Report Number: WUCSE-2006-2

2006-01-01

activePDF-Toolk

Washington University in St. Louis Computer Science Engineering Department

This document provides information for deploying activePDF Toolkit Professional in a development environment. This document is organized into four sections: Getting Started, Tutorials, Technical Reference and the Toolkit Appendices. The Getting Started section covers setup and installation, includes a product overview and information related to operating Toolkit Professional. Tutorials includes examples of many Toolkit features, including PDF generation and form filling. All of the tutorials can be used with activePDF Toolkit. Technical Reference provides detailed information on Toolkit's objects, subobjects, methods and properties.

... Read complete abstract on page 2.

activePDF-Toolk

Washington University in St. Louis Computer Science Engineering Department

Complete Abstract:

This document provides information for deploying activePDF Toolkit Professional in a development environment. This document is organized into four sections: Getting Started, Tutorials, Technical Reference and the Toolkit Appendices. The Getting Started section covers setup and installation, includes a product overview and information related to operating Toolkit Professional. Tutorials includes examples of many Toolkit features, including PDF generation and form filling. All of the tutorials can be used with activePDF Toolkit. Technical Reference provides detailed information on Toolkit's objects, subobjects, methods and properties.

WUDEV-2006-2

activePDF-Toolkit

Authors: activePDF

March 1, 2006

Corresponding Author: jpw4@cec.wustl.edu

Notes: activePDF-Toolkit.pdf
activePDF-Toolkit.pdf

WUDEV-2006-2

Abstract

This document provides information for deploying activePDF Toolkit Professional in a development environment. This document is organized into four sections: Getting Started, Tutorials, Technical Reference and the Toolkit Appendices.
The Getting Started section covers setup and installation, includes a product overview and information related to operating Toolkit Professional.
Tutorials includes examples of many Toolkit features, including PDF generation and form filling. All of the tutorials can be used with activePDF Toolkit.
Technical Reference provides detailed information on Toolkit™s objects, subobjects, methods and properties.

SeasWeb Development Site - Washington University in St. Louis
Campus Box 123456 - St. Louis, MO - 63130 - ph: 555-1212

2006-2

activePDF-Toolkit

Authors: activePDF

March 1, 2006

Corresponding Author: jpw4@cec.wustl.edu

Notes: activePDF-Toolkit.pdf
activePDF-Toolkit.pdf

Abstract

This document provides information for deploying activePDF Toolkit Professional in a development environment. This document is organized into four sections: Getting Started, Tutorials, Technical Reference and the Toolkit Appendices.
The Getting Started section covers setup and installation, includes a product overview and information related to operating Toolkit Professional.
Tutorials includes examples of many Toolkit features, including PDF generation and form filling. All of the tutorials can be used with activePDF Toolkit.
Technical Reference provides detailed information on Toolkit™s objects, subobjects, methods and properties.

activePDF®
Leading the iPaper Revolution™

Toolkit™ Professional

Product Manual

License

activePDF, Inc. END USER LICENSE AGREEMENT

NOTICE TO USER:

THIS IS A CONTRACT. BY INSTALLING THIS SOFTWARE YOU ACCEPT ALL THE TERMS AND CONDITIONS OF THIS AGREEMENT.

This activePDF, Inc. ("ACTIVEPDF") End User License Agreement ("EULA") accompanies this activePDF product and related explanatory materials ("SOFTWARE"). The term "SOFTWARE" shall also include any upgrades, modified versions or updates of the Software licensed to you by ACTIVEPDF. Please read this EULA carefully. If you are reading this prior to electronic distribution, you will be asked to accept this agreement and continue to install or, if you wish to decline this agreement, in which case you will not be able to use the Software.

The SOFTWARE is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The SOFTWARE is licensed, not sold.

1. DEFINITIONS

I. SERVER - A single computer wholly owned, rented or leased by a single individual or entity on which one or more applications load and execute SOFTWARE in the memory space of that computer so that one or more users may access it.

II. VIRTUAL SERVER - A single computer wholly owned, rented or leased by one individual or entity and who, in turn, rents or leases access to this computer to other individuals or entities and on which one or more applications load and execute SOFTWARE in the memory space of that computer so that multiple users may access it.

III. DEVELOPMENT - The act of programming a tool or application that interacts with SOFTWARE.

IV. APPLICATION SERVICE PROVIDER - An individual or entity ("OWNER") and who, in turn, charges other individuals or entities a fee for access to software applications wholly owned or licensed and maintained by the OWNER. The OWNER's application(s) load and execute SOFTWARE in the memory space of that computer so that multiple users may access it.

2. GRANT OF LICENSE : Upon generation of a valid license file (for Standard and Professional) or a valid serial number (for Lite) by ACTIVEPDF for you, this EULA grants you the following rights:

a. SOFTWARE PRODUCT : ACTIVEPDF grants to you as an individual, a non-transferable, nonexclusive license to use the SOFTWARE PRODUCT as follows:

I. IF YOU PURCHASED A SERVER LICENSE - Usage is restricted to a single SERVER for a single individual or entity. All SERVERS that use this product must be licensed.

II. IF YOU PURCHASED A VIRTUAL SERVER LICENSE - Usage is restricted to a single individual or entity. All other individuals or entities that can access or use SOFTWARE must purchase a separate VIRTUAL SERVER LICENSE.

III. IF YOU PURCHAED A DEVELOPMENT LICENSE - Usage is restricted to a single computer for the purposes of DEVELOPMENT AND TESTING ONLY. Files generated or manipulated with the SOFTWARE MAY NOT BE USED IN A PRODUCTION ENVIRONMENT.

b. If you are an entity, ACTIVEPDF grants you the right the same rights as above.

c. UNTIL A VALID LICENSE FILE (FOR STANDARD AND PROFESSIONAL) OR A VALID SERIAL NUMBER (FOR LITE) HAS BEEN GENERATED FOR YOU BY ACTIVEPDF, YOU MAY ONLY USE THE SOFTWARE FOR A TRIAL PERIOD NOT TO EXCEED FIFTEEN (15) DAYS FROM THE FIRST TIME THE SOFTWARE IS RAN. YOU AGREE TO REMOVE ANY COPIES OF THE SOFTWARE UPON EXPIRATION OF THE TRIAL PERIOD IF YOU DO NOT PURCHASE THE SOFTWARE. YOU MAY NOT DISTRIBUTE ANY PORTION OF THE SOFTWARE, INCLUDING THOSE PORTIONS OUTLINED UNDER "REDISTRIBUTABLE CODE" BELOW, UNTIL YOU ARE ISSUED A VALID SERIAL NUMBER. NO LICENSE IS GRANTED UNTIL THAT TIME.

d. ELECTRONIC DOCUMENTS : Solely with respect to electronic documents included with the SOFTWARE, you may make an unlimited number of copies (either in hardcopy or electronic form), provided that such copies shall be used only for internal purposes and are not republished or distributed to any third party.

e. STORAGE/NETWORK USE : You may store or install one (1) copy of the SOFTWARE on a storage device, such as a network server, for backup and archival purposes only. A license for the SOFTWARE may not be shared or used concurrently on different computers.

f. You agree not to modify the Producer or Creator fields within any PDF documents created by SOFTWARE.

g. ACTIVEPDF reserves all rights not expressly granted.

3. COPYRIGHT. The SOFTWARE is owned by ACTIVEPDF and its suppliers, and its structure, organization and code are the valuable trade secrets of ACTIVEPDF and its suppliers. The SOFTWARE is protected by United States Copyright Law and International Treaty provisions. You may use trademarks only insofar as required to comply with Section 1 of this EULA and to identify printed output produced by the Software, in accordance with accepted trademark practice, including identification of trademark owner's name. Such use of any trademark does not give you any rights of ownership in that trademark. Except as stated above, this EULA does not grant you any intellectual property rights in the Software.

4. RESTRICTIONS. You may not resell, transfer, rent or lease the SOFTWARE. You agree not to modify, adapt, translate, reverse engineer, decompile, disassemble or otherwise attempt to discover the source code of the Software. You may not alter or modify in any way the installer for the Software, or create a new installer for the Software. Redistribution rights, if any, are outlined below.

5. ADDITIONAL RESTRICTIONS.

NONE

6. UPGRADES. If the SOFTWARE is labeled as an upgrade, you must be properly licensed to use a product identified by ACTIVEPDF as being eligible for the upgrade in order to use the SOFTWARE. SOFTWARE labeled as an upgrade replaces and/or supplements the product that formed the basis

Copyrights, Trademarks and Credits

Copyright Notice

Trademarks

Credits

Producer: Tim Sullivan

Technical Writers: Nicole Diaz, Arleigh Rogers

Supporting Examples: Derek Andelin, Derek Dieter

Contacting activePDF, Inc.

activePDF is a leading provider of PDF creation, conversion and development tools, catering to the increasing demand for automated document management practices within today's key vertical enterprise markets. Founded in January 2000, activePDF has since built a substantial market presence with over 10,000 customers and approximately 35,000 server licenses worldwide. Headquartered in Mission Viejo, California, activePDF is privately held and self-funded, with a global distribution network spanning every continent.

For more information, please select one of the following options:

- activePDF, Inc. Corporate Headquarters.
- Technical Support.
- Sales.

activePDF, Inc. Corporate Headquarters

You can contact the activePDF, Inc. corporate headquarters at the following address and phone number:

activePDF, Inc.
27405 Puerto Real Suite 100
Mission Viejo, California 92691-6314
United States
Toll Free (U.S.): (866) GOTO PDF
Elsewhere: +1-949-582-9002
Fax: (949) 582-9004
World Wide Web: www.activePDF.com

Technical Support

At activePDF, we are committed to providing you with timely answers to all of your technical support questions. For current hours of operation and details about support offerings, please visit the activePDF website at www.activepdf.com.

Sales

activePDF is a leading provider of server-side PDF generation and manipulation tools. For information on other activePDF product offerings, please contact activePDF's Sales Department by calling (866) GOTO PDF in the U.S., or +1-949-582-9002 elsewhere. You can also visit us on the web at www.activePDF.com.

Table of Contents

About this document

This document provides information for deploying activePDF Toolkit Professional in a development environment. This document is organized into four sections: *Getting Started*, *Tutorials*, *Technical Reference* and the *Toolkit Appendices*.

The Getting Started section covers setup and installation, includes a product overview and information related to operating Toolkit Professional.

Tutorials includes examples of many Toolkit features, including PDF generation and form filling. All of the tutorials can be used with activePDF Toolkit.

Technical Reference provides detailed information on Toolkit's objects, subobjects, methods and properties.

NOTE: Unless otherwise specified, all of the code examples in this document were written using the Visual Basic Scripting language (VBScript).

The *Toolkit Appendices* include a brief overview of the PDF coordinate system, a list of the run time file dependencies for use with the activePDF Toolkit Run Time license, a list of the image types that can be converted using activePDF Toolkit, supported PDF comment colors and an introduction to symbologies found in the Barcode object.

NOTE: For more information regarding the activePDF Toolkit Run Time license, please contact activePDF Sales.

Who Should Read This Guide

This guide has been written for the developer who wants to programmatically generate PDFs and control the resultant output. The guide assumes you have a general knowledge of PostScript® and PDF, and that you are comfortable programming in a COM-enabled environment.

Documentation Feedback

activePDF strives to produce quality technical documentation. If you have comments or suggestions regarding our help files, PDF or print manuals, please send an email to documentation@activePDF.com.

Please include the following information in your message:

- Product name and version number,
- Print manual, PDF or help file,
- Section or Topic title,
- Brief description of content, and
- Your suggestion for improvement or correction.

We greatly appreciate your suggestions for improving the quality of activePDF's documentation.

NOTE: This email address is only for documentation feedback. If you have a technical question, please contact Technical Support.

Getting Started

activePDF Toolkit's programmable COM object simplifies PDF manipulation, affording full control over your PDF output. Licensed per server, Toolkit allows you to append, stamp, stitch, secure, split, merge, form-fill PDFs and more. Some of the functions available in Toolkit include:

- Form Field Creation and Filling
- Dynamic PDF Generation
- Merging and Copying
- Stamping Text and Images
- Stitching
- PDF Security
- Digital Signatures
- Linearization

Form Field Creation and Filling

activePDF Toolkit enables you to populate PDF form fields dynamically, from a data-source, XML data or another PDF form. Additionally, Toolkit allows you to generate form fieldss on the fly to precisely control the layout of database reports.

Dynamic PDF Generation

Toolkit enables you to convert text and images to PDF. You can also create PDF pages, draw lines, and apply colors and text styles.

Merging and Copying

activePDF Toolkit's merge and copy features enable you to append pages to and extract pages from your PDF files, creating comphrensive PDF documents.

Stamping Text and Images

Stamping involves placing text and images on PDF pages. Watermarking, custom headings and page numbering are example applications of this feature.

Stitching

Stitching allows you to combine one or more PDF documents, creating a custom PDF with precise placement, multi-up, on a single page or onto several pages.

PDF Security

Toolkit allows you to encrypt and decrypt files using the PDF security model. Toolkit supports 40 and 128-bit encryption, enabling you to password-protect PDF files, disable end-user printing, and prevent copying of text and graphics. Toolkit also includes a fingerprint technology, enabling you to verify the integrity of your PDF documents.

Digital Signatures

Digital Signatures use strong encryption to authenticate the identity of the PDF creator and the integrity of the PDF content.

Linearization

Toolkit's Linearization features prepare large PDF documents for byte-serving over the web. This allows your users to view a specified page instantly while the remaining pages are loaded silently in the background.

Installing activePDF Toolkit

You can install activePDF Toolkit Professional Edition from the program CD or as an internet download. We strongly recommend that you carefully review the System Requirements prior to installation. When you are ready to install activePDF Toolkit, refer to the instructions that best suit your installation type:

- Installing from a CD.
- Installing from Internet Download.

NOTE: For additional assistance, please contact Technical Support.

System Requirements

In order to install activePDF Toolkit, your computer should be equipped with the following:

Operating System Requirements

- Microsoft Windows NT® 4.0 (Service Pack 5 Minimum), or
- Microsoft Windows® 95, 98, ME, 2000, XP, or
- Microsoft Windows Server™ 2003.
- Strong Encryption (128-bit encryption).

NOTE: Strong encryption is only required for encrypting or decrypting 128-bit PDF documents.

Minimum Recommended Hardware Requirements

- Pentium 200-MHz or higher.
- 32 MB of RAM.
- 5 MB of Hard Disk Space.

Installing from a CD

Use the following procedure if you are installing activePDF Toolkit Professional Edition for the first time, or if you are upgrading from a previous or evaluation version.

If you are upgrading from a previous or evaluation version, we recommend that you first remove the previous or evaluation version prior to installing the new version.

To install activePDF Toolkit from the CD

1. Close all programs.
2. Insert the activePDF Toolkit CD into your computer's CD drive.

NOTE: If AutoRun is enabled on your system, the installation starts automatically and you can skip steps 3 and 4.

3. On the taskbar, click the **Start** button, and then click **Run**.
4. In the **Open** box, type **X:\aptk40pe.exe** and click **OK**.
5. Follow the on-screen instructions. If prompted, restart your computer.

Installing from Internet Download

You can download the evaluation version of activePDF Toolkit Professional Edition from our website, www.activepdf.com. The evaluation copy of Toolkit is a fully functional version of the program, which expires after 15 days and all output contains an activePDF watermark.

To install activePDF Toolkit Professional Evaluation from an internet download

1. Close all programs.
2. Download the necessary file from www.activepdf.com/downloads/serverproducts/index.cfm.
3. On the taskbar, click the **Start** button, and then click **Run**.
4. In the **Open** box, type **X:\aptk40pe.exe** and then click **OK**.
5. Follow the on-screen instructions to complete the installation. If prompted, restart your computer.

Using activePDF Toolkit in .NET

activePDF Toolkit includes a .NET native dll (APToolkitNET.dll) that is used in implementing Toolkit in a .NET environment. There are a few important differences to be aware of when using activePDF Toolkit with .NET. This section details these differences as well as the procedure for properly implementing Toolkit in .NET.

NOTE: Refer to www.activePDF.com for information specific to implementing Toolkit in other development environments.

Namespace in .NET

When using activePDF Toolkit in .NET, the namespace changes from `APToolkit` to `APToolkitNET.[objectname]` (where `objectname` is Toolkit, Text2PDF, PDFFieldInfo, etc.). Refer to Technical Reference for detailed information on each object.

Instantiating the Objects

To instantiate the objects properly in .NET, a reference to the `APToolkitNET.dll` is required in the .NET scripting environment. If the reference to `APToolkitNET.dll` is incorrect or missing, the resultant script will cause a missing type or namespace error.

To add a reference to the APToolkitNET.dll

1. Start **Visual Studio® .NET**.
2. In the **Solution Explorer**, right-click **Reference**, and then select **Add Reference**.
3. In the **Add Reference** dialog, click **Browse**.
4. In the **Browse** field, type the location of the **APToolkitNET.dll** and then click **Open**. By default, this location is *X:\Program Files\dotNetComponents\APToolkitNET.dll*.

After referencing `APToolkitNET.dll` in the project, the Toolkit objects can be instantiated in your .NET script. The correct syntax for instantiating the objects depends on the specific .NET language you are using. The following sections provide the necessary steps for instantiating the objects in VB.NET or C#.

NOTE: Text2PDF, PDFFieldInfo, ListItems and Explorer are subobjects of the Toolkit object. Therefore, the Toolkit object must be instantiated before the Text2PDF, PDFFieldInfo, ListItems and Explorer objects.

To instantiate the objects in VB.NET

1. Add the following line to the beginning of your .NET code:

   ```
   Imports APToolkitNET
   ```
2. Instantiate the object, using the syntax that pertains to the object you are using:

   - **To instantiate the Toolkit object:**

   ```
   Dim TK As APToolkitNET.Toolkit
   TK = new APToolkitNET.Toolkit
   ```

   - **To instantiate the Text2PDF object:**

   ```
   Dim TK As APToolkitNET.Toolkit
   Dim T2P As APToolkitNET.Text2PDF
   TK = new APToolkitNET.Toolkit
   T2P = TK.Text2PDFObject
   ```

   - **To instantiate the PDFFieldInfo object:**

```
Dim TK As APToolkitNET.Toolkit
Dim FIO As APToolkitNET.PDFFieldInfo
TK = new APToolkitNET.Toolkit
' A valid PDF must be opened as input
TK.OpenInputFile("input.pdf") '
' Name and instance of field to get info for
FIO = TK.GetFieldInfo( "fieldname", 1)
```

- **To instantiate the ListItems object:**

```
Dim TK As APToolkitNET.Toolkit
Dim FIO As APToolkitNET.PDFFieldInfo
Dim LST As APToolkitNET.ListItems
TK = new APToolkitNET.Toolkit
' A valid PDF must be opened as input
TK.OpenInputFile("input.pdf") '
' Name and instance of field to get info for
FIO = TK.ListItems
```

- **To instantiate the Explorer object:**

```
Dim TK As APToolkitNET.Toolkit
Dim EXP As APToolkitNET.Explorer
TK = new APToolkitNET.Toolkit
EXP = TK.Explorer
```

- **To instantiate the Flash object:**

```
Dim FL As APToolkitNET.Flash
FL = new APToolkitNET.Flash
```

- **To instantiate the Barcode object:**

```
Dim BC As APToolkitNET.Barcode
BC = new APToolkitNET.Barcode
```

To instantiate the objects in C#

1. Add the following line to the beginning of your .NET code:

```
Using APToolkitNET
```

2. Instantiate the object, using the syntax that pertains to the object you are using:

- **To instantiate the Toolkit object:**

```
APToolkitNET.Toolkit TK = new APToolkitNET.Toolkit();
```

- **To instantiate the Text2PDF object:**

```
APToolkitNET.Toolkit TK = new APToolkitNET.Toolkit();
APToolkitNET.Text2PDF T2P = TK.Text2PDFObject();
```

- **To instantiate the PDFFieldInfo object:**

```
APToolkitNET.Toolkit TK = new APToolkitNET.Toolkit();
```

```
 // A valid PDF must be opened as input
r = TK.OpenInputFile("input.pdf");
// Name and instance of field
APToolkitNET.PDFFieldInfo FIO = TK.GetFieldInfo("fieldname", 1);
```

- **To instantiate the ListItems object:**

```
APToolkitNET.Toolkit TK = new APToolkitNET.Toolkit();
 // A valid PDF must be opened as input
r = TK.OpenInputFile("input.pdf");
// Name and instance of field
APToolkitNET.PDFFieldInfo FIO = TK.ListItems;
```

- **To instantiate the Explorer object:**

```
APToolkitNET.Toolkit TK = new APToolkitNET.Toolkit();
APToolkitNET.Exploer EXP = TK.Explorer();
```

- **To instantiate the Flash object:**

```
APToolkitNET.Flash FL = new APToolkitNET.Flash();
```

- **To instantiate the Barcode object:**

```
APToolkitNET.Barcode BC = new APToolkitNET.Barcode();
```

Properties and Methods specific to .NET

Most of Toolkit's properties and methods use the same syntax in .NET as documented in the Technical Reference section of this guide. However, there are a few exceptions, which are provided in the table below.

Existing Property/Method	Equivalent .NET Property/Method
ImageByteStream	ImageByteArray
InputByteStream	InputByteArray
CustomDocInfo	GetCustomDocInfo SetCustomDocInfo

Toolkit Font Usage

Certain methods, such asSetFont, enable you to specify the font used when performing operations such as adding fields, bookmarks or text to your PDF. You can specify one of the default fonts supported by Toolkit or any accessible font. If you specify font other than the default fonts, Toolkit must locate the font prior to using the font in your PDF.

For more information, refer to one of the following topics:

- Base 14 Fonts
- Double-byte Character Sets
- Specifying a Font

NOTE: For information on using fonts with Toolkit methods and fonts, refer to the Technical Reference section.

Base 14 Fonts

The SetFont method enables you to specify the font used when adding text to your PDF. You can specify one of the following fonts:

- Courier
- Courier Bold
- Courier Oblique
- Courier Bold-Oblique
- Helvetica
- Helvetica Bold
- Helvetica Oblique
- Helvetica Bold-Oblique
- Times Roman
- Times Bold
- Times Italic
- Times Bold-Italic
- Symbol
- Zapf Dingbats

NOTE: Toolkit supports these fonts by default, it is not necessary to supply the full path to the font using the SetFont method. For additional information, refer to the SetFont method.

Double-Byte Character Sets

Toolkit contains built-in Chinese, Japanese and Korean fonts as listed below.

Toolkit contains built-in Chinese, Japanese and Korean fonts as listed below. Additionally, you can specify double-byte characters per the guidelines in the Specifying a Font section.

NOTE: Font packs may be required to view a PDF created with double-byte fonts.

Built -in Chinese (Simplified) fonts:

- STSong-Light-Acro
- STSongStd-Light-Acro

Built-in Chinese (Traditional) fonts:

- MHei-Medium-Acro
- MSung-Light-Acro
- MSungStd-Light-Acro

Built-in Japanese fonts:

- HeiseiMin-W3-Acro
- HeiseiKakuGo-W5-Acro
- KozMinPro-Regular-Acro

Built-in Korean fonts:

- HYGoThic-Medium-Acro
- HYSMyeongJo-Medium-Acro
- HYSMyeongJoStd-Medium-Acro

NOTE: These are proportional width fonts with a default width of 1000 font units. For more information, refer to the SetFont method.

Specifying a Font

If you are not using one of the default fonts, Toolkit attempts to locate the font, using the specified name, in the following order:

1. **Input File:** Upon opening the input file, Toolkit caches the information contained in the PDF. When locating the specified font, Toolkit attempts to locate the first fully subset font instance in the input file cache.
2. **Windows registry:** Not finding the file in the input cache, Toolkit will attempt to locate the font information in the Windows registry.
3. **Derived Font:** If unable to locate the font in the Windows registry, Toolkit generates a substitute font from a similar named font. For example, if the font were set to Arial, Toolkit might generate a font named Arial based on a similar named font, such as ArialMT.
4. **Subset:** If all of the previous methods are unsuccessful, Toolkit will use the first instance of the partially subset font or similar font from the input cache.

NOTE: When a partially subset font is used, the output may be adversely affected. This can include missing characters, text or graphics, undesired formatting, styles, spacing and font usage.

If you would like to specify an exact font name and location, you can pass the name of a *TrueType Font* (TTF), *Open Type Font* (OTF) or *TrueType Collection* (TTC) located on your hard disk. Unless you specify the full path to the font, Toolkit will assume the font is located in the Windows fonts directory. (By default, the location of this directory is *X:\WINDOWS\Fonts*.)

NOTE: You cannot specify PostScript (PS) font names and locations. If you would like to use a PS font, you will need to include it in your input file.

Tutorials

The tutorials in this chapter demonstrate many of the common activePDF Toolkit functions. The following tutorials are provided:

Creating a PDF from Scratch

This tutorial provides two different examples for using Toolkit to generate a PDF from scratch. The PDF generated with this example can be used in other tutorials in this chapter. The first example creates a single page PDF and the second creates a multi-page document.

Example A

This example generates a blank PDF with the Helvetica font embedded.

Example Script

```
Set objTK = CreateObject("APToolkit.Object")

r = objTK.OpenOutputFile("Output1.pdf")
      objTK.SetFont "Helvetica", 15
      objTK.CloseOutputFile

Set objTK = Nothing
```

Example B

This example generates a 10-page PDF with "Your Company Name" printed on each page.

