on the end-user systems. Registered DLLs are also shared among installed applications which can create some version conflicts. In order to avoid registration and DLL conflicts, an alternative method for using the PDF Creator ActiveX controls can be used. This method is described in this forum: https://www.amyuni.com/forum/viewtopic.php?f=13&t=2438.

Windows 98/Me

To work under Windows 98/Me, version 2 and above of the Amyuni PDF Creator requires the Microsoft Unicode Layer for Windows 98/Me. This is a DLL named **unicows.dll** that can be obtained from:
http://www.microsoft.com/msdownload/platformsdk/sdkupdate/psdkredist.htm

General Operation

The Amyuni PDF Creator was designed for two main purposes:

- Viewing, editing and printing PDF documents.
- Creating reports, forms, and general documents directly in PDF format.

The ActiveX controls that are part of the PDF Creator can be integrated into most development environments to provide the final user with advanced PDF document management capabilities.

The PDF Creator control can be in one of four modes:

- Design mode.
- Run or Compiled mode.
- Annotation mode.
- Print preview mode.

When a blank document is first created, the control is in design mode. The user or developer can add objects to the document, delete objects, modify object properties, and do all editing operations allowed by the PDF Creator control.

When an existing PDF document is opened, two things can happen:

1. The PDF document was generated by the PDF Creator and contains all design information generated by the control. In this case, the document is opened in the same state as when it was saved, i.e., Design, Run, Annotate, or PrintPreview modes.
2. The PDF document was generated by another tool such as the Amyuni PDF Converter or other PDF generation tools. In this case, the document is opened immediately in Run mode. The document can only be read or printed and if the user has enough rights on the document, the user can switch the document to Annotation or Design Mode, and Edit it using the PDF Creator interface.

The documents created by the PDF Creator can contain fields with formulas or data coming from a database. Compiling the document instructs the PDF Creator to compute those formulas and fetch information from the database. Once compiled, the document switches to **Run** mode and only the fields or objects defined as Editable can be modified in Run mode.

In Annotation mode, the contents of the original document cannot be modified; the user can only add or modify PDF annotations such as Text, Line, Highlighter, or Sticky note annotations.

Inserting the PDF Creator Control into a Project

Adding the control to the Form Designer

This is specific to each development environment, but the procedure is very similar in all cases. The PDF Creator component should be added to an existing project by using the Form Designer's toolbox Add Components context menu item (Choose Items in VS2017) dialog box as follows:

Right-click on the toolbox in the form designer and select **Choose Items**:

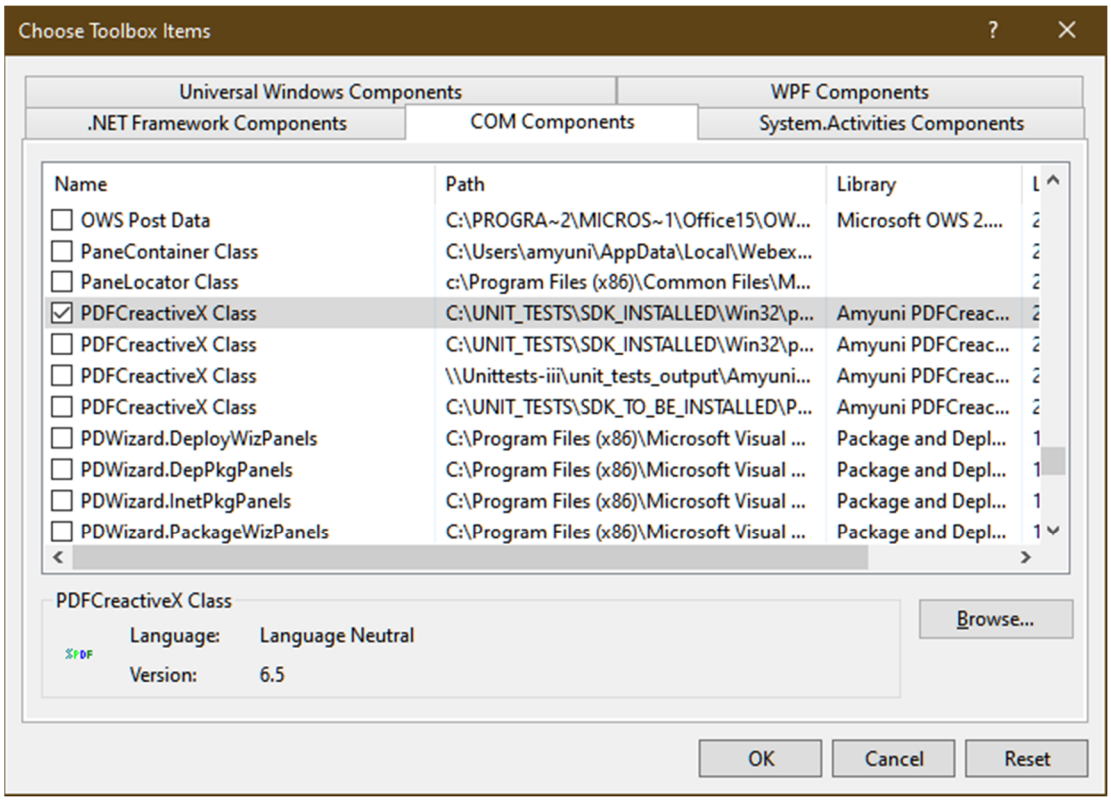

Figure 1: Choose Toolbox Items

The control shows in the list of components as "*Amyuni PDFCreactiveX Component*". Once inserted into the project, the component appears in the components toolbar and can be inserted on a form.

Inserting the control in Visual Studio 2005-2008
When the developer wants to insert the control in Visual Studio 2005, he must first add the OLE Automation COM reference to the project. To do this, Right-click ***References*** in the ***Solution Explorer*** and choose **Add…** from the popup menu. A dialog box will show up. Go to the page on that dialog that is labeled "COM" and from the list of COM type libraries on that page check the ***OLE Automation*** library and hit OK. Now, the component can be dragged from the toolbox onto the form without error.
The reason behind this procedure is that Visual Studio 2005 unlike its predecessor VS2003, does not automatically add the necessary and needed COM reference to the OLE Automation type library - **stdole.tlb** or **stdole2.tlb**.

Using the PDF Creator as a dll in .NET

To use the Amyuni PDF Creator library as a dll, without adding the ActiveX control to your project, you need to generate the type library and ActiveX control wrappers for it and add these to your references by:

- Start the visual studio command prompt by navigating to:
  Start Menu / Programs / Visual Studio 2019/ Visual Studio Tools.
- Starting the application **Developer Command Prompt for VS2019** as administrator. Change to the folder where the PDF Creator library was installed and run the **aximp** command on it:

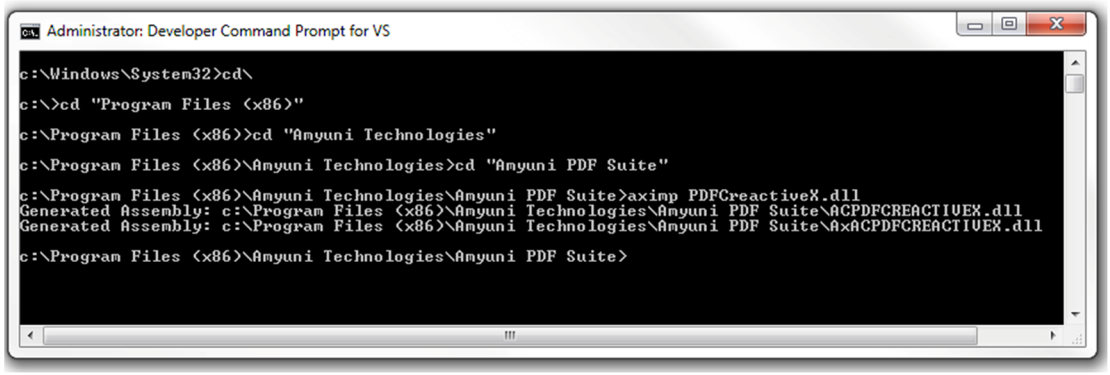

Figure 2: Using the aximp tool.