Example Script

```
Set objTK = CreateObject("APToolkit.Object")

r = objTK.OpenOutputFile("Output1.pdf")

For i = 1 to 10
```

```
        objTK.NewPage
        objTK.SetFont "Helvetica", 15
        objTK.PrintText 10, 20, "Your Company Name"

Next

objTK.CloseOutputFile

Set objTK = Nothing
```

Merge

This tutorial uses activePDF Toolkit to merge a 10 page PDF and a single page PDF into one PDF file. This tutorial contains two different examples for performing the same action, but the resultant PDF will be identical.

NOTE: With minor modifications, you can use the code supplied in this tutorial to perform Append and Extract operations.

Required for example(s)

- 10-paged PDF (Named: *Input1.pdf*).
- Single-paged PDF (Named: *Input2.pdf*).

Example A - CopyForm

This example uses the CopyForm method to merge two PDF documents.

Example Script

```
Set TK = CreateObject ("APToolkit.Object")

'Specify the file that will be generated
      R = TK.OpenOutputFile ("TKMerged-CopyForm.pdf")

'Open the first PDF to merge
      R = TK.OpeninputFile ("Input1.pdf")

'Copy Input1.pdf to the output file
      R = TK.copyform(0, 0)

'Close the InputFile
      R = TK.CloseInputFile()

'Open the second PDF to merge
      R = TK.OpenInputFile ("Input2.pdf")

'Copy Input2.pdf to the output file
      R = TK.copyform(0, 0)

'All done - Close the outputFile
      R = TK.CloseOutputFile()
```

```
   Set TK = Nothing
```

Example B - MergeFile

This example uses the MergeFile method to to merge two PDF documents.

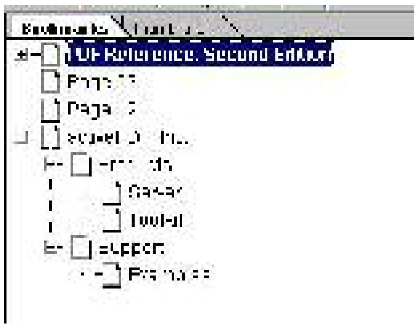
Example Script

```
   Set TK = CreateObject ("APToolkit.Object")

   'Specify the file that will be generated
         R = TK.OpenOutputFile ("TKMerged-MergeFile.pdf")

   'Use MergeFile (equivalent of OpenInputFile and CopyForm together)
         R = TK.MergeFile("Input1.pdf", 0, 0)
         R = TK.MergeFile("Input2.pdf", 0, 0)

   'All done close the outputFile
         R = TK.CloseOutputFile()

   Set TK = Nothing
```

Creating Bookmarks

This tutorial uses activePDF Toolkit to add bookmarks to an existing PDF document. Before you begin this tutorial, you will need a 130 page PDF (Named: *input1.pdf*), which you can generate by altering Example B in the Creating a PDF from Scratch tutorial.

The resultant PDF will contain a series of bookmarks similar to the image below:



Example Script

```
   Set pdf = CreateObject("APToolkit.Object")
   r = pdf.OpenOutputfile("G:\output.pdf")
   If (r <> 0) Then
     MsgBox "Unable to open output file!"
     Exit Sub
   End If

   pdf.AddBookmarks = True
```

```
r = pdf.MergeFile("g:\inputpdf.pdf", 0, 0)
pdf.AddInternalLinkBookmark "Page 59", 59, 0, 0
pdf.AddInternalLinkBookmark "Page 121", 121, 0, 0
pdf.AddURLBookmark "activePDF, Inc.", "http://www.activepdf.com"
pdf.GotoNextBookmarkLevel
pdf.AddTextBookmark "Products"
pdf.GotoNextBookmarkLevel
pdf.AddURLBookmark "Server",
"http://www.activepdf.com/products/serverproducts/server/index.cfm"
pdf.AddURLBookmark "Toolkit",
"http://www.activepdf.com/products/serverproducts/toolkit/index.cfm"
pdf.GotoPreviousBookmarkLevel
pdf.AddTextBookmark "Support"
pdf.GotoNextBookmarkLevel
pdf.AddURLBookmark "Examples",
"http://www.activepdf.com/support/troubleshooting/index.cfm"
pdf.CloseOutputFile
Set pdf = Nothing
```

Stamp

This tutorial uses Toolkit to stamp an image, company name and the page number onto every page of a multi-paged PDF document.

Required for example

- JPEG (Named: *Image1.jpg*).
- Multi-page PDF (Named: *Output1.pdf*).

Example Script

```
Set TK = CreateObject ("APToolkit.Object")

'Specify the file that will be generated
      R = TK.OpenOutputFile ("Output1.pdf")

'Count the number of pages
'Note - When using NumPages, OpenInputFile is not necessary
      strTotalPages = TK.NumPages("TKStamping.pdf")

'Font variables
      strFont = "Arial"
      strFontSizeText = "16"
      strFontSizePage = "12"

'Add the header image, Set header parameters
      imagefile = "Image1.jpg"
      x = 0    '72 = 1 inch, 0 ,0 is bottom left of page
      y = 20    '72 = 1 inch, 0 ,0 is bottom left of page
      width = 0    '0 for no change
      height = 0    '0 for no change
      PersistRatio = True
```

```
'Stamp image on the page
      TK.SetHeaderJPEG imagefile, x, y, width, height, True

'Set the font, location and text
      TK.SetHeaderFont strFont, strFontSizeText
      TK.SetHeaderText 15, 10, "Your Company Name"

'Set the PageNumber location and text
'Use GetTextWidth to stamp in the center of the page
      TK.SetHeaderFont strFont, strFontSizePage
      strPageNumberText = "Page %p of " & strTotalPages
      strPageNumberWidth = TK.GetHeaderTextWidth(strPageNumberText)
      xt = (612 - strPageNumberWidth) / 2
      TK.SetHeaderWPgNbr xt, 10, "Page %p of " & strTotalPages, 1

'Copy the current page to OutputFile
      R = TK.copyform(strPage, strPage)

'Clear the header info because we are stamping dynamic data
      TK.ClearHeaderInfo

'All done close the output file
      R = TK.CloseOutputFile()

Set TK = Nothing
```

Stamping - Page Specific

This tutorial uses Toolkit to stamp an image onto the fourth page of a ten-paged PDF document, leaving the other pages unchanged.

Required for example

- A JPEG (Named: *Image1.jpg*).
- 10 page PDF (Named: *Output1.pdf*).

Example Script

```
Set TK = CreateObject ("APToolkit.Object")

R = TK.OpenOutputFile ("Output1.pdf")
R = TK.OpenInputFile ("Input1.pdf")

'Copy the first 3 pages to the output
      R = TK.CopyForm(1, 3)

'Add the header image, set header parameters
      imagefile = "image1.jpg"
      x = 0 '72 = 1 inch, 0 ,0 is bottom left of page
      y = 0 '72 = 1 inch, 0 ,0 is bottom left of page
      width = 0 '0 for no change
      height = 0 '0 for no change
      PersistRatio = True
```

```
        TK.SetHeaderJPEG imagefile, x, y, width, height, True

    'Copy just the fourth page with the image to the output
        R = TK.copyform(4, 4)

    'Clear the header info so it does not appear on subsequent pages
        TK.ClearHeaderInfo

    'Copy page 5 and on to the output
        R = TK.copyform(5, 0)

    R = TK.CloseOutputFile()

    Set TK = Nothing
```

Stitch

This tutorial uses Toolkit to stitch a single PDF page 4-up on an 8.5 x 11 page of a new PDF. For this tutorial, you will need a single page PDF (Named: *input1.pdf*) containing a line of text, which you can generate by altering Example B in the Creating a PDF from Scratch tutorial.

Example Script

```
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("output1.pdf")
'Stitch the single page of the input file 4-up on a single page of the output PDF.
r = TK.StitchPDF("Input1.pdf", 1, 0, 397, 306, 396, 0)
r = TK.StitchPDF("Input1.pdf", 1, 307, 397, 306, 396, 0)
r = TK.StitchPDF("Input1.pdf", 1, 0, 0, 306, 396, 0)
r = TK.StitchPDF("Input1.pdf", 1, 307, 0, 306, 396, 0)

r = TK.CloseOutputFile()

Set TK = Nothing
```

Form Fields - Generating and Filling

Toolkit enables you to create PDF form fields on the fly or use an existing PDF document with form fields as a template. This tutorial provides two different examples for using Toolkit with PDF forms. The first example covers generating a PDF with form fields on the fly; the second example covers populating and flattening an existing form field.

Generate Form Fields on the Fly

This example uses Toolkit to generate a new PDF document with a single text field. The PDF generated in this example can be used in the other form field tutorials.

Example Script

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"

Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("Output1.pdf")
Set myField = TK.AddField(0, 1, "image", 10, 600, 200, 50, "Helvetica", 24)
myField.Value = "image"
TK.SetFont "Helvetica", 12
TK.PrintText 10, 580, "The name of this field is image"
TK.CloseOutputFile

Set TK = Nothing
```

Populate an Existing Form Field

This example uses Toolkit to place an image into a form field of template PDF and then flatten the form field. A template PDF is a PDF containing form fields that you can use as the input file to populate with activePDF Toolkit. The PDFs generated in the first tutorial are good examples of template PDFs.

Required for example

- JPEG (Named: *Image1.jpg*).
- Single page PDF (Named: *Input1.pdf*, containing a single text box field named: *Image1*).

Example Script

```
Set TK = CreateObject ("APToolkit.Object")

R = TK.OpenOutputFile ("TKSetFormfieldDataImageOutput.pdf")
R = TK.OpenInputFile ("Input1.pdf")

field = "image1"
image = "image1.jpg"
flag = -996 '-996 Flatten field using an image file as named in field data. The
image type is auto-determined. For more flags please consult the Toolkit
Documentation.

TK.SetFormFieldData field, image, -996

R = TK.copyform(0, 0)
R = TK.CloseOutputFile()

Set TK = Nothing
```

Creating a Barcode

This example uses Toolkit to generate and place a Code 39 barcode into a form field of template PDF and then flatten the form field. Before you begin this tutorial, you will need a single page PDF (Named: *input1.pdf*), which contains a single text box field (Named: *Image1*).

Example Script

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"

Set TK = CreateObject("APToolkit.Object")
Set barcode = CreateObject("APToolkit.Barcode")

barcode.Symbology = 0
barcode.BorderStyle = 0
barcode.SymbolMarginBottom = 0
barcode.SymbolMarginTop = 0
barcode.SymbolMarginRight = 0
barcode.SymbolMarginLeft = 0
barcode.Value = "This is the encoded information for the barcode"

r = TK.OpenOutputFile("BarcodeInField.pdf")
r = TK.OpenInputFile("Input1.pdf")

TK.SetFormFieldData "Image1", barcode.AsString, -996
r = TK.CopyForm(0, 0)

TK.CloseOutputFile

Set barcode = Nothing
Set TK = Nothing
```

In-Memory Generation

Toolkit enables you to generate PDFs "In-Memory". This enables you to create PDFs entirely in-memory without writing to disk, and the resultant output can be served directly to the client browser.

In this tutorial, you can use Toolkit with two separate examples. The first example will generate a PDF "In-Memory" and the second example will generate and serve the resultant PDF to the browser.

Example - In-Memory Generation

In this example, you will be using the "In-Memory" generation feature of activePDF Toolkit. The script below illustrates how to use an "In-Memory" input stream to create the resultant output PDF.

Example Script

```
Set objTK = CreateObject("APToolkit.Object")
objTK.OpenOutputFile ("MEMORY")
For i = 1 To 15
objTK.SetFont "Helvetica", 15
objTK.NewPage
Next
objTK.CloseOutputFile
' write this output to a variable
x = objTK.OutputBytestream()
r = objTK.OpenOutputFile("output.pdf")
' retrieve the output bytestream
objTK.InputByteStream = x
r = objTK.OpenInputFile("MEMORY")
```

```
objTK.SetHeaderTextColorCMYK 0, 100, 10, 0
'Let's load a font from disk
objTK.SetHeaderFont "Verdana Bold Italic", 20
objTK.SetHeaderText 300, 600, "activePDF Toolkit"
r = objTK.CopyForm(0, 0)
objTK.CloseOutputFile
Set objTK = Nothing
```

Example - Deliver Content to the Browser

This tutorial uses Toolkit to generate a PDF document "In-Memory" and deliver it to the browser.

NOTE: The example script is written in ASP.

Example Script

```
<%

'Tell ASP not to serve the page until entire page is processed
'Very Important
      response.buffer = True

Set objTK = Server.CreateObject("APToolkit.Object")

'Tell Toolkit to create the PDF in memory
      r = objTK.OpenOutputFile("MEMORY")

'SetFont will generate a new blank page and set the font to be used
'PrintText adds text to our new page
      objTK.SetFont "Helvetica", 15
      objTK.PrintText 15, 760, "activePDF Memory Example"

'Close our generated PDF
      objTK.CloseOutputFile

'Write the output to memory as a BinaryImage
      zz = objTK.binaryImage

'Tell the browser not to cache PDF
      response.expires = 0

'Clear response buffer
      response.Clear

'Tell browser what type of file it is opening
      response.ContentType = "application/pdf"
      response.AddHeader "Content-Type", "application/pdf"
      response.AddHeader "Content-Disposition", "inline;filename=Example.pdf"

'Write the PDF in memory to the browser
      response.BinaryWrite zz 'now let's write to the browser

Set objTK = Nothing

%>
```

Technical Reference

This section provides you with the necessary information to use activePDF Toolkit's objects and subobjects, and their related methods and properties. Each section contains a listing of the related methods and properties as well as instructions for instantiating or creating the relevant object.

Toolkit has the following objects and subobjects:

- Toolkit object
- PDFFieldInfo subobject
- ListItems subobject
- Text2PDF subobject
- Flash object
- Explorer subobject
- Barcode object

Toolkit

Many of activePDF Toolkit's common features are implemented using the various methods and properties of the Toolkit object.

This section includes the following:

- Instantiating the Toolkit object
- Methods
- Properties

Instantiating the Toolkit object

To instantiate the Toolkit object, use the following syntax:

```
Set TK = CreateObject("APToolkit.Object")
```

Methods

The Toolkit object has the following methods:

AddComment

Description

AddComment instructs Toolkit to add a note comment with a signifying icon. Used in conjunction, the icon alerts the reader's attention to the noted area for the signified purpose.

Return type

None

Syntax

object.**AddComment** *LLX, LLY, Width, Height, Contents, Name, NoteType, Flags, Color, Opened, PageNum*

The AddComment method has these required parts:

Part	Value Type	Description
LLX	Float	The horizontal position of the lower-left corner of the comment icon. Uses the PDF Coordinate System.
LLY	Float	The vertical position of the lower-left corner of the comment icon. Uses the PDF Coordinate System.
Width	Float	The width of the comment's popup window.
Height	Float	The height of the comment's popup window.
Contents	String	The contents of the comment's popup window.
Name	String	The comment's name, which appears in the lower half of the comment window's top bar.
Note Type	Long	The type of note comment you want to add to your PDF. This parameter controls which icon appears in the PDF. Values are: 0 = Comment. A comment appears as a speech or thought bubble. 1 = Key. A key tag appears as a key standing on point. 2 = Note. A note tag appears as a page with a folded lower-right corner. 3 = Help. A help tag appears as a question mark "?" in a circle. 4 = New paragraph. A new paragraph tag appears as the letters NP under a solid up arrow "^". 5 = Paragraph. A paragraph tag appears as the standard

		symbol ¶. 6 = Insert. An insert tag appears as a solid up arrow "^"
Flags	Long	Controls the behavior of the comment. Values are: 2 = Hidden (no view OR print) 4 = Print 8 = No zooming. Zooming does not affect appearance. 16 = No rotate. Do not rotate the appearance to match the page's rotation. 32 = No View. Hidden on screen, but printable. 64 = Readonly. Contents cannot be changed. 128 = Locked. The comment properties cannot be deleted, but contents can be changed. 256 = ToggleNoView. This is largely implementation dependent. Values can be "or'ed" together: Flags = 4 or 64.
Color	String	The color of the comment you want to add to your PDF. PDF comment color (controls color of the comment icon and top bar of popup window.) Alternately, you can use hex codes. **NOTE:** Hex codes must be pre-pended with the pound sign (#) e.g. #FF0000. Refer to Appendix F: Supported Comment Colors for a list of values.
Opened	Variant_Bool	Sets the initial view setting of the comment to "open" or "closed". True = Open - Icon and comment displayed. False = Closed - Icon only.
PageNum	Long	The page on which the comment appears. If you specify 0, Toolkit adds the comment to the cover page.

Example

```
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenOutputFile("output.pdf")
r = TK.OpenInputFile("input.pdf")
TK.AddComment 30, 562, 200, 200, "This is a test comment", "Test", 2, 16, "#006600",
false, 1
TK.CopyForm 0, 0
TK.CloseOutputFile
Set TK = Nothing
```

AddExternalLink

Description

AddExternalLink instructs Toolkit to add a link in the current output file that connects to a specified designation in an external PDF document.

Return type

None

Syntax

object.**AddExternalLink** *PageNbr, LLX, LLY, URX, URY, DestFileName, DestPage, DestLLX, DestLLY, Style*

The AddExternalLink method has these required parts:

Part	Value Type	Description
Object	None	An expression of the Toolkit object.
PageNbr	Long	The page in the new PDF. (Use -1 when adding links during CopyForm or MergeFile operations.)
LLX	Short	The horizontal position of the lower-left corner of the link. Uses the PDF Coordinate System.
LLY	Short	The vertical position of the lower-left corner of the link. Uses the PDF Coordinate System.
URX	Short	The horizontal position of the upper-right corner of the link. Uses the PDF Coordinate System.
URY	Short	The vertical position of the upper-right corner of the link. Uses the PDF Coordinate System.
DestFileName	String	The full path to the external PDF file.
DestPage	Long	The page number in the external PDF file containing the destination.
DestLLX	Short	The horizontal position for the linked destination's lower-left corner in the external PDF file. Uses the PDF Coordinate System.
DestLLY	Short	The vertical position for the linked destination's lower-left corner in the external PDF file. Uses the PDF Coordinate System.

Style	Short	Box style of the link. Values are: -1 = Invisible 0 = Black solid 1 = Red dashed 2 = Red solid 3 = Green dashed 4 = Green solid 5 = Blue dashed 6 = Blue solid

AddExternalLinkBookmark

Description

AddExternalLinkBookmark instructs Toolkit to create a bookmark in the current output file that connects to a specified designation in an external PDF document.

Return type

None

Syntax

object.**AddExternalLinkBookmark** *BookmarkText, DestFileName, DestPage, DestLLX, DestLLY*

The AddExternalLinkBookmark method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
BookmarkText	String	The text to add as a bookmark.
DestFileName	String	The full path to the external PDF file.
DestPage	Long	The destination page number in the external PDF file.
DestLLX	Short	The horizontal position for the linked destination's lower-left corner in the external PDF file. Uses the PDF Coordinate System.
DestLLY	Short	The vertical position for the linked destination's lower-left corner in the external PDF file. Uses the PDF Coordinate System.

Remarks

You can generate bookmarks after adding a page. By default, Toolkit generates bookmarks at the highest level of the topic tree. You can call GotoNextBookmarkLevel or GotoPreviousBookmarkLevel prior to any bookmark method to control its level in the tree.

AddField

Description

AddField instructs Toolkit to add one of several types of form fields to your PDF at the specified location. You can also control the field name, height and width, and the font name and size.

NOTE: Toolkit writes the form fields to the PDF when you call CloseOutputFile.

Syntax

object.**AddField** *PageNumber, FieldType, FieldName, LLX, LLY, Width, Height, FontName, FontSize*

The AddField method has these required parts:

Part	Value Type	Description
PageNumber	Int	The specified page number in the output PDF on which Toolkit will add the form field.
FieldType	Int	The type of form field Toolkit will create in the output PDF. Values are: 1 = Text field 2 = Signature 3 = Push button 4 = Checkbox 5 = Radio button 6 = Combo box 7 = List box
FieldName	String	The name of the form field.
LLX	Float	The horizontal position of the lower-left corner of the form field. Uses thePDF Coordinate System.
LLY	Float	The vertical position of the lower-left corner of the form field. Uses the PDF Coordinate System.
Width	Float	The width of the form field.
Height	Float	The height of the form field.
FontName	String	The font used in the form field. **NOTE:** You must set the encoding for the font. Refer to the SetFont method for details and a list of appropriate parameters.

FontSize	Float	The size of the font. Refer to SetFont for details.

AddHyperlink

Description

AddHyperlink instructs Toolkit to add a hyperlink in the current output file that connects to a specified URL. When clicked, the URL opens in a new browser window.

Return type

None

Syntax

object.**AddHyperlink** *PageNbr, LLX, LLY, URX, URY, DestURL, Style*

The AddHyperlink method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
PageNbr	Long	0 = The action will take place on the new or current open output page. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.
LLX	Short	The horizontal position of the lower-left corner of the link. Uses the PDF Coordinate System.
LLY	Short	The vertical position of the lower-left corner of the link. Uses the PDF Coordinate System.
URX	Short	The horizontal position of the upper-right corner of the link. Uses the PDF Coordinate System.
URY	Short	The vertical position of the upper-right corner of the link. Uses the PDF Coordinate System.
DestURL	String	The full path to the destination URL.
Style	Short	Box style of the link. -1 = Invisible 0 = Black solid 1 = Red dashed 2 = Red solid 3 = Green dashed 4 = Green solid

		5 = Blue dashed 6 = Blue solid

Example

```
'AddHyperlink Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("AddHyperlink.pdf")

      'Set the font and color for our visable text
      strFontSize = 20
      TK.SetFont "Helvetica", strFontSize, 0
      TK.SetTextColor 0, 0, 255, 0, 0

      'Get the width of the text so we know how wide to make the link
      strLinkText = "www.activePDF.com"
      strTextWidth = TK.GetTextWidth(strLinkText, 0)

      'Print the text that will show for the link
      strLLX = 30
      strLLY = 740
      TK.PrintText strLLX, strLLY, strLinkText, 0

      'Add the Hypderlink over the recently placed text
      strURL = "http://www.activepdf.com"
      TK.AddHyperlink 0, strLLX, strLLY, strLLX + strTextWidth, strLLY +
strFontSize, strURL, 0

TK.CloseOutputFile

Set TK = Nothing
```

AddInternalLink

Description

AddInternalLink instructs Toolkit to add a link that connects to a specified designation in PDF document in the current output file.

Return type

None

Syntax

object.**AddInternalLink** *PageNbr, LLX, LLY, URX, URY, DestPage, DestLLX, DestLLY, Style*

The AddInternalLink method has these required parts:

Part	**Value Type**	**Description**
Object		An expression of the Toolkit object.
PageNbr	Long	The page to place the hyperlink on the outputted PDF. (Use -1 when adding links during CopyForm or MergeFile operations.)
LLX	Short	The horizontal position of the lower-left corner of the link. Uses the PDF Coordinate System.
LLY	Short	The vertical position of the lower-left corner of the link. Uses the PDF Coordinate System.
URX	Short	The horizontal position of the upper-right corner of the link. Uses the PDF Coordinate System.
URY	Short	The vertical position of the upper-right corner of the link. Uses the PDF Coordinate System.
DestPage	Long	The page number in the current output PDF containing the destination.
DestLLX	Short	The horizontal position for the linked destination's lower-left corner. Uses the PDF Coordinate System.
DestLLY	Short	The vertical position for the linked destination's lower-left corner. Uses the PDF Coordinate System.
Style	Short	Box style of the link. -1 = Invisible 0 = Black solid 1 = Red dashed

		2 = Red solid 3 = Green dashed 4 = Green solid 5 = Blue dashed 6 = Blue solid

AddInternalLinkBookmark

Description

AddInternalLinkBookmark instructs Toolkit to create a bookmark in the current output file that connects to a specified internal designation in PDF.

Return type

None

Syntax

object.**AddInternalLinkBookmark** *BookmarkText, DestPage, DestLLX, DestLLY*

The AddInternalLinkBookmark method has these required parts:

Part	**Value Type**	**Description**
Object		An expression of the Toolkit object.
BookmarkText	String	The text that appears as the bookmark name.
DestPage	Long	The page number containing the destination for the bookmark.
DestLLX	Short	The horizontal position for the linked destination's lower-left corner. Uses the PDF Coordinate System.
DestLLY	Short	The vertical position for the linked destination's lower-left corner. Uses the PDF Coordinate System.

Remarks

You can generate bookmarks after adding a page. By default, Toolkit generates bookmarks at the highest level of the topic tree. You can call GotoNextBookmarkLevel or GotoPreviousBookmarkLevel prior to any bookmark method to control its level in the tree.

AddLaunchBookmark

Description

AddLaunchBookmark instructs Toolkit to create a bookmark in the current output file that executes a command in the OS shell.

Return type

None

Syntax

*object.***AddLaunchBookmark** *BookmarkText, Command*

The AddLaunchBookmark method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
BookmarkText	String	The text that appears as the bookmark name.
Command	String	The specified Windows or Mac that executes when clicking the bookmark. Refer to the related OS documentation for proper command instructions.

Remarks

You can generate bookmarks after adding a page. By default, Toolkit generates bookmarks at the highest level of the topic tree. You can call GotoNextBookmarkLevel or GotoPreviousBookmarkLevel prior to any bookmark method to control its level in the tree.

AddLaunchLink

Description

AddLaunchLink instructs Toolkit to add a hyperlink in the current output file executes a command in the OS shell.

Return type

None

Syntax

object.**AddLaunchLink** *PageNbr, LLX, LLY, URX, URY, Command, Style*

The AddLaunchLink method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
PageNbr	Long	The page of the PDF on which Toolkit adds the link. Use -1 when adding links during CopyForm or MergeFile operations.
LLX	Short	The horizontal position of the lower-left corner of the link. Uses the PDF Coordinate System.
LLY	Short	The vertical position of the lower-left corner of the link. Uses the PDF Coordinate System.
URX	Short	The horizontal position of the upper-right corner of the link. Uses the PDF Coordinate System.
URY	Short	The vertical position of the upper-right corner of the link. Uses the PDF Coordinate System.
Command	String	The specified Windows or Mac that executes when clicking the bookmark. Refer to the related OS documentation for proper command instructions.
Style	Short	Box style of the link. -1 = Invisible 0 = Black solid 1 = Red dashed 2 = Red solid 3 = Green dashed 4 = Green solid

		5 = Blue dashed 6 = Blue solid

AddLogo

Description

A PDF Logo is a full size image of a PDF page, used as a watermark. AddLogo instructs Toolkit to add an existing PDF as watermark in the foreground or background of all pages affected by subsequent to CopyForm and MergeFile. You can call AddLogo prior to PrintLogo to place the logo on a new page.

Return type

Long

Return Value	Description
-1	Unable to add logo.
>0	Logo added successfully. This is the logo index number.

Syntax

object.**AddLogo** *LogoFileName, Background*

The AddLogo method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
LogoFileName	String	The full path to the PDF file of which Toolkit creates a PDF Logo.
Background	Short	1= Add PDF Logo in background. 0 = Add PDF Logo in foreground. (Default.) **NOTE:** If the pages merged from an existing PDF are not transparent, the merged page covers a PDF Logo placed in the background. Similarly, if the source page for the PDF Logo is not transparent, the PDF Logo placed in the foreground covers the page underneath.

Remarks

Toolkit aligns the lower-left corner (0,0) of the PDF Logo to the lower-left corner (0,0) of the new or existing page. If the new or existing page is smaller than the PDF Logo, the image will bleed past the top and right edges.

Toolkit uses a predefined identifier, /Xqx, for the corresponding XObject inside the PDF file. As a result, you can apply the same PDF Logo to multiple PDF documents, but you cannot apply a PDF Logo to a PDF that already contains a PDF Logo.

AddPostScriptComment

Description

A PostScript (PS) comment is a command added to the PS stream of the PDF, which executes upon exporting to PS or printing the document. AddPostScriptComment instructs Toolkit to add a PS comment to the current output PDF.

Return type

None

Syntax

object.**AddPostscriptComment** *PostscriptComment*

The AddPostscriptComment method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
PostscriptComment	String	The PS stream to insert into the output PDF.

AddRelatedQuery

Description

The organized structure of a database enables the easy request of information, or query, based on the columns and rows. AddRelatedQuery instructs Toolkit to perform the master query specified with SetMasterQuery again, replacing the original variable with the related variable. Toolkit appends the data to the original query.

Return type

None

Syntax

object.**AddRelatedQuery** *ConnectionString, UserID, Password, Options, QueryString, MultiRows*

The AddRelatedQuery method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
ConnectionString	String	The connection string used to connect to the database. For example connection strings, see below.
UserID	String	The User ID required by your connection.
Password	String	The password required by your connection.
Options	Long	This should be set to -1.
QueryString	String	The SQL query string to execute.
MultiRows	Long	Indicates whether the related query is a one-to-many query.

Connection Strings

The following are some common examples of connection string values.

ConnectionString	Example Value
Using a named DSN	"DSN=MyDatabase;"
Microsoft Access using the ODBC	Driver"DBQ=C:\InetPub\database\donations.mdb;Driver={Microsoft Access Driver(*.mdb)};"
Microsoft Access using the	"Provider=Microsoft.Jet.OLEDB.4.0;Data Source

Access OLEDB driver	C:\InetPub\database\donations.mdb;"
SQL Server using the ODBC driver	"Driver={SQL Server};Server=activePDF;Database= pubs;Uid=sa; Pwd=;"
SQL Server using the OLEDB driver	"PROVIDER=SQLOLEDB;DATA SOURCE=ServerName;DATABASE= pubs;USER ID=sa;PASSWORD=;"

Remarks

To specify the master query variable to replace, you will need to enclose the related variable in pipe characters ("|") or the RelatedQuerySeparator . For example, if your master query selects an OrderID from the sales table and your related query selects the description of the order from the sales table, you might use the following script:

```
Select * from OrderDetails where OrderId='|OrderID|'
```

If you want to cancel all queries, you can call ClearQueries.

Example CF

```
Public Sub ExampleCodeForAddRelatedQuery()
<CFSET TK.AddRelatedQuery "DBQ=C:\InetPub\database\donations.mdb;Driver={Microsoft
Access Driver (*.mdb)};", "admin", "", -1, "Select * from OrderDetails where
OrderID='|OrderID|'">
End Sub
```

AddTextBookmark

Description

AddTextBookmark instructs Toolkit to create a text-only bookmark in the current output file. This is useful for grouping multiple bookmarks on a similar topic or categorizing bookmarks into more manageable sections. For example, if you have multiple bookmarks relating to topic "*A*", you can create a higher-level bookmark named "*Topic A*". The lower-level bookmarks under "*Topic A*" are only viewable in the bookmark window when you click the text-only bookmark to expand the topic.

Return type

None

Syntax

object.**AddTextBookmark** *BookmarkText*

The AddTextBookmark method has these required parts:

Part	**Value Type**	**Description**
Object		An expression of the Toolkit object.
BookmarkText	String	The text to add as a bookmark.

Remarks

You can generate bookmarks after adding a page. By default, Toolkit generates bookmarks at the highest level of the topic tree. You can call GotoNextBookmarkLevel or GotoPreviousBookmarkLevel prior to any bookmark method to control its level in the tree.

AddToStream

Description

AddToStream instructs Toolkit to add additional bytes the existing PDF Stream.

NOTE: This method requires a strong understanding of PDF code. activePDF does not support functionality or features implemented by adding PDF code.

Return type

Long

Syntax

object.**AddToStream** *Data, PageNr*

The AddToStream method has these required parts:

Part	Value Type	Description
Data	String	The PDF code added to the stream.
PageNr	Long	Optional. 0 = The action will take place on the new or current open page of the input file or cover. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.

AddURLBookmark

Description

AddURLBookmark instructs Toolkit to create a bookmark in the current output file that connects to an external URL.

Return type

None

Syntax

object.**AddURLBookmark** *BookmarkText, URL*

The AddURLBookmark method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
BookmarkText	String	The text to add as a bookmark.
URL	String	The destination URL specified as the endpoint of the link.

Remarks

You can generate bookmarks after adding a page. By default, Toolkit generates bookmarks at the highest level of the topic tree. You can call GotoNextBookmarkLevel or GotoPreviousBookmarkLevel prior to any bookmark method to control its level in the tree.

BinaryImage

Description

Returns the PDF as an array of bytes suitable for `Response.BinaryWrite`.

Return type

Variant

Description
The specified PDF as an array of bytes.

Syntax

*value = object.***BinaryImage**

The BinaryImage method has this part:

Part	Description
Object	An expression of the Toolkit object.

ClearHeaderInfo

Description

Clears the variables used for setting the attributes that are used in SetHeader methods. Also clears the content stream that paints logos or images without deleting the logo or image from memory.

Return type

None

Syntax

`object`.**`ClearHeaderInfo`**

The ClearHeaderInfo method has this part:

Part	Description
Object	An expression of the Toolkit object.

ClearLogosAndImages

Description

ClearLogosAndImages clears the variables created by the Image and Logo methods without clearing the content stream.

Return type

None

Syntax

`object`.**`ClearLogosAndImages`**

The ClearLogosAndImages method has this part:

Part	Description
Object	An expression of the Toolkit object.

ClearQueries

Description

Clears and closes the connections made during calls to SetMasterQuery and AddRelatedQuery.

Return type

None

Syntax

object.**ClearQueries**

The ClearQueries method has this part:

Part	Description
Object	An expression of the Toolkit object.

CloseInputFile

Description

CloseInputFile instructs Toolkit to close the currently open input file.

Return type

None

Syntax

`object`.**`CloseInputFile`**

The CloseInputFile method has this part:

Part	Description
Object	An expression of the Toolkit object.

CloseOutputFile

Description

CloseOutputFile instructs Toolkit to close the currently open output file. If an input file is currently open, Toolkit closes this file as well.

NOTE: You should only use CloseOutputFile after you make all desired changes to the output file. Any additional changes will require you to reopen the file.

Return type

None

Syntax

object.**CloseOutputFile**

The CloseOutputFile method has this part:

Part	Description
Object	An expression of the Toolkit object.

Example

```
'CloseOutputFile Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("CloseOutputFile.pdf")

      'Set the font for the text
      TK.SetFont "Helvetica", 20, 0

      'Stamp Text onto the page
      TK.PrintText 30, 740, "Hello World", 0

'Close Output File we are done creating the PDF
TK.CloseOutputFile

Set TK = Nothing
```

CopyForm

Description

Copies the specified range of pages from the current open input file to the output file.

Return type

Long

Return Value	Description
0	Failure /Incorrect path.
-998	Product not registered/ Evaluation expired.
1	Success.

Syntax

object.**CopyForm** *FirstPage, LastPage*

The CopyForm method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FirstPage	Long	The first page of the input PDF to copy. (If set to 0, will default to the first page.)
LastPage	Long	The last page of the input PDF to copy. (If set to 0, will default to all pages.)

Remarks

If the file contains form fields, the field data is set according to previous SetFormFieldData calls. You can repeatedly call CopyForm and change the header in between calls. If you do so, make sure that you call ResetFormFields to clear out any data previously set with SetFormFieldData. Copy all pages containing forms of an input file exactly once. Leaving out a page with form fields may result in orphan entries.

CountFormFields

Description

Returns the number of form fields in the currently open input file.

Return type

Short

Description
The number of form fields in the currently open input file.

Syntax

*value = object.***CountFormFields**

The CountFormFields method has this part:

Part	Description
Object	An expression of the Toolkit object.

CreateCertificate

Description

Creates a certificate, optionally signs it with another certificate, and inserts it into the Windows registry.

Return type

Long

Return Value	Description
-998	Professional version not registered.
-997	Product not registered/ Evaluation expired.
1	Success.
> 1	A specific error number used by the activePDF Technical Support Team.

Syntax

object.**CreateCertificate** *CommonName, OrgUnit, Org, Locale, State, Country, EMail, UseLocalMachine, CertStoreName, DaysCertIsValid, IssuerUseLocalMachine, IssuerName, IssuerStoreName*

The CreateCertificate method has these required parts:

Part	Value Type	Description
CommonName	String	The common name of the certificate.
OrgUnit	String	The organizational unit (e.g. Sales).
Org	String	The organization (e.g. activePDF).
Locale	Long	The city.
State	String	The state.
Country	String	The country.
EMail	String	The email address.
UseLocalMachine	Long	If set to 1 then the certificate will be stored under HKEY_LOCAL_MACHINE. If set to 0 then the certificate will be stored under

		HKEY_CURRENT_USER. **NOTE:** For web applications, we recommend setting this variable to 1.
CertStoreName	String	The certificate store name (e.g. "My" or "SelfSignedCertificates").
DaysCertIsValid	Short	The number of days the certificate is valid from the current date.
IssuerUseLocalMachine	Long	If set to 1 then the issuer certificate will be read from HKEY_LOCAL_MACHINE. If set to 0 then the issuer certificate will be read from HKEY_CURRENT_USER. **NOTE:** For web applications, we recommend setting this variable to 1. Set to 0 if not signing with another certificate.
IssuerName	String	The common name of the signing certificate. If not signing with another certificate set to "".
IssuerStoreName	String	The certificate store where the signing certificate is stored. If not signing with another certificate set to "".

Example

```
Set tk = CreateObject("APToolkit.Object")
z = TK.CreateCertificate("Joe Kant (Signed)", "Management", "activePDF","Mission
Viejo", "CA", "US", "joe@activepdf.com", 1, "My", 365,1,"activePDF, Inc.", "My")
z = TK.CreateCertificate("Joe Kant (Self-Signed)", "Management",
"activePDF","Mission Viejo", "CA", "US", "joe@activepdf.com", 1, "My", 365,0,"", "")
set TK = nothing
```

DBToForm

Description

Merges data contained in a database to the specified PDF form field.

NOTE: You must call SetMasterQuery and AddRelatedQuery prior to calling this method.

Return type

Long

Return Value	Description
0	Success.
-998	Product not registered/ Evaluation expired.
-2	No master query is set.
-1	Unable to open input file.
>0	A number relating to a specific Windows API error.

Syntax

`object``.`**`DBToForm`** *`MultiPage`*

The DBToForm method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
MultiPage	Long	1 = Generates a PDF page for each record set entry. 0 = Generates a new PDF page for the first entry in the record. (default)

Remarks

You can specify multiple pages to generate from the query.

DecryptFile

Description

Explicitly decrypts an encrypted PDF.

Return type

Long

Return Value	Description
-998	Product not registered/ Evaluation expired.
-997	Required product version not registered.
-99	One or more passwords are invalid.
-98	Input file is not encrypted.
-3	Unable to access output file.
-2	Unable to open input file.
-1	Unable to generate output file.
0	Success.

Syntax

*object.***DecryptFile** *InputFileName, OutputFileName, UserPassword, OwnerPassword*

The DecryptFile method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
InputFileName	String	The input filename.
OutputFileName	String	The output filename.
UserPassword	String	The case-sensitive password required to open the document.
OwnerPassword	String	The case-sensitive password required to modify or print the document.

DeleteFile

Description

Deletes a file from the hard disk.

Return type

Long

Return Value	Description
1	Success.
0	Failure.
>0	A number relating to a specific Windows API error.

Syntax

object.**DeleteFile** *FileName*

The DeleteFile method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FileName	String	The file to delete (Must be specified as an absolute path.)

Example

```
'DeleteFile Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("DeleteFile.pdf")

      'Set the font for the text
      TK.SetFont "Helvetica", 20, 0

      'Stamp Text onto the page
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile
If r = "0" Then
      msgbox "PDF created successfully, PDF will now be deleted"
End If

'Delete the new created PDF
r = TK.DeleteFile("DeleteFile.pdf")
```

```
If r <> 1 Then
      msgbox "Failed to delete file: " & r
End If

Set TK = Nothing
```

DeleteFormField

Description

Removes a form field from the input file during the next call to CopyForm.

Return type

None

Syntax

object.**DeleteFormField** *FieldName*

The DeleteFormField method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FieldName	String	The form field to delete.

DrawTo

Description

DrawTo specifies an endpoint that corresponds to the start point specified with the MoveTo method. Once you specify an endpoint, DrawTo instructs Toolkit to draw a line from the start point to the endpoint on a new or existing PDF page. The LineWidth method determines the width of the line.

Return type

None

Syntax

object.**DrawTo** *EndX, EndY, PageNr*

The DrawTo method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
EndX	Float	The horizontal position for the endpoint of the line. If you specify a line width, EndX corresponds to the horizontal position for the endpoint of the line that is equidistant from the upper and lower edges of the line. Uses the PDF Coordinate System.
EndY	Float	The vertical position for the endpoint of the line. If you specify a line width, EndY corresponds to the vertical position for the endpoint of the line that is equidistant from the upper and lower edges of the line. Uses the PDF Coordinate System.
PageNr	Long	Optional. 0 = The action will take place on the new or current open output page. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.

Example

```
'DrawTo Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("DrawTo.pdf")
```

```
      'Draw a border around the PDF
      strPageWidth = 612 '8.5" (72 = 1")
      strPageHeight = 792 '11" (72 = 1")
      strSpace = 20 'Space between edge of page and border

      'Top of the page line
      TK.MoveTo strSpace, strPageHeight - strSpace, 0
      TK.DrawTo strPageWidth - strSpace, strPageHeight - strSpace, 0

      'Left of the page line
      TK.MoveTo strPageWidth - strSpace, strPageHeight - strSpace, 0
      TK.DrawTo strPageWidth - strSpace, strSpace, 0

      'Bottom of the page line
      TK.MoveTo strSpace, strSpace, 0
      TK.DrawTo strPageWidth - strSpace, strSpace, 0

      'Right of the page line
      TK.MoveTo strSpace, strSpace, 0
      TK.DrawTo strSpace, strPageHeight - strSpace, 0

TK.CloseOutputFile

Set TK = Nothing
```

EmbedFlashFile

Description

Embeds a Flash® file into the PDF document at the location specified.

Return type

Long

Syntax

object.**EmbedFlashFile** *FileName, LLX, LLY, Width, Height, RenditionName, AnnotName, Flags, params, PageNum*

The EmbedFlashFile method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FileName	String	The full path to the Flash file to be embedded. Can use MEMORY, in conjunction with ImageByteStream.
LLX	Float	The horizontal position of the lower-left corner of the Flash window. Uses thePDF Coordinate System.
LLY	Float	The vertical position of the lower-left corner of the Flash window. Uses the PDF Coordinate System.
Width	Float	The width of the Flash window. Specified in PDF Units.
Height	Float	The height of the Flash window. Specified in PDF Units.
RenditionName	String	The name of the Flash rendition used for JavaScript purposes.
AnnotName	String	The name of the annotation used for uniqueness, accessibility and access via JavaScript.
Flags	Long	A series of flags that can be combined via "or" statements: 0 = Read Only. 1 = "As is". All attributes of the field remain unchanged. 2 = Hidden. 4 = Enable Printing. 8 = Disable Zoom. 16 = Disable Rotation.

		32 = The movie will print, but cannot be viewed. 64 = The movie will be hidden and read only.
params	String	Can be separated by a semi-colon. Values are: loop = 0 to n where 0 = continuous. Default is 0. playcommand = indicates when to start playing the Flash file using the following parameters: PO - page open (default). PC - page closed. PV - Page visible. PI - Page invisible. D - Mouse down. U - Mouse up. example :"loop=0;playcommand=D"
PageNum	Long	The page to which the Flash file is to be embedded.

EncryptFile

Description

Explicitly encrypts an unencrypted PDF file using 40-bit cipher strength.

NOTE: If you are calling LinearizeFile, you must call EncryptFile first or the linearization will be removed.

Return type

Long

Return Value	Description
-998	Product not registered/ Evaluation expired.
-997	Required product version not registered.
-1	Unable to generate output file.
-2	Unable to open input file.
-3	Unable to access output file.
0	Success.

Syntax

object.**EncryptFile** *InputFileName, OutputFileName, UserPassword, OwnerPassword, CanPrint, CanEdit, CanCopy,CanModify*

The EncryptFile method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
InputFileName	String	The input filename.
OutputFileName	String	The output filename.
UserPassword	String	Case-sensitive password required to view the document. The maximum length for the password is 32 characters and cannot contain control characters. (If you are using the evaluation version of activePDF Toolkit, the prefix DEMO will be inserted before your password characters and count towards the 32-character maximum. For

		example, the password TEST will be DEMOTEST.)
OwnerPassword	String	Case-sensitive password required to modify or print document. The maximum length for the password is 32 characters and cannot contain control characters. The password will default to the UserPassword if left blank. (If you are using the evaluation version of activePDF Toolkit, the prefix DEMO will be inserted before your password characters and count towards the 32-character maximum. For example, the password TEST will be DEMOTEST.)
CanPrint	Long	Set to 1 to enable printing. Set to 0 to disable printing.
CanEdit	Long	Set to 1 to enable document editing. Set to 0 to disable document editing.
Can Copy	Long	Set to 1 to enable copying of text and graphics. Set to 0 to disable copying of text and graphics.
Can Modify	Long	Set to 1 to enable document modification. Set to 0 to disable document modification.

EncryptFile128

Description

Explicitly encrypts an unencrypted PDF file using 128-bit cipher strength.

NOTE: If you are calling LinearizeFile, you must call EncryptFile128 first or the linearization will be removed.

Return type

Long

Return Value	Description
-998	Product not registered/ Evaluation expired.
-997	Required product version not registered.
-1	Unable to generate output file.
-2	Unable to open input file.
-3	Unable to access output file.
0	Success.

Syntax

object.**EncryptFile128** *InputFileName, OutputFileName, UserPassword, OwnerPassword, CanPrint, CanEdit, CanCopy,CanModify, CanFillInFormFields, CanMakeAccessible, CanAssemble, CanReproduce*

The EncryptFile128 method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
InputFileName	String	The input filename.
OutputFileName	String	The output filename.
UserPassword	String	Case-sensitive password required to view document. The maximum length for the password is 32 characters and cannot contain control characters. Once the password is set, it cannot be changed. (If you are using the evaluation version of activePDF Toolkit, the prefix DEMO will be inserted before your password characters

		and count towards the 32-character maximum. For example, the password TEST will be DEMOTEST.)
OwnerPassword	String	Case-sensitive password required to modify or print document. The maximum length for the password is 32 characters and cannot contain control characters. The password will default to the UserPassword if left blank. Once the password is set, it cannot be changed. (If you are using the evaluation version of activePDF Toolkit, the prefix DEMO will be inserted before your password characters and count towards the 32-character maximum. For example, the password TEST will be DEMOTEST.)
CanPrint	Long	Set to 1 to enable printing. Set to 0 to disable printing.
CanEdit	Long	Set to 1 to enable editing. Set to 0 to disable editing.
CanCopy	Long	Set to 1 to enable copying of text and graphics. Set to 0 to disable copying of text and graphics.
CanModify	Long	Set to 1 to enable document modifications. Set to 0 to disable document modification.
CanFillInFormFields	Long	Set to 1 to enable form field filling. Set to 0 to disable form field filling.
CanMakeAccessible	Long	Set to 1 to enable accessibility features. Set to 0 to disable accessibility features.
CanAssemble	Long	Set to 1 on an encrypted document to enable the user to insert, rotate or delete pages, and generate bookmarks or thumbnails even if CanModify is false. Set to 0 to disable document assembly.
CanReproduce	Long	Set to 1 on an encrypted document to enable the user to print a faithful reproduction of the PDF. Set to 0 to disable document reproduction. If this flag is 0 and CanPrint is 1, printing is limited to a low-resolution version.

ExportComments

Description

Enables you to export PDF comments from a specified page, returning the comments as a string value.

Return type

String

Description
XML strings containing the PDF comment data. You can use this string with the AddComment method.

Syntax

value = *object.***ExportComments** *PgNum*

The ExportComments method has these required parts:

Part	Value Type	Description
PgNum	Long	The number of the page containing the comments you want to export. Specify 0 (zero) to return all comments on all pages.

Remarks

If you specify 0 (zero) to return all comments on all pages, they are retrieved in a single string. If you need the page number for the comments, we recommend using a process similar to the example below.