Next go to your Visual Studio project's References folder and add the two dlls generated above by browsing to their location.

To use the library through the COM IDispatch interface, use the ProgID **"PDFCreactiveX.PDFCreactiveX.6.5"**

C# Sample

This is a sample showing how to create a simple PDF document from a .NET application:

```
PDFCreactiveXClass pdf = new PDFCreactiveXClass();
// activate the control before doing any operation on it
pdf.SetLicenseKey("Amyuni", "07EFC…F198");
// create a static text object
pdf.CreateObject(ObjectTypeConstants.acObjectTypeText,"HelloText");
acObject helloText = pdf.GetObjectByName("HelloText");
helloText["Text"] = "Hello World!";
// set the position and size of the object
helloText["Top"] = 200;
helloText["Left"] = 200;
helloText["Right"] = 1000;
helloText["Bottom"] = 400;
// save the resulting PDF
pdf.Save("HelloWorld.pdf", FileSaveOptionConstants.acFileSaveView);
```

Using the PDF Creator in a C++ environment

Please refer to the online documentation for related samples at:

https://www.amyuni.com/WebHelp/Developer_Documentation.htm

Setting the Licensing Information

Before any operation can be done on the PDF Creator control, the control needs to be activated using the **SetLicenseKey** method as shown below.

C++ Sample

```
BOOL CSampleCodeCPPDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    if (FAILED( hr = pdf->SetLicenseKey( _bstr_t("Amyuni"),
_bstr_t("07...84") ) ))
    {
        // code that will be entered if hr is a failure code and if
        // raw_interfaces_only is used in the #import directive
        s.Format( _T("\nhr = 0x%x; Failed to set the license key\n"), hr
);
        AfxMessageBox( s );
        return FALSE;
    }
    return TRUE;
}
```

C# Sample

This is a sample showing the initialization method in the **Form_Load** event handler of a typical C# windows application:

```
private void Form1_Load(object sender, EventArgs e)
{
        pdf.SetLicenseKey("Amyuni Tech", "07EFCDA...4FE");
}
```

Using the PDF Creator Control

Properties

The properties of the PDF control allow the user to change different aspects of its behavior. For a complete list of creator control properties, please refer to the links provided at the end of this document.

Properties whose values are simple data-types

These could be strings, integers, or floating point values. Below is a small example that hides the status bar, and changes the look of the rulers on the control.

> In C++, Boolean properties are actually integers. If you are programming in .NET, an integer value needs to be used and a direct Boolean will cause a compile error.

C++ Sample

This is a sample showing the initialization method in a typical MFC dialog application:

```
void CMFCDialogAppDlg::SetInitialLook()
{
	CWnd *pdfControl = GetDlgItem( IDC_PDFCREACTIVEX1 );
	ASSERT( pdfControl );
	LPUNKNOWN lpUnknown = pdfControl->GetControlUnknown();
	ASSERT( lpUnknown );
	IPDFCreactiveXPtr pdf = lpUnknown;
	// Remove the status bar
	pdf->StatusBar = false;

	//Set the background color of the ruler.
	//Blue component to 155
	//Green component to 128
	//Red component to 64
	pdf->RulerBackColor = (255 << 16) | (128 << 8) | 64;
	// make the ruler a bit smaller than the default size of 30
	pdf->RulerSize = 20;
}
```

C# Sample

This is a sample showing the initialization method in a typical .NET application:

```
private void Form1_Load(object sender, EventArgs e)
{
        // Remove the status bar
        pdf.StatusBar = 0;
        /*
        Set the background color of the ruler.
        Blue component to 155
        Green component to 128
        Red component to 64
        */
        pdf.RulerBackColor = (255 << 16) | (128 << 8) | 64;

        // make the ruler a bit smaller than the default size of 30
        pdf.RulerSize = 20;
}
```

Properties that have many possible, predefined values

These are of two types:

1. Some have predefined constant names that represent the possible values. The attribute **ReportState** is an example of these. In this case, either the enumeration's literal name can be used or the numerical value. In this case, the property type is a name, and both the ***Name*** and ***Value*** column headers are in italics.