Example

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"
Set TK = CreateObject("APToolkit.Object")
numPgs = TK.Numpages("output.pdf")
Dim filesys, testfile
Set filesys = CreateObject("Scripting.FileSystemObject")
Set testfile= filesys.CreateTextFile(strPath & "comments.txt", True)
For I = 1 to numPgs
      r = TK.ExportComments(I)
      testfile.WriteLine "Comments From Page " & I
      testfile.WriteLine ""
      testfile.WriteLine r
      testfile.WriteLine ""
Next
testfile.Close
TK.CloseInputFile
Set TK = Nothing
```

ExportFormAsXML

Description

Enables you to export data from a PDF form field as XML.

Return type

String

Description
An XML string of the PDF form fields.

Description

The XML string for your form field data.

Syntax

value = *object.***ExportFormAsXML** *TopElement, DefaultSeparator*

The ExportFormAsXML method has these required parts:

Part	Value Type	Description
TopElement	String	Determines how the data is bracketed.
DefaultSeparator	String	The separator to use.

FieldInfo

Description

Passes a fieldname and instance to the method and a PDFFieldInfo object is returned.

Return type

Object

Description
The field info object of the specified field.

Syntax

*value = object.***FieldInfo** *FieldName, Instance*

The FieldInfo method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
FieldName	String	The name of the field to retrieve.
Instance	Short	An instance of the field to retrieve. This is a number from 1 to the number of instances of the field in your input file. For example, if you had 3 copies of the field "FirstName" on your form, setting the "instance" value to 2 would retrieve the second copy.)

FindCertificate

Description

Finds a certificate in the Windows registry and prepares it to be used for signing. This method is a prerequisite for all calls to sign a document.

Return type

Long

Return Value	Description
>0	The certificate was found. The return value should be saved for use in signing.
-998	Product not registered/ Evaluation expired.
-997	Required product version not registered.
-1	The certificate was not found.
-2	Error creating the certificate object.

Syntax

*object.***FindCertificate** *CertificateCommonName, CertificateStore, UseLocalMachine*

The FindCertificate method has these required parts:

Part	Value Type	Description
CertificateCommonName	String	The common name of the certificate.
CertificateStore	String	The certificate store name (e.g. "My" or "SelfSignedCertificates").
UseLocalMachine	Long	If set to 1 then the certificate will be stored under HKEY_LOCAL_MACHINE. If set to 0 then the certificate will be stored under HKEY_CURRENT_USER. **NOTE:** For web applications, we recommend setting this variable to 1.

Example

```
Set tk = CreateObject("APToolkit.Object")
retCode = TK.FindCertificate("Joe Kant","My",1)
If (retCode < 0) Then
```

```
   retCode = TK.CreateCertificate("Joe Kant", "Management", "activePDF","Mission
Viejo", "CA", "US", "joe@activepdf.com", 1, "My", 365,0,"","")
   retCode = TK.FindCertificate("Joe Kant","My",1)
   If (u < 0) Then
      MsgBox("Can't find it!")
   End If
End If
r = tk.OpenOutputFile("output.pdf")
tk.SignOutputFile retCode, "activePDF Headquarters", "Our Document", "949-582-9002"
TK.SetFont "Helvetica",12
TK.PrintText 10,10,"This document should be signed."
Tk.CloseOutputFile
set TK = nothing
```

ForceColorReset

Description

Resets the color scheme the output PDF to black.

Return type

None

Syntax

*object.***ForceColorReset** *PageNr*

The ForceColorReset method has this part:

Part	Value Type	Description
Object		An expression of the Toolkit object.
PageNr	Long	Optional. 0 = The action will take place on the new or current open output page. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.

Remarks

Some PDF documents do not initialize their colors properly and assume that the default color black is available. If you set the text of the color to be printed using SetTextColor, the rest of the text on the following merged pages may be in that color. Call this method to reset the colors to black after the text item.

Example

```
'ForceColorReset Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("ForceColorReset.pdf")

      'Set the font and color for the text
      TK.SetFont "Helvetica", 20, 0
      TK.SetTextColor 50, 100, 255, 0, 0

      'Stamp text onto the page
      TK.PrintText 30, 740, "Hello World", 0

      'Reset the color for next PrintText
      TK.ForceColorReset 0

      'Stamp second text onto the page
      TK.PrintText 30, 700, "Hello World", 0
TK.CloseOutputFile

Set TK = Nothing
```

ForceHeaderColorReset

Description

Resets the color scheme of the PDF header to black.

Syntax

object.**ForceHeaderColorReset**

The ForceHeaderColorReset method has this part:

Part	Description
Object	An expression of the Toolkit object.

Remarks

Some PDF documents do not initialize their colors properly and assume that the default color black is available. If you set the text of the color to be printed using SetTextColor, the rest of the text on the following merged pages may be in that color. Call this method to reset the colors to black after the text item.

FromPDFDate

Description

Converts a PDF date to the internal variant date structure.

Return type

String

Description
The internal variant date.

Description

The converted date in variant date structure.

Syntax

*value = object.***FromPDFDate** *InDate*

The FromPDFDate method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
InDate	String	The date in PDF Date Format. For more information, refer to PDF Date Format.

Remarks

Refer to the ToPDFDate method and ModDate property.

GetBoundingBox

Description

Loads the bounding box information for a file and page number. The bounding box is the printable area of a PDF page.

Return type

Long

Return Value	Description
-1	Unable to open input file.
-2	The page number is invalid.
-3	Unable to locate page object.
0	Success.

Syntax

`object``.`**`GetBoundingBox`** *`FileName, PageNbr`*

The GetBoundingBox method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FileName	String	Filename to load. This does not become current input file. If a blank string ("") is passed, then the current input file from OpenInputFile is used.
PageNbr	Long	The page number to view.

GetCustomDocInfo (.NET only)

Description

GetCustomDocInfo enables you to retrieve the PDF custom document information fields when merging or copying a PDF document.

NOTE: This method is intended for use in a .NET environment. Refer to the CustomDocInfo property if you are implementing activePDF Toolkit an environment other than .NET.

Return type

String

Description
The data contained in the PDF custom document information field.

Syntax

*value = object.***GetCustomDocInfo** *ItemName*

The GetCustomDocInfo method has these parts:

Part	**Value Type**	**Description**
Object		An expression of the Toolkit object.
ItemName	String	The item name.

Remarks

Common fields used with the GetCustomDocInfo method are *DocVersion*, *URL*, *LogonID* and *Cookie Value*. If you want to access one of the standard fields, use the corresponding Toolkit property such as Author or Title. To retrieve data that is set with SetCustomDocInfo, you will need to call GetPDFInfo prior to calling GetCustomDocInfo.

Example C#

```
string myPath = System.Windows.Forms.Application.StartupPath;APToolkitNET.Toolkit TK
= new APToolkitNET.Toolkit();
// Open the output PDF
TK.OpenOutputFile(myPath + @"\output.pdf");
// Open the input file to get FieldInfo from
TK.OpenInputFile(myPath + @"\input.pdf");
// Set a CustomDocInfo value
TK.SetCustomDocInfo("This is my test field", "This is my test value");
// Copy the input to the output
TK.CopyForm(0, 0);
// Close the output file
TK.CloseOutputFile();
```

```
// Use GetPDFInfo to open the output.pdf as an input field
// and retrieve the standard and custom document info
TK.GetPDFInfo(myPath + @"\output.pdf");
// Pop up the contents of our new custom field
MessageBox.Show(TK.GetCustomDocInfo("This is my test field"));
// Close the input file
TK.CloseInputFile();
```

GetFormFieldData

Description

For a particular field number, returns the data stored in the currently open input file. Use this routine to extract data from a PDF containing form field and update the database.

Return type

String

Description
Returns data stored in the PDF file for that field name. (If no value is stored or the value specified for the FieldNum parameter is incorrect, an empty string is returned.)

Syntax

```
value = object.GetFormFieldData FieldNum
```

The GetFormFieldData method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FieldNum	Short	The specified field for data extraction.

GetFormFieldDataByName

Description

For a particular field name, returns the data stored in the currently open input file. Use this routine to extract data from a PDF form file and update the database.

Return type

String

Description
This data is stored in the PDF file for that field name. (If no value is stored or the value specified for the FieldNum parameter is incorrect, an empty string is returned.)

Syntax

*value = object.***GetFormFieldDataByName** *FieldName*

The GetFormFieldDataByName method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FieldName	String	The name of the field to extract the data.

GetFormFieldName

Description

Returns the field name stored in the currently open input PDF file for a particular field number.

Return type

String

Description
The field name stored in the currently open input file associated with a particular field number.

Syntax

*value = object.***GetFormFieldName** *FieldNum*

The GetFormFieldName method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FieldNum	Short	The field number associated with the form field.

GetHeaderTextWidth

Description

Retrieves text width for a string based on the current font information in the header.

Return type

Float

Description
The width of the text based on current font information. Specified inPDF Units.

Syntax

*value = object.***GetHeaderTextWidth** *TextString*

The GetHeaderTextWidth method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
TextString	String	The actual string to measure.

GetInputFields

Description

Returns a reference to a field instance collection containing fields with the same name in the PDF.

Return type

Collection

Description
The collection of field instances for fields with the same name, allowing for the application of settings, input, and retrieval.

Syntax

*value = object.***GetInputFields**

The GetInputPageField method has these required parts:

Part	Description
Object	An expression of the Toolkit object.

Remarks

All instances of the field in the current open input file will be returned. If a field is flattened, it may not be retrievable or changeable.

GetInputPageRotation

Description

The method returns the page rotation of a specified page in the currently open input file.

Return type

Short

Description
The counter-clockwise rotation of the page in degrees.

Syntax

*value = object.***GetInputPageRotation** *PageNumber*

The GetInputPageRotation method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
PageNumber	Long	The page number of which you want to know the rotation.

GetPDFInfo

Description

GetPDFInfo instructs Toolkit to open the specified PDF as the input file and loads the information from the Document Property fields for use the Author, Title, Subject, Keywords, Creator, Producer, CreateDate, ModDate and CustomDocInfo properties.

Return type

Long

Return Value	Description
-1	Unable to open input file.
-998	Product not registered/ Evaluation expired.
0	Success.

Syntax

object.**GetPDFInfo** *FileName*

The GetPDFInfo method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FileName	String	The PDF to retrieve the information from, which opens as the input file. If FileName is blank (""), the current input file will be used instead.

GetTextWidth

Description

Retrieves the text width of a string based on the font information specified with the SetFont method.

Return type

Float

Description
The text width of the string.

Syntax

*value = object.***GetTextWidth** *TextString, PageNr*

The GetTextWidth method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
TextString	String	The designated string.
PageNr	Long	Optional. 0 = The action will take place on the new or current open output page. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.

Example

```
'GetTextWidth Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("GetTextWidth.pdf")

      'Set the font for the printed text
      TK.SetFont "Helvetica", 20, 0

      'Get the width of the text so that it can be centered on the page
      strText = "Hello World"
      strTextWidth = TK.GetTextWidth(strText)

      'Print the text centered
      strPageWidth = 612 '8.5" (72 = 1")
      strLLX = (strPageWidth - strTextWidth) / 2
```

```
		strLLY = 740
		TK.PrintText strLLX, strLLY, strText, 0

	TK.CloseOutputFile

	Set TK = Nothing
```

GetUniqueFileName

Description

GetUniqueFileName instructs Toolkit to generate a unique file name appended with the .PDF suffix. The file name consists of the date and time for easy identification, as well as an incremented number to eliminate concurrency issues. You can use this unique file name with the OpenOutputFile method.

Return type

String

Description
The unique file name.

Syntax

*value = object.***GetUniqueFileName**

The GetUniqueFileName method has this part:

Part	Description
Object	An expression of the Toolkit object.

Example

```
'GetUniqueFileName Example
Set TK = CreateObject("APToolkit.Object")

'Get the unique file name
strUniqueID = TK.GetUniqueFileName

'Set the Output File to the unique ID
r = TK.OpenOutputFile(strUniqueID)

      'Set the font for the text
      TK.SetFont "Helvetica", 20, 0

      'Stamp Text onto the page
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile

Set TK = Nothing
```

GotoNextBookmarkLevel

Description

Indents another level on the bookmark tree. You can generate bookmarks after any pages are added and before you close the output document.

NOTE: Bookmarks default to the highest level of the tree. Subsequent uses of GotoNextBookmarkLevel and GotoPreviousBookmarkLevel will indent and outdent additional levels respectively.

Return type

None

Syntax

object.**GotoNextBookmarkLevel**

The GotoNextBookmarkLevel method has this part:

Part	**Description**
Object	An expression of the Toolkit object.

GotoPreviousBookmarkLevel

Description

Returns to the previous level on the bookmark tree.

NOTE: Bookmarks default to the highest level of the tree. Subsequent uses of GotoNextBookmarkLevel and GotoPreviousBookmarkLevel will indent and outdent additional levels respectively.

Return type

None

Syntax

object.**GotoPreviousBookmarkLevel**

The GotoPreviousBookmarkLevel method has this part:

Part	Description
Object	An expression of the Toolkit object.

GreyBar

Description

GreyBar instructs Toolkit to create a grey bar on new or existing page at the specified location.

Return type

None

Syntax

object.**GreyBar** *LLX, LLY, Width, Height, GreyLevel, PageNr*

The GreyBar method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
LLX	Float	The horizontal position for the lower-left corner of the bar. Uses the PDF Coordinate System.
LLY	Float	The vertical position for the lower-left corner of the bar. Uses the PDF Coordinate System.
Width	Float	The width of the bar. Specified in PDF Units.
Height	Float	The height of the bar. Specified in PDF Units.
GreyLevel	Float	The amount of grey in the bar, from 0.0 to 1.0 with 0.0 being black and 1.0 being white.
PageNr	Long	Optional. 0 = The action will take place on the new or current open output page. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.

Example

```
'GrayBar Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("GreyBar.pdf")

      'Set the font for the printed text
      TK.SetFont "Helvetica", 20, 0

      'Get the width of the text so that it can be centered on the page
```

```
        strText = "This is the Page Title"
        strTextWidth = TK.GetTextWidth(strText)

        'Print the text centered
        strPageWidth = 612 '8.5" (72 = 1")
        strLLX = (strPageWidth - strTextWidth) / 2
        strLLY = 740
        TK.PrintText strLLX, strLLY, strText, 0

        'Set a grey bar under the page title
        TK.GreyBar 30, 720, strPageWidth - 60, 3, 0.8, 0

    TK.CloseOutputFile

    Set TK = Nothing
```

ImageToPDF

Description

ImageToPDF converts one of the supported image types (see Appendix E: Supported Image Types) directly to PDF.

NOTE: If you require the resultant PDF to be encrypted, you will need to encrypt it after the PDF has been generated.

Return type

Long

Value	Description
-2999	Unable to call internal function/ invalid DLL specified.
-1999	Unable to load APTKIMGC.DLL.
-998	Product not registered/ Evaluation expired.
-997	Required product version not registered.
-3	Unable to open input file.
-1	Invalid image specified.
>0	Success.

Syntax

object.**ImageToPDF** *ImageFileName, PDFFileName*

The ImageToPDF method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
ImageFileName	String	The full path to the image. Setting ImageByteStream to a valid image allows you to set the ImageFileName parameter to "MEMORY".
PDFFileName	String	The full path to the resultant PDF.

InvisiblySignFile

Description

Instructs Toolkit to sign an existing PDF file invisibly.

NOTE: Toolkit appends the signature to the file and does not modify the contents.

Return type

Long

Return Value	Description
-998	Product not registered/ Evaluation expired.
-997	Required product version not registered.
-25	Invalid internal PDF structure.
-13	Unable to read forms structure.
-12	Invalid internal forms reference.
-11	Invalid internal forms reference.
-10	Invalid Internal page structure.
-9	Invalid signature.
-8	Invalid signature number.
-1	Unable to open input file.
0	Success.

Syntax

object.**InvisiblySignFile** *SigNumber, FileName, OutputFileName, Location, Reason, ContactInfo, SignatureType*

The InvisiblySignFile method has these required parts:

Part	Value Type	Description
SigNumber	Long	The value returned from FindCertificate.
FileName	String	The full path to the file to be signed. If set to MEMORY then

		InputByteStream must be called first.
OutputFileName	String	The full path to where you want the output file stored. If set to a blank string ("") the file specified with FileName is overwritten. If set to "MEMORY" or if FileName = "MEMORY" and this parameter is set = "", an output byte stream is generated.
Location	String	The location where the signature is applied. Typically, this is city and state or company location.
Reason	String	The reason for signing the document.
ContactInfo	String	Contact information of the signer.
SignatureType	Long	0 = PKCS#1 Acrobat 4+ signature (best backwards compatibility) 1 = PKCS#7 Acrobat 4+ signature. 2 = VeriSign Signature®. This requires the VeriSign plug-in. Certificate authority must be VeriSign. 3 = Microsoft Signature (Acrobat 6+).

Remarks

If the file is encrypted, you must call SetInputPasswords prior to calling InvisiblySignFile.

Example

```
Set TK = CreateObject("APToolkit.Object")
retCode = TK.FindCertificate("Joe Kant","My",1)
If (retCode < 0) Then
   retCode = TK.CreateCertificate("Joe Kant", "Management", "activePDF","Mission
Viejo", "CA", "US", "joe@activepdf.com", 1, "My", 365,0,"","")
   retCode = TK.FindCertificate("Joe Kant","My",1)
   If (u < 0) Then
      MsgBox("Can't find it!")
   End If
End If
r = tk.InvisiblySignFile(retCode, "test.pdf", "test-output.pdf", "activePDF
Headquarters", "Our Document", "949-582-9002",0)
set TK = nothing
```

IsFileLinearized

Description

Determines if a PDF is linearized (web optimized).

Return type

Long

Return Value	Description
1	The file is linearized.
0	The file is not linearized.

Syntax

*value = object.***IsFileLinearized** *FileName*

The IsFileLinearized method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FileName	String	The full path to the file to check. If FileName is blank ("") then the current input file, from OpenInputFile, is used.

IsFingerprintValid

Description

Detect whether or not the PDF contents have been altered by verifying the integrity of the fingerprint.

Return type

Long

Return Value	Description
1	The fingerprint is valid.
0	The fingerprint is not valid.

Syntax

*value = object.***IsFingerprintValid** *FileName*

The IsFingerprintValid method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FileName	String	The full path to the file to check.

Remarks

You can use the FingerprintOutputFile property apply a fingerprint to the output file. A fingerprint is a hash generated from the contents of the PDF, which is appended to the end of the output PDF, enabling you to verify the integrity of the file contents.

JPEGToPDF

Description

Converts a JPEG image to PDF.

NOTE: If you require the resultant PDF to be encrypted, you will need to encrypt it after the PDF has been generated.

Return type

Long

Return Value	Description
-2999	Unable to call internal function/invalid DLL specified.
-1999	Unable to load APTKIMGC.DLL.
-998	Product not registered/ Evaluation expired.
-997	Required product version not registered.
-3	Unable to open input file.
-1	Invalid JPEG specified.
>0	Success.

Syntax

object.**JPEGToPDF** *ImageFileName, PDFFileName*

The JPEGToPDF method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
ImageFileName	String	The full path to the JPEG. If you set the contents of ImageByteStream to a valid JPEG image, then you can set the ImageFileName parameter to "MEMORY".
PDFFileName	String	The full path to the resultant PDF document.

Example

```
'JPEGToPDF Example
```

```
Set TK = CreateObject("APToolkit.Object")

'Convert the JPEG to PDF
r = TK.JPEGToPDF("image.jpg", "JPEGToPDF.pdf")
If r < 0 Then
      MsgBox "JPEG failed to convert: " & r
End If

Set TK = Nothing
```

LinearizeFile

Description

Linearizes and prepares a PDF document for byte-serving.

NOTE: Files cannot be linearized in memory.

Return type

Long

Return Value	Description
-998	Product not registered/ Evaluation expired.
-997	Required product version not registered.
-1	Unable to open input file.
-2	Unable to generate output file.
>0	Success.

Syntax

object.**LinearizeFile** *InputFileName, OutputFileName, UserPassword*

The LinearizeFile method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
InputFileName	String	The input filename.
OutputFileName	String	The output filename.
UserPassword	String	If the input document is encrypted, the case-sensitive password required to view the document.

LineWidth

Description

Sets the width of the line drawn by the DrawTo method.

Return type

None

Syntax

object.**LineWidth** *Width, PageNr*

The LineWidth method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Width	Float	The width of the line. Specified in PDF Units.
PageNr	Long	Optional. 0 = The action will take place on the new or current open output page. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.

Example

```
'LineWidth Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("LineWidth.pdf")

      'Draw a border around the PDF
      strPageWidth = 612 '8.5" (72 = 1")
      strPageHeight = 792 '11" (72 = 1")
      strSpace = 20 'Space between edge of page and border

      'Set the width of the line from DrawTo
      TK.LineWidth 5, 0

      'Top of the page line
      TK.MoveTo strSpace, strPageHeight - strSpace, 0
      TK.DrawTo strPageWidth - strSpace, strPageHeight - strSpace, 0

      'Left of the page line
      TK.MoveTo strPageWidth - strSpace, strPageHeight - strSpace, 0
      TK.DrawTo strPageWidth - strSpace, strSpace, 0
      'Bottom of the page line
      TK.MoveTo strSpace, strSpace, 0
      TK.DrawTo strPageWidth - strSpace, strSpace, 0

      'Right of the page line
      TK.MoveTo strSpace, strSpace, 0
      TK.DrawTo strSpace, strPageHeight - strSpace, 0

TK.CloseOutputFile

Set TK = Nothing
```

LoadDBMapFile

Description

Loads a map file used to ensure the corresponding database fields and PDF form fields match correctly. A map file is useful when the database and PDF contain different field names for the same data.

Return type

Long

Return Value	Description
-1	Unable to open map file.
0	Success.

Syntax

*object.***LoadDBMapFile** *PathToMapFile*

The LoadDBMapFile method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
PathToMapFile	String	The full path to the map file. See the following section for information on the map file.

Map File

The map file contains a series of text lines, which correlates the desired data. The following is an example of a common text line:

```
PDFFieldName|DBFieldName|HowToMerge
```

The map file has these parts:

Parts	Description
PDFFieldName	The name of the field in your PDF document. In you are populating multiple PDF fields with the same name; do not append the field number to the end of the field name.
DBFieldName	The name of the field as it appears in the record set.
HowToMerge	The SetFormFieldData field attribute. For example, 0, 1, -995, and so on.

Remarks

To populate a field with an image, set the `DBFieldName` equal to `FILE:fieldname` (where `fieldname` is the path to a valid image).

MergeFile

Description

Concatenates a file to the end of the current output file.

Return type

Long

Return Value	Description
-998	Product not registered/ Evaluation expired.
-1	Unable to open input file.
0	No output file specified/Invalid output file specified.
>0	Success.

Syntax

object.**MergeFile** *FileName, StartPage, EndPage*

The MergeFile method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FileName	String	The name and path for the PDF file to use as the input for the merge. This file will become the current input file. If you set the contents of InputByteStream to a valid PDF, you can pass "MEMORY" as the file name.
StartPage	Long	The first page of the document to copy. (If set to 0, the page defaults to 1.)
EndPage	Long	The last page of the document to copy. (If set to 0, the page defaults to all pages.)

Remarks

This method is equivalent to calling:

```
x = APTOOLKIT.OpenInputFile(FileName)
x = APTOOLKIT.CopyForm(FirstPage,LastPage)
```

MoveTo

Description

By default, the starting point of the PDF Coordinate System is the lower-left corner of a PDF page or 0,0. MoveTo moves the starting point to the specified coordinates. This starting point corresponds to the endpoint specified with the DrawTo method, which also instructs Toolkit to draw a line from the starting point to the endpoint. The LineWidth method determines the width of the line.

Return type

None

Syntax

object.**MoveTo** *StartX, StartY, PageNr*

The MoveTo method has these required parts:

Part	**Value Type**	**Description**
Object		An expression of the Toolkit object.
StartX	Float	The horizontal position for the start point of the line. If you specify a line width, StartX corresponds to the horizontal position for the start point of the line that is equidistant from the upper and lower edges of the line. Uses the PDF Coordinate System.
StartY	Float	The vertical position for the start point of the line. If you specify a line width, StartY corresponds to the vertical position for the start point of the line that is equidistant from the upper and lower edges of the line. Uses the PDF Coordinate System.
PageNr	Long	Optional. 0 = The action will take place on the new or current open output page. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.

Example

```
'MoveTo Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("MoveTo.pdf")
```

```
        'Draw a border around the PDF
        strPageWidth = 612 '8.5" (72 = 1")
        strPageHeight = 792 '11" (72 = 1")
        strSpace = 20 'Space between edge of page and border

        'Top of the page line
        TK.MoveTo strSpace, strPageHeight - strSpace, 0
        TK.DrawTo strPageWidth - strSpace, strPageHeight - strSpace, 0

        'Left of the page line
        TK.MoveTo strPageWidth - strSpace, strPageHeight - strSpace, 0
        TK.DrawTo strPageWidth - strSpace, strSpace, 0

        'Bottom of the page line
        TK.MoveTo strSpace, strSpace, 0
        TK.DrawTo strPageWidth - strSpace, strSpace, 0

        'Right of the page line
        TK.MoveTo strSpace, strSpace, 0
        TK.DrawTo strSpace, strPageHeight - strSpace, 0

  TK.CloseOutputFile

  Set TK = Nothing
```

NewPage

Description

Specifies that the current page is finished and a new page is generated.

Return type

None

Syntax

*object.***NewPage**

The NewPage method has this part:

Part	Description
Object	An expression of the Toolkit object.

Example

```
'NewPage Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("NewPage.pdf")

      'Set the font for the text on page 1
      TK.SetFont "Helvetica", 20, 0

      'Stamp Text onto page 1
      TK.PrintText 30, 740, "Page 1", 0

      'Create the second page
      TK.NewPage

      'Set the font for the text on page 2
      TK.SetFont "Courier", 20, 0

      'Stamp Text onto page 2
      TK.PrintText 30, 740, "Page 2", 0

TK.CloseOutputFile

Set TK = Nothing
```

NumPages

Description

Returns the number of pages for the specified PDF file.

NOTE: NumPages closes any open input files and opens the specified file as the new input.

Return type

Long

Return Value	Description
#	The number of pages.
0	Unable to open input file.

Syntax

*value = object.***NumPages** *FileName*

The NumPages method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FileName	String	The name of the file from which Toolkit loads the PDF information. The file becomes the current input file. If the contents of InputByteStream are set to a valid PDF, then you can pass “MEMORY” for the file name. If you pass a blank string, the currently open input file is used.

OpenInputFile

Description

Opens the specified PDF as an input file, which is used as the source for many Toolkit functions.

Return type

Short

Return Value	Description
-1	Unable to open input file/ Invalid byte stream.
0	Success.

Syntax

object.**OpenInputFile** *InputFileName*

The OpenInputFile method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
InputFileName	String	Name of the file to open. If the contents of InputByteStream are set to a valid PDF, then you can pass “MEMORY” for the file name. If you pass a blank string, the currently open input file is used.

Remarks

OpenInputFile must be called if you want to retrieve field names and field data variables from a PDF. Only one input file can be active at a time. A subsequent call to OpenInputFile automatically closes the previous input file. Calls to any other function that sets the current input file, for example, GetPDFInfo, closes the current input file.

OpenOutputFile

Description

Generates a new PDF for subsequent output. You can set this value to memory and the file is generated in memory rather than disk.

Return type

Long

Return Value	Description
-998	Product not registered/ Evaluation expired.
-1	Unable to generate output file.
0	Success.

Syntax

object.**OpenOutputFile** *FileName*

The OpenOutputFile method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FileName	String	The name of the file to generate. (Passing "MEMORY" for the FileName will cause the output file to be generated in memory.)

Example

```
'OpenOutputFile Example
Set TK = CreateObject("APToolkit.Object")

'Set the output file to the new file name
r = TK.OpenOutputFile("OpenOutputFile.pdf")

      'Set the font for the text
      TK.SetFont "Helvetica", 20, 0

      'Stamp Text onto the page
      TK.PrintText 30, 740, "Hello World", 0

'Close Output File we are done creating the PDF
TK.CloseOutputFile

Set TK = Nothing
```

ParseDataStream

Description

Parses a delimited stream of string data containing valid field and value marks and populates the corresponding form fields in the PDF.

Syntax

*object.***ParseDataStream** *DataStream, ValueDelimiter, FieldDelimiter, LeaveReadOnlyFlag*

The ParseDataStream method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
DataStream	String	The delimited content to parse.
ValueDelimiter	String	The delimiter used to separate the value mark from the data.
FieldDelimiter	String	The delimiter used to separate the field name and data.
LeaveReadOnlyFlag	Short	Refer to Flags below.

Flags

Toolkit has these flags:

Toolkit Flag	Description
-4096	All bits will be cleared (set to 0). You can 'OR' 4096 with other bits to achieve the desired effect. (This affects the line on which it is called.)
-998	Flatten field and reset font, color and rotation information to field defaults. (You must use -998 on the line prior to the line you wish to reset.)
-997	Flatten field and do not reset font, color and rotation information.
-996	Flatten field using an image file as named in field data. The image type is auto-determined.
-995	Flatten field as a known JPEG using an image file as named in field data.
-994	Flatten field as a known TIFF using an image file as named in field data.
0	Read Only.

1	"As is". All attributes of the field remain unchanged.
2	Hidden.
4	Enable Printing.
8	Disable Zoom.
16	Disable Rotation.
32	The field will print, but cannot be viewed.
64	The field will be hidden and read only.

PrintImage

Description

Prints an image onto a new or existing PDF page. For a complete list of supported image types, refer to Appendix E: Supported Image Types.

NOTE: For JPEGs or TIFFs, use the PrintJPEG or PrintTIFF method.

Return type

Long

Return Value	Description
-998	Product not registered/ Evaluation expired.
-997	Required product version not registered.
99	Invalid path of filename specified.
97	Invalid image type specified.
1	Success.

Syntax

```
object.PrintImage ImageFileName, LLX, LLY, Width, Height, PersistRatio, PageNr
```

The PrintImage method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
ImageFileName	String	The full path to the image. If the ImageByteStream property is set to a valid image from a record set, or other source, you can set the FileName parameter to "MEMORY".
LLX	Float	The horizontal position for the lower-left corner of the image. Uses the PDF Coordinate System.
LLY	Float	The vertical position for the lower-left corner of the image. Uses the PDF Coordinate System.
Width	Float	The width of the image specified in PDF Units. 0 = Uses the current width of the image specified with the

		ImageFileName parameter. >0 = Overrides the current width of the image specified with the ImageFileName parameter.
Height	Float	The height of the image. Specified in PDF Units. 0 = Uses the current height of the image specified with the ImageFileName parameter. >0 = Overrides the current height of the image specified with the ImageFileName parameter.
PersistRatio	Long	1 = Height and width remain proportional if greater than 0. When printing a file type that contains no DPI information, you must pass a Height and Width and set PersistRatio to equal 1. 0 = Stretch to width and height.
PageNr	Long	Optional. 0 = The action will take place on the new or current open page of the input file or cover. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.

PrintJPEG

Description

Prints a JPEG onto a new or existing PDF page.

NOTE: For additional image types or TIFFs, use the PrintImage or PrintTIFF method.

Return type

Long

Return Value	Description
-998	Product not registered/ Evaluation expired.
-997	Required product version not registered.
-3	Unable to open input file.
-1	Invalid JPEG file or stream specified.
>0	Success.

Syntax

object.**PrintJPEG** *FileName, LLX, LLY, Width, Height, PersistRatio, PageNr*

The PrintJPEG method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FileName	String	The full path to the JPEG. If the ImageByteStream property is set to a valid JPEG from a record set, or other source, you can set the FileName parameter to "MEMORY".
LLX	Float	The horizontal position for the lower-left corner of the JPEG. Uses the PDF Coordinate System.
LLY	Float	The vertical position for the lower-left corner of the JPEG. Uses the PDF Coordinate System.
Width	Float	The width of the image. Specified in PDF Units. 0 = Uses the current width of the image specified with the FileName parameter.

		>0 = Overrides the current width for the image specified with the FileName parameter.
Height	Float	The height of the image. Specified in PDF Units. 0 = Uses the current height of the file specified with the FileName parameter. >0 = Overrides the current height for the file specified with the FileName parameter.
PersistRatio	Long	1 = Height and width remain proportional if greater than 0. 0 = Stretch to width and height.
PageNr	Long	Optional. 0 = The action will take place on the new or current open output page. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.

Example

```
'PrintJPEG Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("PrintJPEG.pdf")

      'Print the JPEG on the page
      TK.PrintJPEG "image.jpg", 0, 250, 0, 0, 1, 0

TK.CloseOutputFile

Set TK = Nothing
```

PrintLogo

Description

Prints a PDF Logo onto an existing PDF page. If no page is specified, a new page is generated.

NOTE: You must call AddLogo prior to calling PrintLogo. If AddLogo is called after PrintLogo, then the logo will only be placed on pages specified in a MergeFile or CopyForm function.

Return type

None

Syntax

object.**PrintLogo** *PageNr*

The PrintLogo method has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
PageNr	Long	Optional. 0 = The action will take place on the new or current open page of the input file or cover. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.

PrintMultilineText

Description

Prints a string of multi-line text onto the currently open output page. If no ouput page is currently open, a new page is generated.

Return type

None

Syntax

object.**PrintMultilineText** *FontName, FontSize, TextLLX, TextLLY, Width, Height, Text, Alignment, PageNr*

The PrintMultilineText method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
FontName	String	The font to use.
FontSize	Float	The size of the font. If you are using auto size text, a maximum font size is set by specifying a negative value.
TextLLX	Float	The horizontal position for the lower-left corner of the first font glyph in the first line of printed text. Uses the PDF Coordinate System.
TextLLY	Float	The vertical position for the lower-left corner of the first font glyph in the first line of printed text. Uses the PDF Coordinate System.
Width	Float	The width of the space to insert the text. Specified in PDF Units.
Height	Float	The height of the space to insert the text. Specified in PDF Units.
Text	String	The actual text to print.
Alignment	Short	The alignment for the text. 0 = Left justified (Default) 1 = Center 2 = Right justified

		3 = Full justified
PageNr	Long	Optional. 0 = The action will take place on the new or current open output page. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.

Remarks

If justified is selected, the last line of the text in the defined area will be left justified. To justify the last line, pass a carriage return after the last character.

Example

```
'PrintMultilineText Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("PrintMultilineText.pdf")

      'Stamp multiline text onto the page
      strText = "This is multiline text that is printed on the page"
      TK.PrintMultilineText "Helvetica", 16, 241, 700, 130, 100, strText, 1, 0

TK.CloseOutputFile

Set TK = Nothing
```

PrintText

Description

Prints a string of text onto the currently open output page. If no ouput page is currently open, a new page is generated.

NOTE: Must be called after the SetFont method.

Return type

None

Syntax

object.**PrintText** *LLX, LLY, Text, PageNr*

The PrintText method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
LLX	Float	The horizontal position for the lower-left corner of space defined by the string of text. Uses the PDF Coordinate System.
LLY	Float	The vertical position for the lower-left corner of the space defined by the string of text. Uses the PDF Coordinate System.
Text	String	The string of text to print.
PageNr	Long	Optional. 0 = The action will take place on the new or current open output page. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.

Example

```
'PrintText Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("PrintText.pdf")

      'Set the font for the text
      TK.SetFont "Helvetica", 20, 0

      'Stamp Text onto the page
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile
```

```
Set TK = Nothing
```

PrintTIFF

Description

Prints a TIFF onto the currently open output page. If no ouput page is currently open, a new page is generated.

NOTE: For additional image types or JPEGs, use the PrintImage or PrintJPEG method.

Return type

Long

Return Value	Description
-998	Product not registered/ Evaluation expired.
-997	Required product version not registered.
99	Unable to open input file.
97	Invalid TIFF file specified.
1	Success.

Syntax

object.**PrintTIFF** *FileName, LLX, LLY, Width, Height, PersistRatio, PageNr*

The PrintTIFF method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FileName	String	The full path to the TIFF. If the ImageByteStream property is set to a valid TIFF from a record set, or other source, you can set the FileName parameter to "MEMORY".
LLX	Float	The horizontal position for the lower-left corner of the TIFF. Uses the PDF Coordinate System.
LLY	Float	The vertical position for the lower-left corner of the TIFF. Uses the PDF Coordinate System.
Width	Float	The width of the TIFF specified in PDF Units. 0 = Uses the current width of the file specified with the

		FileName parameter. >0 = Overrides the current width for the file specified with the FileName parameter.
Height	Float	The height of the TIFF. Specified in PDF Units. 0 = Uses the current height of the file specified with the FileName parameter. >0 = Overrides the current height for the file specified with the FileName parameter.
PersistRatio	Long	1 = Persist height and width proportionately if greater than 0. 0 = Stretch to width and height.
PageNr	Long	Optional. 0 = The action will take place on the new or current open output page. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.

Example

```
'PrintTIFF Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("PrintTIFF.pdf")

      'Print the TIFF on the page
      TK.PrintTIFF "image.tif", 0, 250, 0, 0, 1, 0

TK.CloseOutputFile

Set TK = Nothing
```

ResetFormFields

Description

Resets the all form fields to the default values. You can call ResetFormFields after the CopyForm method to prepare the form fields for additional input using the SetFormFieldData method.

Return type

None

Syntax

`object`.**`ResetFormFields`**

The ResetFormFields method has this part:

Part	Description
Object	An expression of the Toolkit object.

ResetHeaderTextColor

Description

Resets the header text color scheme to black.

Return type

None

Syntax

`object`.**`ResetHeaderTextColor`**

The ResetHeaderTextColor method has this part:

Part	Description
Object	An expression of the Toolkit object.

ResetTextColor

Description

Resets the text color scheme to black.

Return type

None

Syntax

*object.***ResetTextColor** *PageNr*

The ResetTextColor method has this part:

Part	Value Type	Description
Object		An expression of the Toolkit object.
PageNr	Long	Optional. 0 = The action will take place on the new or current open output page. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.

Example

```
'ResetTextColor Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("ResetTextColor.pdf")

      'Set the font and color for the text
      TK.SetFont "Helvetica", 20, 0
      TK.SetTextColor 50, 100, 255, 0, 0

      'Stamp text onto the page
      TK.PrintText 30, 740, "Hello World", 0

      'Reset the text color for next PrintText
      TK.ResetTextColor 0

      'Stamp second text onto the page
      TK.PrintText 30, 700, "Hello World", 0

TK.CloseOutputFile

Set TK = Nothing
```

SaveMemoryToDisk

Description

After creating a PDF in memory, this method saves the resultant PDF to disk.

NOTE: You must call CloseOutputFile first.

Return type

Long

Return Value	Description
0	Success.
not = 0	A number relating to a specific Win32 error.

Syntax

object.**SaveMemoryToDisk** *FileName*

The SaveMemoryToDisk method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FileName	String	The full path to the output file destination.

Example

```
'SaveMemoryToDisk Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("MEMORY")

      TK.SetFont "Helvetica", 20, 0
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile

'Save the created PDF in memory to the hard disk
r = TK.SaveMemoryToDisk("SaveMemoryToDisk.pdf")
If r <> 0 Then
      msgbox "Failed to save memory to disk:  " & r
End If

Set TK = Nothing
```

SetCustomDocInfo (.NET only)

Description

SetCustomDocInfo enables you to set the PDF custom document information fields when merging or copying a PDF document. Common fields used with the SetCustomDocInfo method are *DocVersion*, *URL*, *LogonID* and *Cookie Value*.

NOTE: This method is intended for use in a .NET environment. Refer to the CustomDocInfo property if you are implementing activePDF Toolkit an environment other than .NET.

Return type

None

Syntax

object.**SetCustomDocInfo***(ItemName) = value*

The SetCustomDocInfo method has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	String	The data to populate the PDF custom document information fields.

Remarks

If you want to access one of the standard fields, use the corresponding Toolkit property such as Author or Title.

Example C#

```
string myPath = System.Windows.Forms.Application.StartupPath; APToolkitNET.Toolkit
TK = new APToolkitNET.Toolkit();
// Open the output PDF
TK.OpenOutputFile(myPath + @"\output.pdf");
// Open the input file to get FieldInfo from
TK.OpenInputFile(myPath + @"\input.pdf");
// Set a CustomDocInfo value
TK.SetCustomDocInfo("This is my test field", "This is my test value");
// Copy the input to the output
TK.CopyForm(0, 0);
// Close the output file
TK.CloseOutputFile();
// Use GetPDFInfo to open the output.pdf as an input field
// and retrieve the standard and custom document info
TK.GetPDFInfo(myPath + @"\output.pdf");
// Pop up the contents of our new custom field
MessageBox.Show(TK.GetCustomDocInfo("This is my test field"));
// Close the input file
TK.CloseInputFile();
```

SetDBInputTemplate

Description

Sets the name of the input file for form field population from a database. This is useful in a multi-page operation, as the input file does not have to be reopened for each page.

Return type

None

Syntax

object.**SetDBInputTemplate** *InputPDFPath*

The SetDBInputTemplate method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
InputPDFPath	String	Full path to the template file.

SetDBMultiRowSeparator

Description

Specifies the delimiter to use in delimiting the value between form field name and the row number set in the PDF template.

Return type

None

Syntax

`object.``**SetDBMultiRowSeparator**` *`MultiRowSeparator`*

The SetDBMultiRowSeparator method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
MultiRowSeparator	String	The value that separates the field name from the number of the row in the PDF file.

Remarks

When executing a one-to-many query, Toolkit can fill in many rows within the PDF template. To specify rows within the template, you can use PDF field names like *Fieldname.1*, *Fieldname-1,* or *Fieldname1*.

For example, you can call APTOOLKIT.SetDBMultiRowSeparator "" to make your field names appear like this:

```
Qty1 Item1 Description1 Price1
Qty2 Item2 Description2 Price2
Qty3 Item3 Description3 Price3
```

Alternatively, you can call APTOOLKIT.SetDBMultiRowSeparator "-" to make your field names appear like this:

```
Qty-1 Item-1 Description-1 Price-1
Qty-2 Item-2 Description-2 Price-2
Qty-3 Item-3 Description-3 Price-3
```

SetDefaultDBMergeFlag

Description

Sets a flag specifying the default form field state to be applied to all form fields after a DBToForm operation.

Return type

None

Syntax

object.**SetDefaultDBMergeFlag** *DefaultMergeFlag*

The SetDefaultDBMergeFlag method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
DefaultMergeFlag	Short	The flag to use. (Refer to the following section for a complete list of flags.)

Flags

Toolkit has these flags:

Toolkit Flag	Description
-4096	All bits will be cleared (set to 0). You can 'OR' 4096 with other bits to achieve the desired effect. (This affects the line on which it is called.)
-998	Flatten field and reset font, color and rotation information to field defaults. (You must use -998 on the line prior to the line you wish to reset.)
-997	Flatten field and do not reset font, color and rotation information.
-996	Flatten field using an image file as named in field data. The image type is auto-determined.
-995	Flatten field as a known JPEG using an image file as named in field data.
-994	Flatten field as a known TIFF using an image file as named in field data.
0	Read Only.
1	"As is". All attributes of the field remain unchanged.

2	Hidden.
4	Enable Printing.
8	Disable Zoom.
16	Disable Rotation.
32	The field will print, but cannot be viewed.
64	The field will be hidden and read only.

Remarks

To switch the flag, use ‘OR’, fldFlags = 64 or 2.

SetEPMParams

Description

Sets the parameters to be used in creating an electronic postmark.

NOTE: For valid electronic postmarks, you must have access to the USPS EPM Server. For additional information, visit www.activepdf.com or contact the USPS.

Syntax

object.**SetEPMParams** *SigNumber, UserID, Password*

The SetEPMParams method has these required parts:

Part	**Value Type**	**Description**
SigNumber	Long	The value returned from FindCertificate.
UserID	String	The UserID required to alter the signature parameters.
Password	String	The password required to alter the signature parameters.

Remarks

Use FindCertificate to pass the user ID and password to return the certificate number.

SetFlattenedColorInfo

Description

Sets the color of the text in form fields that are flagged for flattening using SetFormFieldData or SetDefaultDBMergeFlag in the RGB color-space.

NOTE: If you are flattening the form field and want to change the color, SetFlattenedColorInfo must be called before SetFormFieldData.

Return type

None

Syntax

object.**SetFlattenedColorInfo** *AmountRed, AmountGreen, AmountBlue, AmountGreyscale, FillMode, AmountStrokeRed,AmountStrokeGreen, AmountStrokeBlue, AmountStrokeGreyscale*

The SetFlattenedColorInfo method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
AmountRed	Short	The amount of red being applied to the text color. The value ranges from 0 to 255 with 255 being true red.
AmountGreen	Short	The amount of green being applied to the text color. The value ranges from 0 to 255 with 255 being true green.
AmountBlue	Short	The amount of blue being applied to the text color. The value ranges from 0 to 255 with 255 being true blue.
AmountGreyscale	Short	The amount of white being applied to the text color. The value ranges from 0 to 255 with 255 being true white. (Setting the color to greyscale changes the internal color space to greyscale. Set the value to 0 for true black.)
FillMode	Short	The type of fill to apply. 0 = Fill only (Default). 1 = Stroke only. 2 = Fill then stroke. 3 = No fill or stroke.
AmountStrokeRed	Short	The amount of red being applied to the text stroke color. The value ranges from 0 to 255 with 255 being

		true red.
AmountStrokeGreen	Short	The amount of green being applied to the text stroke color. The value ranges from 0 to 255 with 255 being true green.
AmountStrokeBlue	Short	The amount of blue being applied to the text stroke color. The value ranges from 0 to 255 with 255 being true blue.
AmountStrokeGreyscale	Short	The amount of white being applied to the text stroke color. The value ranges from 0 to 255 with 255 being true white. (Setting the color to greyscale changes the internal color space to greyscale. Set the value to 0 for true black.)

SetFlattenedColorInfoCMYK

Description

Sets the color of the text in form fields that are flagged for flattening using SetFormFieldData or SetDefaultDBMergeFlag, in the CMYK color-space.

NOTE: If you are flattening the form field and want to change the color, SetFlattenedColorInfoCMYK must be called before SetFormFieldData.

Syntax

object.**SetFlattenedColorInfoCMYK** *AmountCyan, AmountMagenta, AmountYellow, AmountBlack, FillMode,AmountStrokeCyan, AmountStrokeMagenta, AmountStrokeYellow, AmountStrokeBlack*

The SetFlattenedColorInfoCMYK method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
AmountCyan	Short	The amount of cyan to be applied to the text color. The value ranges from 0 to 100 with 100 being true cyan.
AmountMagenta	Short	The amount of magenta to be applied to the text color. The value ranges from 0 to 100 with 100 being true magenta
AmountYellow	Short	The amount of yellow to be applied to the text color. The value ranges from 0 to 100 with 100 being true yellow.
AmountBlack	Short	The amount of black to be applied to the text color. The value ranges from 0 to 100 with 100 being true black. (To reset color set all other colors to 0 and AmountBlack to 100).
FillMode	Short	The type of fill to apply. 0 = Fill only (Default). 1 = Stroke only. 2 = Fill then stroke. 3 = No fill or stroke.
AmountStrokeCyan	Short	The amount of cyan to be applied to the text stroke color. The value ranges from 0 to 100 with 100 being true cyan.

AmountStrokeMagenta	Short	The amount of magenta to be applied to the text stroke color. The value ranges from 0 to 100 with 100 being true magenta
AmountStrokeYellow	Short	The amount of yellow to be applied to the text stroke color. The value ranges from 0 to 100 with 100 being true yellow.
AmountStrokeBlack	Short	The amount of black to be applied to the text stroke color. The value ranges from 0 to 100 with 100 being true black. (To reset color set all other colors to 0 and AmountBlack to 100).

SetFlattenedFont

Description

Sets the font to use for the resultant text when flattening a form field. If no font is specified, the text will retain its current font settings.

NOTE: SetFlattenedFont must be prior to SetFormFieldData when flattening fields.

Return type

None

Syntax

object.**SetFlattenedFont** *FontName, FontSize*

The SetFlattenedFont method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FontName	String	The case-sensitive name of the font to use.
FontSize	Float	The size of the font.

Remarks

For information on how Toolkit uses and locates fonts, refer to Toolkit Font Usage.

SetFlattenedRotation

Description

The amount of rotation to use on form fields that are flagged for flattening.

NOTE: If you want to flatten the form field and rotate it, you must call SetFlattenedRotation prior to calling SetFormFieldData.

Return type

None

Syntax

object.**SetFlattenedRotation** *Degrees*

The SetFlattenedRotation method has these required parts:

Part	**Value Type**	**Description**
Object		An expression of the Toolkit object
Degrees	Short	The amount of counterclockwise rotation in degrees. (Set less than 0 for clockwise rotation.)

SetFont

Description

Specifies the font to be used for Toolkit font operations. If there is no currently open output page, a new page is generated.

Return type

None

Syntax

object.**SetFont** *FontName, FontSize, PageNr*

The SetFont method has these required parts:

Part	**Value Type**	**Description**
Object		An expression of the Toolkit object.
FontName	String	The case-sensitive name of the font. Refer to the sections below for additional details.
FontSize	Float	The size of the font to use. Specified in PDF Units.
PageNr	Long	Optional. 0 = The action will take place on the new or current open output page. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.

Parameters

SetFont contains additional sub-parameters for the `FontName` parameter, which follow the font name and are separated by a bar character "|". `FontName` has these additional parameters:

Parameter	**Description**
encoding	Specifies the type of encoding to use. Refer to the Encoding section below.
bold	1 = Force font to be bold. 0 = As is (Default).
italic	1 = Force font to italic. 0 = As is. (Default)

Index	1 to n, where n is undefined. If the font is a TrueType Collection, this defines the index to use. (Defaults to 1)
defaultwidth	Overrides the default width. (Specified in font units.) **NOTE:** Many fonts, including the default double-byte fonts contained in Toolkit are proportional width fonts.

You can pass multiple sub-parameters after `FontName` by separating the parameters with a comma. For example, if the FontName is Arial and you wanted it to be bold, you would pass “`Arial|encoding, bold =1`”.

The following are additional examples of the Encoding Parameter:

- `"Helvetica|encoding=WinAnsiEncoding"`
- `"mscomic.ttf|encoding=WinAnsiEncoding,bold=1"`
- `"msmincho.ttf|encoding=UniJIS-UCS2-H"`

You can also specify the font to force bold or italic in the name by specifying *Bold*, *Italic* or *BoldItalic* after the name, separated by a comma:

- `Helvetica,Bold`.
- `Helvetica,Italic`.
- `Helvetica,BoldItalic`.

NOTE: If a bold or italic (or bold italic) font does not exist, the normal font will be loaded and the PDF viewer may synthesize the attributes. This applies to the bold and index parameters as well.

Encoding

The encoding parameter supports the following standard, Chinese, Japanese and Korean encodings:

Encoding Type	Encoding
Standard	WinAnsiEncoding. MacRomanEncoding. MacExpertEncoding. PDFDocEncoding. Identity-H. Identity-V.
Chinese (Simplified)	Adobe-GB1-UCS2. UniGB-UCS2-H. UniGB-UCS2-V.
Chinese (Traditional)	Adobe-CNS1-UCS2. UniCNS-UCS2-H.

	UniCNS-UCS2-V.
Japanese	Adobe-Japan1-UCS2. UniJIS-UCS2-H. UniJIS-UCS2-V. UniJIS-UCS2-HW-H. UniJIS-UCS2-HW-V.
Korean	Adobe-Korea1-UCS2. UniKS-UCS2-H. UniKS-UCS2-V.

Remarks

For information on how Toolkit uses and locates fonts, refer to Toolkit Font Usage.

Example

```
'SetFont Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("SetFont.pdf")

      'Set the font for the text to be printed
      TK.SetFont "Helvetica", 20, 0

      'Stamp Text onto the page
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile

Set TK = Nothing
```

SetFormFieldData

Description

SetFormFieldData instructs Toolkit to populate the form fields of the currently open input file with data while writing the fields to the output file during the next call to CopyForm.

Return type

None

Syntax

object.**SetFormFieldData** *FieldName, FieldData, LeaveReadOnlyFlag*

The SetFormFieldData method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
FieldName	String	The field name to set the data.
FieldData	String	The data to set.
LeaveReadOnlyFlag	Short	The flag to use. (Refer to the following section for a complete list of flags.)

Flags

Toolkit has these flags:

Toolkit Flag	Description
-4096	All bits will be cleared (set to 0). You can 'OR' 4096 with other bits to achieve the desired effect. (This affects the line on which it is called.)
-998	Flatten field and reset font, color and rotation information to field defaults. (You must use -998 on the line prior to the line you wish to reset.)
-997	Flatten field and do not reset font, color and rotation information.
-996	Flatten field using an image file as named in field data. The image type is auto-determined.
-995	Flatten field as a known JPEG using an image file as named in field data.
-994	Flatten field as a known TIFF using an image file as named in field data.

0	Read Only.
1	"As is". All attributes of the field remain unchanged.
2	Hidden.
4	Enable Printing.
8	Disable Zoom.
16	Disable Rotation.
32	The field will print, but cannot be viewed.
64	The field will be hidden and read only.

Remarks

To switch the flag, use ‘OR’, fldFlags = 64 or 2.

When defining multiple fields with the same name, the fields will have data in common but may differ in appearance (For example, placement, font, and alignment may be different). SetFormFieldData sets the data in all instances while respecting their individual appearance settings. You will need to ensure these are set in the inherent field. If you are using printable characters, you can use the Chr$(13) exception to mark a new line in text.

If you want to set a *checkbox* or *radio* button, you will need to pass the value of the “`Exported Value`” When setting a radio button, you will need to ensure all buttons have the same name in the group.

SetHeaderFont

Description

Set the font to use for headers.

Return type

None

Syntax

object.**SetHeaderFont** *FontName, FontSize*

The SetHeaderFont method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FontName	String	The font name.
FontSize	Float	The size of the font.

Remarks

For additional information on how Toolkit uses fonts, refer to Toolkit Font Usage.

SetHeaderGreyBar

Description

Places a gray rectangle or bar starting at the specified coordinates on all subsequent calls to MergeFile and CopyForm.

Return type

None

Syntax

object.**SetHeaderGreyBar** *ULX, ULY, Width, Height, GreyLevel*

The SetHeaderGreyBar method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
ULX	Float	The horizontal position for the upper-left corner of the bar. Uses the PDF Coordinate System.
ULY	Float	The vertical position for the upper-left corner of the bar. Uses the PDF Coordinate System.
Width	Float	The width of the bar specified in PDF Units.
Height	Float	The height of the bar. Specified in PDF Units.
GreyLevel	Float	The amount of grey in the bar, from 0.0 to 1.0 with 0.0 being black and 1.0 being white.

SetHeaderHLine

Description

SetHeaderHLine instructs Toolkit to draw a line from a specified start point to an endpoint along a horizontal axis. Toolkit draws the line on all pages affected by subsequent calls to MergeFile and CopyForm.

Return type

None

Syntax

object.**SetHeaderHLine** *StartX, EndX, Y, Width*

The SetHeaderHLine method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
StartX	Float	The horizontal position for the start point of the line. Uses the PDF Coordinate System.
EndX	Float	The horizontal position for the endpoint of the line. Uses the PDF Coordinate System.
Y	Float	The vertical position of the line. Uses the PDF Coordinate System.
Width	Float	The width of the line. Specified in PDF Units.

SetHeaderImage

Description

Specifies an image to be printed on all pages affected by subsequent calls to MergeFile and CopyForm.

NOTE: For a list of supported image types for the SetHeaderImage method, see Appendix E: Supported Image Types.

Return type

Long

Return Value	Description
-998	Product not registered/ Evaluation expired.
-997	Required product version not registered.
-1999	Unable to load APTKIMGC.DLL.
-2999	Unable to call internal function/ invalid DLL specified.
-3	Unable to open input file.
-1	Invalid image type specified.
>0	Success.

Syntax

object.**SetHeaderImage** *ImageFileName, LLX, LLY, Width, Height, PersistRatio*

The SetHeaderImage method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
FileName	String	The full path to the image.
LLX	Float	The horizontal position of the lower-left corner of the image. Uses the PDF Coordinate System.
LLY	Float	The vertical position of the lower-left corner of the image. Uses the PDF Coordinate System.
Width	Float	The width of the image. Specified in PDF Units.

		0 = Uses the current width of the file specified with the FileName parameter. >0 = Overrides the current width for the file specified with the FileName parameter.
Height	Float	The height of the image. Specified in PDF Units. 0 = Uses the current height of the file specified with the FileName parameter. >0 = Overrides the current height for the file specified with the FileName parameter.
PersistRatio	Long	1 = Persist height and width proportionately if greater than 0. 0 = Stretch to width and height.

SetHeaderJPEG

Description

Specifies a JPEG image to be printed on all pages affected by subsequent calls to MergeFile and CopyForm.

Return type

Long

Return Value	Description
-998	Product not registered/ Evaluation expired.
-997	Required product version not registered.
-3	Unable to open input file.
-1	Invalid JPEG file or stream specified.
>0	Success.

Syntax

object.**SetHeaderJPEG** *FileName, LLX, LLY, Width, Height, PersistRatio*

The SetHeaderJPEG method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
FileName	String	The full path to the JPEG.
LLX	Float	The horizontal position of the lower-left corner of the JPEG. Uses the PDF Coordinate System.
LLY	Float	The vertical position of the lower-left corner of the JPEG. Uses the PDF Coordinate System.
Width	Float	The width of the JPEG. Specified in PDF Units. 0 = Uses the current width of the image specified with the FileName parameter. >0 = Overrides the current width for the image specified with the FileName parameter.

Height	Float	The height of the JPEG specified in PDF Units. 0 = Uses the current height of the image specified with the FileName parameter. >0 = Overrides the current height for the image specified with the FileName parameter.
PersistRatio	Long	1 = Persist height and width proportionately if greater than 0. 0 = Stretch to width and height.

SetHeaderMultilineText

Description

Specifies a multi-line text string to be printed on all pages affected by subsequent calls to MergeFile and CopyForm.

Return type

None

Syntax

object.**SetHeaderMultilineText** *FontName, FontSize, ULX, ULY, Width, Height, Text, Alignment*

The SetHeaderMultilineText method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
FontName	String	The font to use.
FontSize	Float	The size of the font. You can specify auto-text size by selecting a negative font size. The value specified as negative will be set as the maximum font size and auto-sized down as needed.
ULX	Float	The horizontal position for the upper-left corner of the defined space to print the header text. Uses the PDF Coordinate System.
ULY	Float	The vertical position for the upper-left corner of the defined space to print the header text. Uses the PDF Coordinate System.
Width	Float	The width of the defined space. Specified in PDF Units.
Height	Float	The height of the defined space. Specified in PDF Units.
Text	String	The string of text to print in the header.
Alignment	Short	The alignment for the text. 0 = Left justified (Default) 1 = Center 2 = Right justified

SetHeaderRotation

Description

Sets rotation for text applied in subsequent calls to SetHeaderText.

Return type

None

Syntax

object.**SetHeaderRotation** *RotationAngle*

The SetHeaderRotation method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
RotationAngle	Short	The amount of counterclockwise rotation in degrees. (Set less than 0 for clockwise rotation.)

SetHeaderText

Description

Specifies a string of text to be printed on all pages affected by subsequent calls to MergeFile and CopyForm.

NOTE: SetHeaderFont must be called prior to SetHeaderText.

Return type

None

Syntax

object.**SetHeaderText** *LLX, LLY, Text*

The SetHeaderText method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
LLX	Float	The horizontal position for the lower-left corner of the space defined by the string of text. Uses the PDF Coordinate System.
LLY	Float	The vertical position for the lower-left corner of the space defined by the string of text. Uses the PDF Coordinate System.
Text	String	The text to print.

Remarks

Any art, crop or trim boxes should be taken into consideration as they will affect the placement of your header text.

SetHeaderTextBackground

Description

Specifies whether header text is rendered in the foreground or the background.

Return type

None

Syntax

object.**SetHeaderTextBackground** *UseBackground*

The SetHeaderTextBackground method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
UseBackground	Short	1 = The header text will be in the background. 0 = The header text will be in the foreground. (Default)

Remarks

If SetHeaderTextBackground is called before SetHeaderImage, SetHeaderJPEG, or SetHeaderTIFF the specified image will also appear in the background.

SetHeaderTextColor

Description

Sets the text color for subsequent calls to SetHeaderText.

Return type

None

Syntax

object.**SetHeaderTextColor** *AmountRed, AmountGreen, AmountBlue, Greyscale*

The SetHeaderTextColor method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
AmountRed	Short	The amount of red being applied to the text stroke color. The value ranges from 0 to 255 with 255 being true red.
AmountGreen	Short	The amount of green being applied to the text stroke color. The value ranges from 0 to 255 with 255 being true green.
AmountBlue	Short	The amount of blue being applied to the text stroke color. The value ranges from 0 to 255 with 255 being true blue.
Greyscale	Short	The amount of white being applied to the text stroke color. The value ranges from 0 to 255 with 255 being true white. (Setting the color to greyscale changes the internal color space to greyscale. Set the value to 0 for true black.)

SetHeaderTextColorCMYK

Description

Sets the color of header text to be printed in CMYK format.

Return type

None

Syntax

object.**SetHeaderTextColorCMYK** *AmountCyan, AmountMagenta, AmountYellow, AmountBlack*

The SetHeaderTextColorCMYK method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
AmountCyan	Short	The amount of cyan being applied to the text stroke color. The value ranges from 0 to 100 with 100 being true cyan.
AmountMagenta	Short	The amount of magenta to be applied to the text color. The value ranges from 0 to 100 with 100 being true magenta.
AmountYellow	Short	The amount of yellow to be applied to the text color. The value ranges from 0 to 100 with 100 being true yellow.
AmountBlack	Short	The amount of black to be applied to the text color. The value ranges from 0 to 100 with 100 being true black. (To reset color set all other colors to 0 and AmountBlack to 100).

SetHeaderTextFillMode

Description

Defines how text is filled during subsequent calls to SetHeaderText.

Return type

None

Syntax

object.**SetHeaderTextFillMode** *FillMode*

The SetHeaderTextFillMode method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
FillMode	Short	The type of fill to apply. 0 = Fill only (Default). 1 = Stroke only. 2 = Fill then stroke. 3 = No fill or stroke.

SetHeaderTextStrokeColor

Description

Defines the color of stroke, versus fill, during subsequent calls to SetHeaderText in RGB color mode.

NOTE: To use SetHeaderTextStrokeColor, SetHeaderTextFillMode must be set equal to 1 or 2.

Return type

None

Syntax

object.**SetHeaderTextStrokeColor** *AmountRed, AmountGreen, AmountBlue, Greyscale*

The SetHeaderTextStrokeColor method has these required parts:

Part	**Value Type**	**Description**
Object		An expression of the Toolkit object
AmountRed	Short	The amount of red being applied to the text stroke color. The value ranges from 0 to 255 with 255 being true red.
AmountGreen	Short	The amount of green being applied to the text stroke color. The value ranges from 0 to 255 with 255 being true green.
AmountBlue	Short	The amount of blue being applied to the text stroke color. The value ranges from 0 to 255 with 255 being true blue.
Greyscale	Short	The amount of white being applied to the text stroke color. The value ranges from 0 to 255 with 255 being true white. (Setting the color to greyscale changes the internal color space to greyscale. Set the value to 0 for true black.)

SetHeaderTextStrokeColorCMYK

Description

Defines the color of stroke, versus fill, during subsequent calls to SetHeaderText in CMYK color mode.

Return type

None

Syntax

object.**SetHeaderTextStrokeColorCMYK** *AmountCyan, AmountMagenta, AmountYellow, AmountBlack*

The SetHeaderTextStrokeColorCMYK method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
AmountCyan	Short	The amount of cyan being applied to the text stroke color. The value ranges from 0 to 100 with 100 being true cyan.
AmountMagenta	Short	The amount of magenta to be applied to the text color. The value ranges from 0 to 100 with 100 being true magenta.
AmountYellow	Short	The amount of yellow to be applied to the text color. The value ranges from 0 to 100 with 100 being true yellow.
AmountBlack	Short	The amount of black to be applied to the text color. The value ranges from 0 to 100 with 100 being true black. (To reset color set all other colors to 0 and AmountBlack to 100).

SetHeaderTIFF

Description

Specifies a TIFF image to be printed on all pages affected by subsequent calls to MergeFile and CopyForm.

Return type

Long

Return Value	Description
-998	Product not registered/ Evaluation expired.
-997	Required product version not registered.
99	Unable to open input file.
97	Invalid TIFF specified.
1	Success.

Syntax

object.**SetHeaderTIFF** *FileName, LLX, LLY, Width, Height, PersistRatio*

The SetHeaderTIFF method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FileName	String	The full path to the TIFF.
LLX	Float	The horizontal position for the lower-left corner of the TIFF. Uses the PDF Coordinate System.
LLY	Float	The vertical position for the lower-left corner of the TIFF. Uses the PDF Coordinate System.
Width	Float	The width of the TIFF specified in PDF Units. 0 = Uses the current width of the image specified with the FileName parameter. >0 = Overrides the current width of the image specified with the FileName parameter.

Height	Float	The height of the TIFF. Specified in PDF Units. 0 = Uses the current height of the image specified with the FileName parameter. >0 = Overrides the current height of the image specified with the FileName parameter.
PersistRatio	Long	1 = Persist height and width proportionately if width and height are greater than 0. 0 = Stretch to width and height.

SetHeaderVLine

Description

SetHeaderVLine instructs Toolkit to draw a line from a specified start point to an endpoint along a vertical axis. Toolkit draws the line on all pages affected by subsequent calls to MergeFile and CopyForm.

Return type

None

Syntax

object.**SetHeaderVLine** *StartY, EndY, X, Width*

The SetHeaderVLine method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
StartY	Float	The vertical position for the start point of the line. Uses the PDF Coordinate System.
EndY	Float	The vertical position for the endpoint of the line. Uses the PDF Coordinate System.
X	Float	The horizontal position for the line. Uses the PDF Coordinate System.
Width	Float	The width of the line. Specified in PDF Units.

SetHeaderWPgNbr

Description

SetHeaderWPgNbr instructs Toolkit to print text with special formatting, defined by a page-number format-string, beginning at the specified starting point. Toolkit prints the text on all pages affected by subsequent calls to MergeFile and CopyForm.

NOTE: You must call SetHeaderFont prior to SetHeaderWPgNbr. If you specify multiple fonts or font settings, the most recent setting will be used.

Return type

None

Syntax

object.**SetHeaderWPgNbr** *LLX, LLY, Text, FirstPageNbr*

The SetHeaderWPgNbr method has these required parts:

Part	**Value Type**	**Description**
Object		An expression of the Toolkit object
LLX	Float	The horizontal position for the lower-left corner of the area defined by the printed text. Uses the PDF Coordinate System.
LLY	Float	The vertical position for the lower-left corner of the area defined by the printed text. Uses the PDF Coordinate System.
Text	String	The text string printed in the defined area. This parameter permits one special page marker, *%p*, which corresponds to the current page number. Example usage: "Page *%p*" "Page *%p* of *totalpages*" **Note:** You can find the total number of pages using NumPages. If used, you must call NumPages prior to SetHeaderWPgNbr.
FirstPageNbr	Long	The page number used as the starting number.

Remarks

Toolkit will continue numbering per the last use of SetHeaderWPgNbr, removing any previously set format string. You can pass an empty string or call the ClearHeaderInfo method to stop numbering pages.

SetInfo

Description

Sets the PDF document properties including title, subject, author and keywords for the current open output file.

NOTE: You cannot set the creator or producer of the document.

Return type

None

Syntax

object.**SetInfo** *Title, Subject, Author, Keywords*

The SetInfo method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
Title	String	The title to set for the current open output file.
Subject	String	The subject to set for the current open output file.
Author	String	The author to set for the current open output file.
Keywords	String	The keywords to set for the current open output file. Keywords are comma delimited (No Space).

Example

```
'SetInfo Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("SetInfo.pdf")

      TK.SetFont "Helvetica", 20, 0
      TK.PrintText 30, 740, "Hello World", 0

      'Set PDF info
      TK.SetInfo "Hello World", "Testing", "John Doe", "test, hello"

TK.CloseOutputFile

Set TK = Nothing
```

SetInputPasswords

Description

Used to set any passwords required for opening an input document.

NOTE: If the document requires a User or Owner password, this method must be set before OpenInputFile.

Return type

None

Syntax

object.**SetInputPasswords** *UserPassword, OwnerPassword*

The SetInputPasswords method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
UserPassword	String	Case-sensitive password required to view the document.
OwnerPassword	String	Case-sensitive password required to modify or print document.

SetMasterQuery

Description

Sets the master query used when calling DBToForm.

Return type

None

Syntax

object.**SetMasterQuery** *ConnectionString, UserID, Password, Options, QueryString*

The SetMasterQuery method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
ConnectionString	String	The connection string used to connect to the database. For connection string examples, see remarks.
UserID	String	Set the user ID required by your connection.
Password	String	Set password required by your connection.
Options	Long	This should be set to -1.
QueryString	String	The SQL query string to execute.

Remarks

The following are some common examples of connection string values.

ConnectionString	Example Value
Using a named DSN	"DSN=MyDatabase;"
Microsoft Access using the ODBC	Driver"DBQ=C:\InetPub\database\donations.mdb;Driver={Microsoft Access Driver(*.mdb)};"
Microsoft Access using the Access OLEDB driver	"Provider=Microsoft.Jet.OLEDB.4.0;Data Source C:\InetPub\database\donations.mdb;"
SQL Server using the ODBC driver	"Driver={SQL Server};Server=activePDF;Database= pubs;Uid=sa; Pwd=;"
SQL Server using the	"PROVIDER=SQLOLEDB;DATA SOURCE=ServerName;DATABASE=

OLEDB driver	pubs;USER ID=sa;PASSWORD=;"

Example

```
APTOOLKIT.SetMasterQuery "DBQ=C:\InetPub\database\donations.mdb;Driver={Microsoft
Access Driver (*.mdb)};","
```

SetOutputArtBox

Description

Specifies the placement and size of the art box for the output file.

Return type

None

Syntax

object.**SetOutputArtBox** *LLX, LLY, URX, URY*

The SetOutputArtBox method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
LLX	Double	The horizontal position of the lower-left corner of the link. Uses the PDF Coordinate System.
LLY	Double	The vertical position of the lower-left corner of the link. Uses the PDF Coordinate System.
URX	Double	The horizontal position of the upper-right corner of the link. Uses the PDF Coordinate System.
URY	Double	The vertical position of the upper-right corner of the link. Uses the PDF Coordinate System.

Example

```
'SetOutputArtBox Example
Set TK = CreateObject("APToolkit.Object")

strPageWidth = 8.5 * 72 '72 = 1"
strPageHeight = 11 * 72 '72 = 1"
TK.OutputPageWidth = strPageWidth
TK.OutputPageHeight = strPageHeight

r = TK.OpenOutputFile("SetOutputArtBox.pdf")

      'Set the output PDF art box dimensions
      TK.SetOutputArtBox 30, 30, strPageWidth - 30, strPageHeight - 30

      TK.SetFont "Helvetica", 20, 0
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile
```

```
Set TK = Nothing
```

SetOutputBleedBox

Description

Sets the placement and size of the bleed box for the output file.

Return type

None

Syntax

object.**SetOutputBleedBox** *LLX, LLY, URX, URY*

The SetOutputBleedBox method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
LLX	Double	The horizontal position of the lower-left corner of the link. Uses the PDF Coordinate System.
LLY	Double	The vertical position of the lower-left corner of the link. Uses the PDF Coordinate System.
URX	Double	The horizontal position of the upper-right corner of the link. Uses the PDF Coordinate System.
URY	Double	The vertical position of the upper-right corner of the link. Uses the PDF Coordinate System.

Remarks

In a production or similar environment, a bleed area is designed to accommodate physical limitations of cutting, folding, and trimming equipment. The actual printed page may include printer marks that fall outside the bleed box.

Example

```
'SetOutputBleedBox Example
Set TK = CreateObject("APToolkit.Object")

strPageWidth = 8.5 * 72 '72 = 1"
strPageHeight = 11 * 72 '72 = 1"
TK.OutputPageWidth = strPageWidth
TK.OutputPageHeight = strPageHeight

r = TK.OpenOutputFile("SetOutputBleedBox.pdf")

      'Set the output PDF art box dimensions
```

```
        TK.SetOutputBleedBox 30, 30, strPageWidth - 30, strPageHeight - 30

        TK.SetFont "Helvetica", 20, 0
        TK.PrintText 30, 740, "Hello World", 0

    TK.CloseOutputFile

    Set TK = Nothing
```

SetOutputCropBox

Description

Sets the placement and size of the crop box for the output file.

Return type

None

Syntax

*object.***SetOutputCropBox** *LLX, LLY, URX, URY*

The SetOutputCropBox method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
LLX	Double	The horizontal position of the lower-left corner of the link. Uses the PDF Coordinate System.
LLY	Double	The vertical position of the lower-left corner of the link. Uses the PDF Coordinate System.
URX	Double	The horizontal position of the upper-right corner of the link. Uses the PDF Coordinate System.
URY	Double	The vertical position of the upper-right corner of the link. Uses the PDF Coordinate System.

Example

```
'SetOutputCropBox Example
Set TK = CreateObject("APToolkit.Object")

strPageWidth = 8.5 * 72 '72 = 1"
strPageHeight = 11 * 72 '72 = 1"
TK.OutputPageWidth = strPageWidth
TK.OutputPageHeight = strPageHeight

r = TK.OpenOutputFile("SetOutputCropBox.pdf")

      'Set the output PDF art box dimensions
      TK.SetOutputCropBox 30, 30, strPageWidth - 30, strPageHeight - 30

      TK.SetFont "Helvetica", 20, 0
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile
```

```
Set TK = Nothing
```

SetOutputRotation

Description

Sets the amount of rotation for the output PDF.

Return type

None

Syntax

*object.***SetOutputRotation** *Rotation*

The SetOutputRotation method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Rotation	Short	The amount of counterclockwise rotation in degrees. (Set less than 0 for clockwise rotation.)

Example

```
'SetOutputRotation Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("SetOutputRotation.pdf")

      'Set the output PDF rotation
      TK.SetOutputRotation 90

      TK.SetFont "Helvetica", 20, 0
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile

Set TK = Nothing
```

SetOutputSecurity

Description

Sets 40-bit security for the output file.

NOTE: SetOutputSecurity must be called before calling OpenOutputFile.

Syntax

object.**SetOutputSecurity** *UserPassword, OwnerPassword, CanPrint, CanEdit, CanCopy, CanModify*

The SetOutputSecurity method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
UserPassword	String	Case-sensitive password required to view the document. The maximum length for the password is 32 characters and cannot contain control characters. (If you are using the evaluation version of activePDF Toolkit, the prefix DEMO will be inserted before your password characters and count towards the 32-character maximum. For example, the password TEST will be DEMOTEST.)
OwnerPassword	String	Case-sensitive password required to modify or print document. The maximum length for the password is 32 characters and cannot contain control characters. The password will default to the UserPassword if left blank. (If you are using the evaluation version of activePDF Toolkit, the prefix DEMO will be inserted before your password characters and count towards the 32-character maximum. For example, the password TEST will be DEMOTEST.)
CanPrint	Long	Set to 1 to enable printing. Set to 0 to disable printing.
CanEdit	Long	Set to 1 to enable document editing. Set to 0 to disable document editing.
Can Copy	Long	Set to 1 to enable copying of text and graphics. Set to 0 to disable copying of text and graphics.
Can Modify	Long	Set to 1 to enable document modification.

		Set to 0 to disable document modification.

SetOutputSecurity128

Description

Sets 128-bit security for the output file.

NOTE: SetOutputSecurity128 must be called before calling OpenOutputFile. You must have *Strong Encryption*.

Return type

None

Syntax

object.**SetOutputSecurity128** *UserPassword, OwnerPassword, CanPrint, CanEdit, CanCopy, CanModify,CanFillInFormFields, CanMakeAccessible, CanAssemble, CanReproduce*

The SetOutputSecurity128 method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
UserPassword	String	Case-sensitive password required to view document. The maximum length for the password is 32 characters and cannot contain control characters. Once the password is set, it cannot be changed. (If you are using the evaluation version of activePDF Toolkit, the prefix DEMO will be inserted before your password characters and count towards the 32-character maximum. For example, the password TEST will be DEMOTEST.)
OwnerPassword	String	Case-sensitive password required to modify or print document. The maximum length for the password is 32 characters and cannot contain control characters. The password will default to the UserPassword if left blank. Once the password is set, it cannot be changed. (If you are using the evaluation version of activePDF Toolkit, the prefix DEMO will be inserted before your password characters and count towards the 32-character maximum. For example, the password TEST will be DEMOTEST.)
CanPrint	Long	Set to 1 to enable printing. Set to 0 to disable printing.
CanEdit	Long	Set to 1 to enable editing. Set to 0 to disable editing.

CanCopy	Long	Set to 1 to enable copying of text and graphics. Set to 0 to disable copying of text and graphics.
CanModify	Long	Set to 1 to enable document modifications. Set to 0 to disable document modification.
CanFillInFormFields	Long	Set to 1 to enable form field filling. Set to 0 to disable form field filling.
CanMakeAccessible	Long	Set to 1 to enable accessibility features. Set to 0 to disable accessibility features.
CanAssemble	Long	Set to 1 on an encrypted document to enable the user to insert, rotate or delete pages, and generate bookmarks or thumbnails even if CanModify is false. Set to 0 to disable document assembly.
CanReproduce	Long	Set to 1 on an encrypted document to enable the user to print a faithful reproduction of the PDF. Set to 0 to disable document reproduction. If this flag is 0 and CanPrint is 1, printing is limited to a low-resolution version.

SetOutputTrimBox

Description

Specifies the trim box of the output file.

Return type

None

Syntax

*object.***SetOutputTrimBox** *LLX, LLY, URX, URY*

The SetOutputTrimBox method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
LLX	Double	The horizontal position of the lower-left corner of the link. Uses the PDF Coordinate System.
LLY	Double	The vertical position of the lower-left corner of the link. Uses the PDF Coordinate System.
URX	Double	The horizontal position of the upper-right corner of the link. Uses the PDF Coordinate System.
URY	Double	The vertical position of the upper-right corner of the link. Uses the PDF Coordinate System.

Example