Property	**Descritption**	**Type**	**Default Value**
ReportState	Current state for document or report	ReportStateConstants	acReportStateRun

ReportStateConstants

Name	**Description**	***Value***
acReportStateRun	Document in Run mode	0
acReportStateDesign	Document in Design mode	1
acReportStateLoading	Document is being loaded from file	2
acReportStatePrintPreview	Document in PrintPreview mode	3
acReportStateAnnotate	Document in annotation mode	4

Example usage:

C++

```
pdf->ReportState = acReportStateRun;
pdf->PutReportState(acReportStateRun);
pdf->put_ReportState(acReportStateRun);
pdf->put_ReportState(0);
```

C#

```
pdf.ReportState = ACPDFCREACTIVEX.ReportStateConstants.acReportStateRun;
```

2. Some do not have predefined names, and the numerical values must be used, like the **Duplex** property. In this case the property type is an integer only the ***Value*** column header is in italics.

Property	**Descritption**	**Type**	**Default Value**
Duplex	Activate duplex printing of document	Integer	0
Duplex Options (value needs to be entered in code, since no enumeration is defined)			
Name	**Description**		***Value***
Printer default	Printer default		0
No duplex	No duplex		1
Vertical duplex	Vertical duplex		2
Horizontal duplex	Horizontal duplex		3

```
C++
pdf->Duplex = 0;
C#
pdf.Duplex = 0;
```

C# Example on using the TemplateMode property:

The control supports the use of two documents simultaneously:

- A passive document that acts as a background (the template document).
- A document in the foreground where all the editing takes place.

This property is used to load a document that will be used as the passive background document. To load a background document, do the following steps:

1. Set the **TemplateMode** attribute to the value one "1".
2. Load a background document like you would load any other document.
3. After loading, the value of **TemplateMode** will be automatically reset to zero "0", a foreground "active" document can be loaded and manipulated normally.

If the template document has a smaller number of pages than the foreground document, the template document will be repeated as needed, depending on the value of **TemplateRepeat**.

```
axPDFCreactiveX1.TemplateMode = 1;
// open a PDF file as the background, or template, document.
axPDFCreactiveX1.Open("Background.pdf", "");
// create a text object inthe active or foreground document.
axPDFCreactiveX1.CreateObject(ACPDFCREACTIVEX.ObjectTypeConstants.acObject
   TypeText, "Text1" );
axPDFCreactiveX1.set_ObjectAttribute ("Text1", "Left", 200);
axPDFCreactiveX1.set_ObjectAttribute ("Text1", "Top", 200);
axPDFCreactiveX1.set_ObjectAttribute ("Text1", "Right", 600);
axPDFCreactiveX1.set_ObjectAttribute ("Text1", "Bottom", 400);
axPDFCreactiveX1.set_ObjectAttribute ("Text1", "Text", "Hello");
// refresh the control to make the text object appear
axPDFCreactiveX1.Refresh ();
```

Methods

Note on return values, note on handling HRESULT, and its possible values.

Return values

This method launches an exception if the object cannot be created. The messages in the exception are:

E_NOTIMPL The license key that is provided does not enable modifying the document

E_ACCESSDENIED The document security settings do not allow the user to modify the document

E_FAIL Failed to create the object

Links to Support and Documentation:

If you have any questions or problems with our products, the following resources are available to you through our web site:

Online Documentation:
https://www.amyuni.com/WebHelp/Developer_Documentation.htm#index.htm

Frequently Asked Questions:
https://www.amyuni.com/forum/viewforum.php?f=18

Technical Notes:
https://www.amyuni.com/en/resources/technicalnotes/

User forum:
https://www.amyuni.com/forum/index.php

Posting questions to our technical support staff:
https://www.amyuni.com/en/support/getsupport/

We also provide some additional tools that can be downloaded free of charge and used with the PDF Creator product. These tools are available at:
https://www.amyuni.com/en/resources/freetools/