```
'SetOutputTrimBox Example
Set TK = CreateObject("APToolkit.Object")

strPageWidth = 8.5 * 72 '72 = 1"
strPageHeight = 11 * 72 '72 = 1"
TK.OutputPageWidth = strPageWidth
TK.OutputPageHeight = strPageHeight

r = TK.OpenOutputFile("SetOutputTrimBox.pdf")

      'Set the output PDF art box dimensions
      TK.SetOutputTrimBox 30, 30, strPageWidth - 30, strPageHeight - 30

      TK.SetFont "Helvetica", 20, 0
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile
```

```
Set TK = Nothing
```

SetTextColor

Description

Sets the RGB color used for text in all subsequent calls to PrintText and PrintMultilineText.

Return type

None

Syntax

object.**SetTextColor** *AmountRed, AmountGreen, AmountBlue, Greyscale, PageNr*

The SetTextColor method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
AmountRed	Short	The amount of red being applied to the text color. The value ranges from 0 to 255 with 255 being true red.
AmountGreen	Short	The amount of green being applied to the text color. The value ranges from 0 to 255 with 255 being true green.
AmountBlue	Short	The amount of blue being applied to the text color. The value ranges from 0 to 255 with 255 being true blue.
Greyscale	Short	The amount of white being applied to the text color. The value ranges from 0 to 255 with 255 being true white. (Setting the color to greyscale changes the internal color space to greyscale. Set the value to 0 for true black.)
PageNr	Long	Optional. 0 = The action will take place on the new or current open output page. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.

Example

```
'SetTextColor Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("SetTextColor.pdf")

      TK.SetFont "Helvetica", 20, 0

      'Set the color of the printed text
      TK.SetTextColor 50, 100, 255, 0, 0
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile

Set TK = Nothing
```

SetTextColorCMYK

Description

Sets the CMYK color used for text in all subsequent calls to PrintText and PrintMultilineText.

Return type

None

Syntax

object.**SetTextColorCMYK** *AmountCyan, AmountMagenta, AmountYellow, AmountBlack, PageNr*

The SetTextColorCMYK method has these required parts:

Part	**Value Type**	**Description**
Object		An expression of the Toolkit object.
AmountCyan	Short	The amount of cyan to be applied to the text color. The value ranges from 0 to 100 with 100 being true cyan.
AmountMagenta	Short	The amount of magenta to be applied to the text color. The value ranges from 0 to 100 with 100 being true magenta.
AmountYellow	Short	The amount of yellow to be applied to the text color. The value ranges from 0 to 100 with 100 being true yellow.
AmountBlack	Short	The amount of black to be applied to the text color. The value ranges from 0 to 100 with 100 being true black. (To reset color set all other colors to 0 and AmountBlack to 100).
PageNr	Long	Optional. 0 = The action will take place on the new or current open output page. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.

Example

```
'SetTextColorCMYK Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("SetTextColorCMYK.pdf")

      TK.SetFont "Helvetica", 20, 0

      'Set the color of the printed text
      TK.SetTextColorCMYK 50, 100, 24, 15, 0
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile

Set TK = Nothing
```

SetTextFillMode

Description

Specifies the fill mode for text in all subsequent calls to PrintText and PrintMultilineText.

Return type

None

Syntax

object.**SetTextFillMode** *FillMode*

The SetTextFillMode method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
FillMode	Short	The type of fill to apply to the text. 0 = Fill only. 1 = Stroke only. 2 = Fill then stroke. 3 = No fill or stroke.

Example

```
'SetTextFillMode Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("SetTextFillMode.pdf")

      'Set the fill mode of the subsequent printed text
      TK.SetTextFillMode 2

      TK.SetFont "Helvetica", 20, 0
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile

Set TK = Nothing
```

SetTextRotation

Description

Sets the desired rotation for text in all subsequent calls to PrintText.

Return type

None

Syntax

object.**SetTextRotation** *RotationAngle*

The SetTextRotation method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
RotationAngle	Short	The amount of counterclockwise rotation in degrees. (Set less than 0 for clockwise rotation.)

Example

```
'SetTextRotation Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("SetTextRotation.pdf")

      'Set the rotaion of the printed text
      TK.SetTextRotation -45

      TK.SetFont "Helvetica", 20, 0
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile

Set TK = Nothingb
```

SetTextStrokeColor

Description

Sets the RGB color of the stroke for all subsequent calls to PrintText and PrintMultilineText.

NOTE: To use SetTextStrokeColor, SetTextFillMode must be set equal to 1 or 2.

Return type

None

Syntax

object.**SetTextStrokeColor** *AmountRed, AmountGreen, AmountBlue, Greyscale, PageNr*

The SetTextStrokeColor method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
AmountRed	Short	The amount of red being applied to the text stroke color. The value ranges from 0 to 255 with 255 being true red.
AmountGreen	Short	The amount of green being applied to the text stroke color. The value ranges from 0 to 255 with 255 being true green.
AmountBlue	Short	The amount of blue being applied to the text stroke color. The value ranges from 0 to 255 with 255 being true blue.
Greyscale	Short	The amount of white being applied to the text stroke color. The value ranges from 0 to 255 with 255 being true white. (Setting the color to greyscale changes the internal color space to greyscale. Set the value to 0 for true black.)
PageNr	Long	Optional. 0 = The action will take place on the new or current open output page. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.

Example

```
'SetTextStrokeColor Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("SetTextStrokeColor.pdf")

        TK.SetTextFillMode 1
```

```
      'Set the stroke color of the printed text
      TK.SetTextStrokeCOlor 255, 100, 0, 0, 0

      TK.SetFont "Helvetica", 20, 0
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile

Set TK = Nothing
```

SetTextStrokeColorCMYK

Description

Sets the CMYK color of the stroke used for all subsequent calls to PrintText and PrintMultilineText.

Return type

None

Syntax

*object.***SetTextStrokeColorCMYK** *AmountCyan, AmountMagenta, AmountYellow, AmountBlack, PageNr*

The SetTextStrokeColorCMYK method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
AmountCyan	Short	The amount of cyan to be applied to the text stroke color. The value ranges from 0 to 100 with 100 being true cyan.
AmountMagenta	Short	The amount of magenta to be applied to the text stroke color. The value ranges from 0 to 100 with 100 being true magenta.
AmountYellow	Short	The amount of yellow to be applied to the text stroke color. The value ranges from 0 to 100 with 100 being true yellow.
AmountBlack	Short	The amount of black to be applied to the text stroke color. The value ranges from 0 to 100 with 100 being true black. (To reset color set all other colors to 0 and AmountBlack to 100).
PageNr	Long	Optional. 0 = The action will take place on the new or current open output page. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.

Example

```
'SetTextStrokeColorCMYK Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("SetTextStrokeColorCMYK.pdf")

      TK.SetTextFillMode 1
      'Set the stroke color of the printed text
      TK.SetTextStrokeCOlorCMYK 50, 50, 0, 0, 0

      TK.SetFont "Helvetica", 20, 0
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile

Set TK = Nothing
```

SetViewerPreferences

Description

Sets the initial viewer preferences used when the document is first opened in a PDF Viewer.

NOTE: Depending on the user settings, the viewer preferences may be different the second time a document is viewed. Some PDF Viewers may not support viewer preferences.

Return type

None

Syntax

object.**SetViewerPreferences** *HideToolbar, HideMenubar, HideWindowUI, FitWindow, CenterWindow*

The SetViewerPreferences method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
HideToolbar	Long	Toolbar will not be visible when PDF is first opened. 1 = The toolbar will not be visible when the PDF is opened. 0 = Defaults to the user preferences.
HideMenubar	Long	Menu bar will not be visible when PDF is first opened. 1 = The menu bar will not be visible when the PDF is opened. 0 = Defaults to the user preferences.
HideWindowUI	Long	User interface will not be visible when PDF is first opened. 1 = The user interface will not be visible when the PDF is opened. 0 = Defaults to the user preferences.
FitWindow	Long	FitWindow wraps the display window around the size of the document when it is opened. If the display window was previously maximized, the document will open based on the previous settings. 1 = The PDF will be opened in FitWindow mode. 0 = Defaults to the user preferences.
CenterWindow	Long	CenterWindow will center the Acrobat display window onscreen when the document is opened. 1 = The PDF will be opened in CenterWindow mode.

		0 = Defaults to the user preferences.

Example

```
'SetViewerPreferences Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("SetViewerPreferences.pdf")

      'Set the viewer preferences of the created PDF
      TK.SetViewerPreferences 1, 0, 1, 1, 0

      TK.SetFont "Helvetica", 20, 0
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile

Set TK = Nothing
```

SignExistingField

Description

Instructs Toolkit to sign and existing file visibly.

NOTE: Toolkit appends the signature to the file and does not modify the contents.

Return type

Long

Return Value	Description
-998	Product not registered/ Evaluation expired.
-997	Required product version not registered.
-25	Invalid internal PDF structure.
-13	Unable to read forms structure.
-12	Invalid internal forms reference.
-11	Invalid internal forms reference.
-10	Invalid Internal page structure.
-9	Invalid signature.
-8	Invalid signature number.
-1	Unable to open input file.
0	Success.

Syntax

object.**SignExistingField** *SigNumber, FileName, OutputFileName, FieldName, Location, Reason, ContactInfo, AppearanceInfo, Flags, AltText, AltTextFont, AltTextFontSize, SignatureType*

The SignExistingField method has these required parts:

Part	Value Type	Description
SigNumber	Long	The value returned from FindCertificate.

FileName	String	The full path to the file to be signed. If set to MEMORY then InputByteStream must be called first.
OutputFileName	String	The full path to where you want the output file stored. If set to a blank string ("") the file specified with FileName is overwritten. If set to "MEMORY" or if FileName = "MEMORY" and this parameter is set = "", an output byte stream is generated.
FieldName	String	The name of the field the signature is applied.
Location	String	The location where the signature is applied. Typically, this is city and state, or company location.
Reason	String	The reason for signing the document.
ContactInfo	String	Contact information of the signer.
AppearanceInfo	String	The AppearanceInfo parameter depends upon the value of Flags. 256 = AppearanceInfo must be the path to the image file (or MEMORY and set ImageByteStream = to the image in memory). 512 = AppearanceInfo must be the path to the PDF file (Not yet supported). 2048 = AppearanceInfo is the PDF command string to be inserted.
Flags	Long	A series of flags that can be combined via "or" statements: 1 = Show Common Name. 2 = Show Location. 4 = Show Distinguished Name. 8 = Show activePDF Logo. 16 = Show date. 32 = Show reason. 64 = Show labels. 256 = Set graphic to an image. 512 = Set graphic to a PDF (Not yet supported). 1024 = Set graphic to common name. 2048 = Set graphic to a PDF command stream. 4096 = Show checkmark/red x icons.
AltText	String	If set, this string will be printed in the signature field.
AltTextFont	String	The font name of the alternate text. Refer to SetFont.

AltTextFontSize	Float	The font size of the alternate text.
SignatureType	Long	0 = PKCS#1 Acrobat 4+ signature (best backwards compatibility). 1 = PKCS#7 Acrobat 4+ signature. 2 = VeriSign Signature (requires VeriSign plug-in. Certificate authority MUST be VeriSign). 3 = Microsoft Signature (Acrobat 6+).

Remarks

If the file is encrypted, you must call SetInputPasswords prior to calling SignExistingField.

Example

```
Set tk = CreateObject("APToolkit.Object")
retCode = TK.FindCertificate("Joe Kant","My",1)
If (retCode < 0) Then
   retCode = TK.CreateCertificate("Joe Kant", "Management", "activePDF","Mission
Viejo", "CA", "US", "joe@activepdf.com", 1, "My", 365,0,"","")
   retCode = TK.FindCertificate("Joe Kant","My",1)
   If (u < 0) Then
      MsgBox("Can't find it!")
   End If
End If
Flags = &H8 or &H256
r = tk.SignExistingField(retCode, "test.pdf", "", "SignatureField", "activePDF
Headquarters", "Our Document", "949-582-9002","sig.tif", Flags, "","",0, 72, 72,
144, 144, 1,0)
set TK = nothing
```

SignOutputFile

Description

Instructs Toolkit to sign the output file invisibly after any creation, merge or append operation.

Return type

None

Syntax

*object.***SignOutputFile** *SigNumber, Location, Reason, ContactInfo, SignatureType*

The SignOutputFile method has these required parts:

Part	Value Type	Description
SigNumber	Long	The value returned from FindCertificate.
Location	String	The location where the signature is applied. Typically, this is city and state or company location.
Reason	String	The reason for signing the document.
ContactInfo	String	Contact information of the signer.
SignatureType	Long	The signature type. 0 = PKCS#1 Acrobat 4+ signature (best backwards compatibility). 1 = PKCS#7 Acrobat 4+ signature. 2 = VeriSign Signature (requires VeriSign plug-in. Certificate authority MUST be VeriSign). 3 = Microsoft Signature (Acrobat 6+).

Remarks

SignOutputFile must be called after OpenOutputFile. Calling it before will clear out the certificate number.

Example

```
Set tk = CreateObject("APToolkit.Object")
retCode = TK.FindCertificate("Joe Kant","My",1)
If (retCode < 0) Then
retCode = TK.CreateCertificate("Joe Kant", "Management", "activePDF","Mission
Viejo", "CA", "US", "joe@activepdf.com", 1, "My", 365,0,"","")
retCode = TK.FindCertificate("Joe Kant","My",1)
If (u < 0) Then
MsgBox("Can't find it!")
```

```
End If
End If
r = tk.OpenOutputFile("output.pdf")
tk.SignOutputFile retCode, "activePDF Headquarters", "Our Document", "949-582-
9002",0
TK.SetFont "Helvetica",12
TK.PrintText 10,10,"This document should be signed."
Tk.CloseOutputFile
set TK = nothing
```

StitchPDF

Description

The StitchPDF method allows you to combine multiple PDFs onto a single page by specifying starting coordinates, size and rotation.

NOTE: Stitch operations works only on a blank page.

Return type

Short

Return Value	Description
-997	Required product version not registered.
<1	Unable to open input file.
>1	Success.

Syntax

object.**StitchPDF** *FileName, PageNumber, LLX, LLY, Width, Height, Rotation, PageNr*

The StitchPDF method has these parts

Part	Value Type	Description
Object		An expression of the Toolkit object.
FileName	String	The file name of the PDF to be placed in stitching.
PageNumber	Long	The number of the desired page from within the PDF to be placed in stitching.
LLX	Float	The horizontal position on the PDF page indicating where Toolkit will stitch the lower-left corner of the PDF you are adding. Uses the PDF Coordinate System.
LLY	Float	The vertical position on the PDF page indicating where Toolkit will stitch the lower-left corner of the PDF you are adding. Uses the PDF Coordinate System.
Width	Float	The desired width of the PDF you are adding. Set to 0 to use the width of the PDF specified with the FileName parameter. Set to anything else to override the width of the PDF specified with the FileName parameter.

Height	Float	The desired height of the PDF you are adding. Set to 0 to use the height of the PDF specified in FileName. Set to anything else to override the height of the PDF specified with the FileName parameter.
Rotation	Short	The amount of counterclockwise rotation in degrees. (Set less than 0 for clockwise rotation.)
PageNr	Long	Optional. 0 = The action will take place on the new or current open page of the input file or cover. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.

Remarks

To avoid conflict, any font names other than the 14 standard fonts will be renamed during the Stitch function.

TIFFToPDF

Description

Converts a TIFF image to PDF.

NOTE: If you require the resultant PDF to be encrypted, you will need to encrypt it after the PDF has been generated.

Return type

Long

Return Value	Description
-998	Product not registered/ Evaluation expired.
-997	Required product version not registered.
99	Unable to open input file.
97	Invalid TIFF specified.
1	Success.

Syntax

object.**TIFFToPDF** *ImageFileName, PDFFileName*

The TIFFToPDF method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
ImageFileName	String	The full path to the TIFF.
PDFFileName	String	The full path to the resultant PDF document.

ToPDFDate

Description

Converts a variant date to the PDF Date Format equivalent.

Return type

Converted String

Description
The variant date in PDF Date Format.

Syntax

object.**ToPDFDate** *(InDate)*

The ToPDFDate method has these required parts:

Part	Value	Description
Object		An expression of the Toolkit object.
InDate	String	The incoming date can include time.

Remarks

Refer to the FromPDFDate method and ModDate property.

VisiblySignFile

Description

Instructs Toolkit to dynamically create a form field and visibly sign an existing file.

NOTE: Toolkit appends the signature to the file and does not modify the contents.

Return type

Long

Return Value	Description
-998	Product not registered/ Evaluation expired.
-997	Required product version not registered.
-25	Invalid internal PDF structure.
-13	Unable to read forms structure.
-12	Invalid internal forms reference.
-11	Invalid internal forms reference.
-10	Invalid Internal page structure.
-9	Invalid signature.
-8	Invalid signature number.
-1	Unable to open input file.
0	Success.

Syntax

object.**VisiblySignFile** *SigNumber, FileName, FieldName, Location, Reason, ContactInfo, AppearanceInfo, Flags, AltText, AltTextFont, AltTextFontSize, LLX, LLY, Width, Height, pageNr, SignatureType*

The VisiblySignFile method has these required parts:

Part	Value Type	Description
SigNumber	Long	The value returned from FindCertificate.

FileName	String	The full path to the file to be signed. If set to MEMORY then InputByteStream must be called first.
FieldName	String	The full path to where you want the output file stored. If set to a blank string ("") the file specified with the FileName parameter is overwritten. If set to "MEMORY" of if FileName = "MEMORY" and this parameter is set = "", an output byte stream is generated.
Location	String	The location where the signature is applied. Typically, this is city and state or company location.
Reason	String	The reason for signing the document.
ContactInfo	String	Contact information of the signer.
AppearanceInfo	String	The AppearanceInfo parameter depends upon the value of Flags. 256 = AppearanceInfo must be the path to the image file (or MEMORY and set ImageByteStream = to the image in memory). 512 = AppearanceInfo must be the path to the PDF file (Not yet supported). 2048 = AppearanceInfo is the PDF command string to be inserted.
Flags	Long	A series of flags that can be combined via "or" statements: 1 = Show Common Name. 2 = Show Location. 4 = Show Distinguished Name. 8 = Show activePDF Logo. 16 = Show date. 32 = Show reason. 64 = Show labels. 256 = Set graphic to an image. 512 = Set graphic to a PDF (Not yet supported). 1024 = Set graphic to common name. 2048 = Set graphic to a PDF command stream. 4096 = Show checkmark/red x icons.
AltText	String	If set, this string will be printed in the signature field.
AltTextFont	String	The font name of the alternate text. Refer to SetFont.

AltTextFontSize	Float	The font size of the alternate text.
LLX	Float	The horizontal position of the lower-left corner of the signature. Uses the PDF Coordinate System.
LLY	Float	The vertical position of the lower-left corner of the signature. Uses the PDF Coordinate System.
Width	Float	Width of the signature field.
Height	Float	Height of the signature field.
PageNr	Long	Optional. 0 = The action will take place on the new or current open page of the input file or cover. (Default) >1 = The action will occur on the specified page number. -1 = The action will occur on all pages.
SignatureType	Long	0 = PKCS#1 Acrobat 4+ signature (best backwards compatibility). 1 = PKCS#7 Acrobat 4+ signature. 2 = VeriSign Signature (requires VeriSign plug-in. Certificate authority MUST be VeriSign). 3 = Microsoft Signature (Acrobat 6+).

Remarks

If the file is encrypted, you must call SetInputPasswords prior to calling VisiblySignFile.

Example

```
Set tk = CreateObject("APToolkit.Object")
retCode = TK.FindCertificate("Joe Kant","My",1)
If (retCode < 0) Then
   retCode = TK.CreateCertificate("Joe Kant", "Management", "activePDF","Mission
Viejo", "CA", "US", "joe@activepdf.com", 1, "My", 365,0,"0","0")
   retCode = TK.FindCertificate("Joe Kant","My",1)
   If (u < 0) Then
      MsgBox("Can't find it!")
   End If
End If
Flags = &H1 or &H256
r = tk.VisiblySignFile(retCode, "test.pdf", "", "activePDF Headquarters", "Our
Document", "949-582-9002","sig.tif", Flags, "","",0, 72, 72, 144, 144, 1,0)
set TK = nothing
```

WriteFingerprint

Description

WriteFingerprint appends an activePDF fingerprint to the end of the specified file. The resultant PDF is readable with any PDF viewer that adheres to the PDF specification, or by using activePDF Toolkit.

Return type

None

Syntax

object.**WriteFingerprint** *FileName*

The WriteFingerprint method has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
FileName	String	The name of the file to apply the activePDF fingerprint.

Remarks

Use IsFingerprintValid to determine if the fingerprint integrity is intact.

XMLSetFormFieldData

Description

XMLSetFormFieldData instructs Toolkit to populate the form fields of the currently open input file with XML data while writing the fields to the output file during the next call to CopyForm.

Return type

Long

Return Value	Description
-998	Product not registered/ Evaluation expired.
99	Unable to open input file.
97	Invalid path or file name specified.
1	Success.

Syntax

object.**XMLSetFormFieldData** *XMLData, DefaultFlag, Options, DefaultSeparator*

The XMLSetFormFieldData method has these required parts:

Part	Value Type	Description
XMLData	String	The XML data to be mapped.
DefaultFlag	Long	The default setting for the field: -4096 = All bits will be cleared (set to 0). You can 'OR' 4096 with other bits to achieve the desired effect. (This affects the line on which it is called.) -998 = Flatten field and reset font, color and rotation information to field defaults. (You must use -998 on the line prior to the line you wish to reset.) -997 = Flatten field and do not reset font, color and rotation information. -996 = Flatten field using an image file as named in field data. The image type is auto-determined. -995 = Flatten field as a known JPEG using an image file as named in field data. -994 = Flatten field as a known TIFF using an image file as named in field data. 0 = Read Only.

		1 = "As is". All attributes of the field remain unchanged. 2 = Hidden. 4 = Enable Printing. 8 = Disable Zoom. 16 = Disable Rotation. 32 = The field will print, but cannot be viewed. 64 = The field will be hidden and read only. Values can be "OR'ed" together: Flags = -4 or -64.
Options	Long	1 = ignore start and end tags. 0 = Do not ignore start and end tags.
DefaultSeparator	String	The separator to use in separating data records.

Properties

The Toolkit object has the following properties:

AddBookmarks

Description

If the specified input file contains bookmarks, AddBookmarks will pass the bookmarks to the output file.

NOTE: AddBookmarks must be called prior to OpenOutputFile.

Return type

Long

Syntax

object.**AddBookmarks** = *value*

The AddBookmarks property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	Long	1 = PDF bookmarks will be copied into the resultant PDF when calling the CopyForm method or MergeFile method. 0 = PDF bookmarks will not be copied.

Remarks

The output file will contain a top-level bookmark, which will be the path to the file or the value from the SetInputBookmark property. The existing bookmarks will appear as a sublevel of this top-level bookmark.

Author

Description

Returns the PDF author data.

NOTE: Must be called after GetPDFInfo.

Return type

String

Description
The name of the Author.

Syntax

*value = object.***Author**

The Author property has these parts:

Part	Description
Object	An expression of the Toolkit object.

BBHeight

Description

Returns the height of the bounding box after calling GetBoundingBox. The bounding box is the size of the printable area within the PDF.

Return type

Short

Description
The height of the bounding box.

Syntax

*value = object.***BBHeight**

The BBHeight property has these parts:

Part	Description
Object	An expression of the Toolkit object.

Remarks

You can use this property in conjunction with BBLeft, BBTop and BBWidth.

BBLeft

Description

Returns the left most coordinate of the bounding box after calling GetBoundingBox.

Return type

Short

Description
The left most position of the bounding box.

Syntax

value = *object*.**BBLeft**

The BBLeft property has these parts:

Part	Description
Object	An expression of the Toolkit object.

Remarks

You can use this property in conjunction with BBHeight, BBTop and BBWidth.

BBTop

Description

Returns the top coordinate of the bounding box calling GetBoundingBox.

Return type

Short

Description
The top of the bounding box.

Syntax

value = *object*.**BBTop**

The BBTop property has these parts:

Part	Description
Object	An expression of the Toolkit object.

Remarks

You can use this property in conjunction with BBHeight, BBLeft and BBWidth.

BBWidth

Description

Returns the width of the bounding box after calling GetBoundingBox.

Return type

Short

Description
The width of the bounding box.

Syntax

*value = object.***BBWidth**

The BBWidth property has these parts:

Part	Description
Object	An expression of the Toolkit object.

Remarks

You can use this property in conjunction with BBHeight, BBLeft and BBTop.

CharSpacing

Description

Manually sets the spacing between characters. This can be useful when the font definition is missing information, such as kerning.

NOTE: To set the space between words, use WordSpacing.

Return type

Float

Description
The current character spacing.

Syntax

*value = object.***CharSpacing** = *value*

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	Float	The float value is determined in PDF Units. Use WordSpacing to set the space between words manually.

Example

```
'CharSpacing Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("CharSpacing.pdf")

      'Set the spacing between characters
      TK.CharSpacing = 10

      TK.SetFont "Helvetica", 20, 0
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile

Set TK = Nothing
```

ClipText

Description

Returns any text that does not fit into the multi-line text field specified with PrintMultilineText.

Return type

String

Description
Any text that is clipped when flattening a multi-line text box.

Syntax

*value = object.***ClipText**

The ClipText property has these required parts:

Part	Description
Object	An expression of the Toolkit object.

Example

```
'ClipText Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("ClipText.pdf")

      'Make a large text string for the example.
      For i = 1 to 500
            strText = strText & "This is multiline text that is printed on the page.
"
      Next
      strText = strText & vbcr

      'Print as much text as possible on the first page
      TK.PrintMultilineText "Helvetica", 12, 30, 732, 552, 702, strText, 3, 0

      'Check to see if all the text was printed
      'If not loop adding a new page with remaining text until all text is printed
      Do Until TK.ClipText = ""
            TK.NewPage
            TK.PrintMultilineText "Helvetica", 12, 30, 762, 552, 732, TK.ClipText,
0, 0
      Loop

TK.CloseOutputFile

Set TK = Nothing
```

CompressImages

Description

Sets and retrieves the compression status of images.

Return type

Variant_Bool

Return Value	Description
True	Images are compressed.
False	Images are uncompressed.

Syntax

*value = object.***CompressImages** *= value*

The CompressImages property has these required parts:

Part	Value Type	Description
value	Variant_Bool	True = compress images. False = to leave images uncompressed. If left blank, returns a value indicating whether or not the image is compressed.

Example

```
'CompressImages Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("CompressImages.pdf")

      'Enable image compression for the created PDF
      TK.CompressImages = True

      TK.PrintJPEG "image.jpg", 0, 250, 0, 0, 1, 0

TK.CloseOutputFile

Set TK = Nothing
```

CreateDate

Description

Retrieves the internal PDF creation date.

NOTE: Must be called after GetPDFInfo.

Return type

String

Description
The internal PDF creation date in PDF Date Format.

Syntax

```
value = object.CreateDate
```

The CreateDate property has these parts:

Part	Description
Object	An expression of the Toolkit object.

Creator

Description

Retrieves the PDF creator information.

NOTE: Must be called after GetPDFInfo.

Return type

String

Description
The name of the PDF creator.

Syntax

*`value = object.`***`Creator`**

The Creator property has these parts:

Part	Description
Object	An expression of the Toolkit object.

CurrentLogoNumber

Description

CurrentLogoNumber enables you to select a specific page number other than the default first page when calling AddLogo or PrintLogo.

Return type

Short

Syntax

object.**CurrentLogoNumber** = *value*

The CurrentLogoNumber property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	Short	The specified page number other than the default 1.

CustomDocInfo

Description

CustomDocInfo enables you to retrieve and set the PDF custom document information fields when merging or copying a PDF document. Common fields used with the CustomDocInfo property are *DocVersion*, *URL*, *LogonID* and *Cookie Value*.

NOTE: To retrieve data that is set with CustomDocInfo, you will need to call GetPDFInfo prior to calling CustomDocInfo.

Return type

String

Description
The information contained in a custom document field.

Syntax

*value = object.***CustomDocInfo***(ItemName) = value*

The CustomDocInfo property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
ItemName		The specified field.
Value	String	The information to set in a custom document field.

Remarks

If you want to access one of the standard fields, use the corresponding Toolkit property such as Author or Title.

DebugMode

Description

If set, generates a log file used by the activePDF Technical Support Team when debugging Toolkit.

Return type

Long

Return Value	Description
0	Debug mode is off. (Default)
1	Debug mode is on.

Syntax

value = *object*.**DebugMode** = *value*

The DebugMode property has these required parts:

Part	Value Type	Description
Value	Long	0 = Debug Mode is Off (Default). 1 = Debug Mode is On.

DoFormFormatting

Description

Sets whether the designated form field output format will be recognized. Form field output formats include date, numeric, currency, percentage and other formats.

NOTE: You must call DoFormFormatting prior to SetFormFieldData.

Return type

Long

Syntax

`object.DoFormFormatting = value`

The DoFormFormatting property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	Long	1 = Settings are automatically recognized and the fields will be formatted according to output. 0 = Settings will not be automatically recognized and the fields will be formatted according to output.

EncryptionError

Description

This property is used by activePDF Technical Support to assist in troubleshooting.

NOTE: You will only need to call EncryptionError when instructed by an activePDF Technical Support Engineer.

Return type

Long

Description
A reference code pertinent to activePDF Technical Support.

Syntax

```
value1 = object.EncryptionError
```

The EncryptionError property has these required parts:

Part	Description
Object	An expression of the Toolkit object.

Explorer

Description

Passes an instance to the property and an Explorer object is returned for the current input PDF.

Return type

Object.

Description
The explorer object of the current input PDF.

Syntax

value = *object.***Explorer**

The Explorer property has these required parts:

Part	Description
Object	An expression of the Toolkit object.

FingerprintOutputFile

Description

Generates a unique fingerprint based on the file content. Generates a unique fingerprint based on the file content and applies it to the output file. A fingerprint is a hash generated from the contents of the PDF, which is appended to the end of the output PDF, enabling you to verify the integrity of the file contents.

Syntax

object.**FingerprintOutputFile** = *value*

The FingerprintOutputFile property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
Value	Long	1 = The fingerprint will be written to the end of the file. 0 = The fingerprint will not be written to the end of the file. (Default.)

Remarks

You can verify the integrity of a fingerprint using IsFingerprintValid.

FlattenRemainingFormFields

Description

Flattens any fields that you have not explicitly set. This is useful when you reduced file size is important.

NOTE: This is equivalent to calling SetFormFieldData `FieldName,"", -998` for any fields you do not explicitly set.

Return type

Long

Syntax

`object.FlattenRemainingFormFields = value`

The FlattenRemainingFormFields property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
Value	Long	1 = Flattens the remaining form fields. 0 = Does not flatten the remaining form fields.

FormNumbering

Description

Assigns a form number to form fields in the output file.

NOTE: Must be called prior to CopyForm.

Return type

Short

Description
The assigned form number for the specified field.

Syntax

object.**FormNumbering** = *value*

The FormNumbering property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
Value	Short	If you set FormNumbering equal to a number greater than zero, Toolkit renames the field on the first copy to: NAME__1 (2 underscores) ADDRESS__1 On the second copy: NAME__2 (2 underscores) ADDRESS__2 The number increments with each copy.

Remarks

Using Toolkit you can copy the same form multiple times, which saves space internally in the PDF by only moving the data. If your fields are not marked “read-only” and the form fields are set to different values, any output fields with the same name will be overwritten when a user enters data into the first instance of the named field. To prevent this from occurring, you can use the FormNumbering property.

HeightPrinted

Description

Returns the height of printed text in PDF Units.

Return type

Float

Description
The height of the printed text in PDF Units.

Syntax

*value = object.***HeightPrinted**

The HeightPrinted property has these required parts:

Part	Description
Object	An expression of the Toolkit object.

Example

```
'HeightPrinted Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("HeightPrinted.pdf")

      strText = "This is multiline text that is printed on the page"
      TK.PrintMultilineText "Helvetica", -20, 30, 650, 50, 80, strText, 0, 0

      strTextHeight = TK.HeightPrinted
      msgbox strTextHeight

      TK.SetFont "Helvetica", 20, 0
      TK.PrintText 30, 740, "Hello World", 0

      strTextHeight = TK.HeightPrinted
      msgbox strTextHeight

TK.CloseOutputFile

Set TK = Nothing
```

ImageByteArray (.NET only)

Description

Specifies an image file in binary format to be used with Toolkit's image methods, for "in-memory" generation.

NOTE: This property is intended for use in a .NET environment. Refer to the ImageByteStream property if you are implementing activePDF Toolkit an environment other than .NET.

Syntax

object.**ImageByteArray** = *value*

The ImageByteArray property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	Safearray	The specified safearray.

Remarks

Use ImageByteArray when setting the file name to “MEMORY”, using any of the following graphics functions:

- SetHeaderJPEG.
- SetHeaderTIFF.
- PrintJPEG.
- PrintTIFF.
- SetHeaderImage.
- PrintImage.

Example C#

```
APToolkitNET.Toolkit TK = new APToolkitNET.Toolkit();

TK.OpenOutputFile("MEMORY");
TK.SetFont("Helvetica", 24);
TK.PrintText(100, 600, "This is a test");
TK.CloseOutputFile();

// assign a byte array image of
// the output file to binImage
byte[] binImage = TK.BinaryImage;

// open a new output file to disk
TK.OpenOutputFile("output.pdf");

// use the binImage variable to
// populate the input byte stream
// using InputByteArray
TK.InputByteArray = binImage;
TK.OpenInputFile("MEMORY");
TK.CopyForm(0, 0);
TK.CloseOutputFile();
```

ImageByteStream

Description

Specifies an image file in binary format to be used with Toolkit's Image methods for "in-memory" generation.

Syntax

```
object.ImageByteStream = value
```

The ImageByteStream property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	Variant	The value to assign it can be either a String or a Safearray (array of integers representing the image).

Remarks

Use ImageByteStream property when setting the filename to memory in any of the graphics functions:

- SetHeaderJPEG.
- SetHeaderTIFF.
- PrintJPEG.
- PrintTIFF.
- SetHeaderImage.
- PrintImage.

ImageRotation

Description

Sets or returns the rotation for the specified image.

Return type

Short

Description
The current rotation for the specified image.

Syntax

value = *object*.**ImageRotation** = *value*

The ImageRotation property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	Long	The value is entered as a multiple of 90°. 90, 180, 270, 360, -90, etc. If a positive value is used, the image will rotate counter-clockwise.

Remarks

ImageRotatation can be used with the following methods:

- PrintImage.
- PrintJPEG.
- PrintTIFF.
- SetHeaderImage.
- SetHeaderJPEG.
- SetHeaderTIFF.

Example

```
'ImageRotation Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("ImageRotation.pdf")

      'Rotate the image 180 degrees
      TK.ImageRotation = 180

      TK.PrintJPEG "image.jpg", 0, 250, 0, 0, 1, 0
TK.CloseOutputFile

Set TK = Nothing
```

InputByteArray (.NET only)

Description

Specifies a PDF file in binary format to be used with OpenInputFile, for "in-memory" generation.

NOTE: This property is intended for use in a .NET environment. Refer to the InputByteStream property if you are implementing activePDF Toolkit an environment other than .NET.

Syntax

*value = object.***InputByteArray** *= value*

The InputByteArray property has these parts:

Part	**Value Type**	**Description**
Object		An expression of the Toolkit object.
Value	Safearray	An array.

Remarks

When setting the file name to “MEMORY”, you must call InputByteArray prior to calling OpenInputFile or MergeFile.

Example C#

```
APToolkitNET.Toolkit TK = new APToolkitNET.Toolkit();

TK.OpenOutputFile("MEMORY");
TK.SetFont("Helvetica", 24);
TK.PrintText(100, 600, "This is a test");
TK.CloseOutputFile();

// assign a byte array image of
// the output file to binImage
byte[] binImage = TK.BinaryImage;

// open a new output file to disk
TK.OpenOutputFile("output.pdf");

// use the binImage variable to
// populate the input byte stream
// using InputByteArray
TK.InputByteArray = binImage;
TK.OpenInputFile("MEMORY");
TK.CopyForm(0, 0);
TK.CloseOutputFile();
```

InputByteStream

Description

Specifies a PDF file in binary format to be used with Toolkit's OpenInputFile method for "in-memory" generation.

NOTE: If you call OpenInputFile or MergeFile and set the file name to "MEMORY", you must set this property prior to calling either of those requirements.

Syntax

*value = object.***InputByteStream** *= value*

The InputByteStream property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	Variant	The value to assign it can be either a String or a Safearray data type.

InputCanAssemble

Description

Detects whether the currently open input file can be assembled.

Return type

Variant_Bool

Return Value	Description
True	File can be assembled.
False	File cannot be assembled.

Syntax

`value = object.``**InputCanAssemble**`

The InputCanAssemble property has these parts:

Part	Description
Object	An expression of the Toolkit object.

InputCanCopy

Description

Detects whether the currently open input file can be copied.

Return type

Variant_Bool

Return Value	Description
True	File can be copied.
False	File cannot be copied.

Syntax

`value = object.InputCanCopy`

The InputCanCopy property has these parts:

Part	Description
Object	An expression of the Toolkit object.

InputCanEdit

Description

Detects whether the currently open input file can be edited.

Return type

Variant_Bool

Return Value	Description
True	File can be edited.
False	File cannot be edited.

Syntax

*value = object.***InputCanEdit**

The InputCanEdit property has these parts:

Part	Description
Object	An expression of the Toolkit object.

InputCanFillInFormFields

Description

Detects whether the currently open input file allows form fields to be filled.

Return type

Variant_Bool

Return Value	Description
True	File allows form field filling.
False	File does not allow form field filling.

Syntax

*value = object.***InputCanFillInFormFields**

The InputCanFillInFormFields property has these parts:

Part	Description
Object	An expression of the Toolkit object.

InputCanMakeAccessible

Description

Detects whether the currently open input file can be made accessible.

Return type

Variant_Bool

Return Value	Description
True	File can be made accessible.
False	File cannot be made accessible.

Syntax

*value = object.***InputCanMakeAccessible**

The InputCanMakeAccessible property has these parts:

Part	Description
Object	An expression of the Toolkit object.

InputCanModify

Description

Detects whether the currently open input file can be modified.

Return type

Variant_Bool

Return Value	Description
True	File can be modified.
False	File cannot be modified.

Syntax

*value = object.***InputCanModify**

The InputCanModify property has these parts:

Part	Description
Object	An expression of the Toolkit object.

InputCanPrint

Description

Detects whether the currently open input file is allowed to be printed.

Return type

Variant_Bool

Return Value	Description
True	File allows printing.
False	File does not allow printing.

Syntax

*value = object.***InputCanPrint**

The InputCanPrint property has these parts:

Part	Description
Object	An expression of the Toolkit object.

InputCanReproduce

Description

Detects whether the currently open input file can be reproduced.

Return type

Variant_Bool

Return Value	Description
True	File can be reproduced.
False	File cannot be reproduced.

Syntax

*value = object.***InputCanReproduce**

The InputCanReproduce property has these parts:

Part	Description
Object	An expression of the Toolkit object.

InputKeySize

Description

If an input file is encrypted, returns the level of encryption.

Return type

Short

Return Value	Description
0	No encryption.
40	40-bit encryption.
128	128-bit encryption.

Syntax

*value = object.***InputKeySize**

The InputKeySize property has these required parts:

Part	Description
Object	An expression of the Toolkit object.

JPEGMemoryAllocationSize

Description

Sets and returns the memory allocation size when working with JPEGs.

Return type

Long

Description
The current file size allocation.

Syntax

*value = object.***JPEGMemoryAllocationSize** *= value*

The JPEGMemoryAllocationSize property has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	Long	In bytes. Default is 16384.

Remarks

If you are working with larger JPEG files, increasing this may improve performance. Conversely, if you are working with smaller JPEGs, decreasing this may improve performance.

Example

```
'JPEGMemoryAllocationSize Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("JPEGMemoryAllocationSize.pdf")

      'Check the current memory size for JPEG images
      'if less than or equal to 16384 set it to 32768
      If TK.JPEGMemoryAllocationSize <= 16384 Then
            TK.JPEGMemoryAllocationSize = 32768
      End If

      TK.PrintJPEG "image.jpg", 0, 250, 0, 0, 1, 0

TK.CloseOutputFile

Set TK = Nothing
```

Keywords

Description

Returns the Keyword data from a PDF.

NOTE: Must be called after GetPDFInfo.

Return type

String

Description
The information contained in the keywords PDF document field.

Syntax

*`value = object.`***`Keywords`**

The Keywords property has these parts:

Part	Description
Object	An expression of the Toolkit object.

MaxAutoMultiLineSize

Description

Sets or retrieves the maximum font size used when auto-calculating the font size for printed text.

Return type

Float

Description
The current maximum font size used when auto-calculating.

Syntax

*value = object.***MaxAutoMultiLineSize** *= value*

The MaxAutoMultiLineSize property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	Float	The maximum font size to use when auto-calculating.

Example

```
'MaxAutoMultiLineSize Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("MaxAutoMultiLineSize.pdf")

      'Set the maximum size of the auto-size
      TK.MaxAutoMultiLineSize = 20

      strText = "This is multiline text that is printed on the page"
      TK.PrintMultilineText "Helvetica", -20, 30, 650, 50, 80, strText, 0, 0

TK.CloseOutputFile

Set TK = Nothing
```

MemoryFileAllocationSize

Description

Sets and returns the memory file size allocation.

Return type

Long

Description
The current file size allocation.

Syntax

*value = object.***MemoryFileAllocationSize** *= value*

The MemoryFileAllocationSize property has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object
Value	Long	In bytes. Default is 1024.

Remarks

If this value is set too high, it may be wasting memory. If the value is set too low, performance may be decreased.

Example

```
'MemoryFileAllocationSize Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("MemoryFileAllocationSize.pdf")

      'Check the current memory size for files
      'if less than or equal to 1024 set it to 2048
      If TK.MemoryFileAllocationSize <= 1024 Then
            TK.MemoryFileAllocationSize = 2048
      End If

      TK.SetFont "Helvetica", 20, 0
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile

Set TK = Nothing
```

Metadata

Description

Sets and retrieves the metadata from a specified PDF page.

Return type

String

Description
The metadata contained on the specified PDF page.

Syntax

*value = object.***Metadata** *PageNum*

The Metadata property has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
PageNum	Long	The page number to retrieve the metadata.

ModDate

Description

Returns the internal PDF modification date.

NOTE: Must be called after GetPDFInfo.

Return type

String

Description
The date is stored as a string value in PDF Date Format.

Syntax

value = *object*.**ModDate** = *value*

The ModDate property has these parts:

Part	Description
Object	An expression of the Toolkit object.

Remarks

You can use ModDate in conjunction with FromPDFDate to convert the value to a variant date field

MultilineSpacing

Description

Sets or retrieves the line spacing used when calling PrintMultilineText.

Return type

Float

Description
The current line spacing.

Syntax

*value = object.***MultilineSpacing** *= value*

The MultilineSpacing property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	Float	The line spacing to be used.

NeedAppearances

Description

Retains the existing field appearance stream when SetFormFieldData is called.

Return type

None

Syntax

object.**NeedAppearances** = *value*

The NeedAppearances property has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	Long	1 = Generate appearance streams. 0 = Do not generate appearance streams. (Default).

OutputByteStream

Description

Retains the entire PDF as a string variable after you call CloseOutputFile and set the output file name to "MEMORY".

Return Type

String

Description
The PDF from “MEMORY”.

Syntax

`*Value* = *object*.**OutputByteStream**`

The OutputByteStream property has these parts:

Part	Description
Object	An expression of the Toolkit object.

Remarks

You can use this string with InputByteStream or store it to a database field. To deliver data to the client's browser in ASP, use the BinaryImage method.

Example

```
'OutputByteStream Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("MEMORY")

      TK.SetFont "Helvetica", 20, 0
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile

'Set the memory PDF to a string to insert in a DB
'or stream to the clients browser
strPDF = TK.OutputByteStream

Set TK = Nothing
```

OutputLinearized

Description

Sets whether or not to linearize when closing the document.

NOTE: OutputLinearized must be called before OpenOutputFile. If encryption is turned on, you must call SetInputPasswords during the CopyForm or MergeFile operation. Otherwise, you must pass the user password to the linearization routine.

Syntax

object.**OutputLinearized** = *value*

The OutputLinearized property has these parts:

Part	**Value Type**	**Description**
Object		An expression of the Toolkit object.
Value	Long	1 = The file will be linearized. 0 = The file will not be linearized. (Default.)

OutputPageHeight

Description

Sets or returns the output page height.

NOTE: You must set OutputPageHeight before calling OpenOutputFile.

Return type

Float

Description
The height in PDF Units.

Syntax

*value = object.***OutputPageHeight** *= value*

The OutputPageHeight property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	Float	The height in PDF Units. (Default is 792)

Remarks

OutputPageHeight can only be used in conjunction with methods like PrintText, DrawTo and PrintLogo , which generate a new PDF page in the output document.

Example

```
'OutputPageHeight Example
Set TK = CreateObject("APToolkit.Object")

'Set page dimensions (must go before OpenOutputFile)
strPageWidth = 8.5 * 72 '72 = 1"
strPageHeight = 14 * 72 '72 = 1"
TK.OutputPageWidth = strPageWidth
TK.OutputPageHeight = strPageHeight

r = TK.OpenOutputFile("OutputPageHeight.pdf")

      TK.SetFont "Helvetica", 20, 0
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile
```

```
Set TK = Nothing
```

OutputPageWidth

Description

Sets or returns the output page width.

NOTE: You must set OutputPageWidth before calling OpenOutputFile.

Return type

Float

Description
The width in PDF Units.

Syntax

*value = object.***OutputPageWidth** *= value*

The OutputPageWidth property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	Float	The width in PDF Units. (Default is 792)

Remarks

OutputPageHeight can only be used in conjunction with methods like PrintText, DrawTo and PrintLogo , which generate a new PDF page in the output document.

Example

```
'OutputPageWidth Example
Set TK = CreateObject("APToolkit.Object")

'Set page dimensions (must go before OpenOutputFile)
strPageWidth = 11 * 72 '72 = 1"
strPageHeight = 8.5 * 72 '72 = 1"
TK.OutputPageWidth = strPageWidth
TK.OutputPageHeight = strPageHeight

r = TK.OpenOutputFile("OutputPageWidth.pdf")

      TK.SetFont "Helvetica", 20, 0
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile

Set TK = Nothing
```

PDFVersion

Description

Returns the PDF version for the current input document when called after OpenInputFile and sets the PDF version for the output document when called before OpenOutputFile.

NOTE: If OpenInputFile is called before OpenOutputFile, the PDF version from the Input file will be applied to the output file.

Return type

String

Return Value	Description
1.1	Legacy programs.
1.2	Acrobat 3.x and some functionality of 4.x.
1.3	Acrobat 4.x.
1.4	Acrobat 5.x.
1.5	Acrobat 6.x and 7.x

Syntax

value = *object*.**PDFVersion** = *value*

The PDFVersion property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	String	1.1 - Legacy programs. 1.2 - Acrobat 3.x and some functionality of 4.x. 1.3 - Acrobat 4.x. 1.4 - Acrobat 5.x. 1.5 – Acrobat 6.x and 7.x

Remarks

If you merge documents that contain functionality specific to a higher PDF version and set the resultant PDF to a lower version number, you may experience unpredictable results.

Example

```
'PDFVersion Example
Set TK = CreateObject("APToolkit.Object")

'Set the version of the PDF being created
TK.PDFVersion = 1.3

r = TK.OpenOutputFile("PDFVersion.pdf")

      TK.SetFont "Helvetica", 20, 0
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile

Set TK = Nothing
```

Producer

Description

Returns the PDF producer data.

NOTE: Must be called after GetPDFInfo.

Return type

String

Description
The data stored in the Producer field.

Syntax

`value = object.Producer`

The Producer property has these parts:

Part	Description
Object	An expression of the Toolkit object.

ReadOnlyOnMerge

Description

Overrides Toolkit's default behavior to set a *ReadOnly* flag on any fields not explicitly set by SetFormFieldData during a merge.

Return type

Long

Syntax

`object``.`**`ReadOnlyOnMerge`**` = `*`value`*

The ReadOnlyOnMerge property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	Long	1 = Keep the default ReadOnly behavior. 0 = Override the default ReadOnly behavior.

RelatedQuerySeparator

Description

Overrides the default pipe character ("|") used by AddRelatedQuery when performing parametric replacements in a SQL string.

NOTE: This can be useful if you assign the pipe character to another purpose in your database.

Syntax

object.**RelatedQuerySeparator** = *value*

The RelatedQuerySeparator property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	String	The desired override for the related query replacement separator. (Default is: "|")

RemoveDuplicateObjects

Description

If set to true, RemoveDuplicateObjects instructs Toolkit to remove duplicate objects when closing the output file. You can also retrieve the status of this property.

Return type

Variant Bool

Return Value	Description
True	Remove Duplicate objects is currently on.
False	Remove Duplicate objects is currently off.

Syntax

*value = object.***RemoveDuplicateObjects** *= value*

The RemoveDuplicateObjects property has these required parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	Variant_Bool	False = Duplicate objects will not be removed. (Default) True = Duplicate objects will be removed.

Example

```
'RemoveDuplicateObjects Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("RemoveDuplicateObjects.pdf")

      'Enable the removal of any duplicate objects
      TK.RemoveDuplicateObjects = True

      TK.SetFont "Helvetica", 20, 0
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile

Set TK = Nothing
```

RemoveWhiteSpace

Description

Removes the white space that may appear around imported images. The white space will be rendered as transparent allowing the any underlying images or text to remain visible.

NOTE: The quality of the resultant image is affected by the method of diffusion used to generate the TIFF.

Return type

Long

Syntax

object.**RemoveWhiteSpace** = *value*

The RemoveWhiteSpace property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	Long	1 = The white space in imported TIFF files will be rendered transparent. 0 = The white space will remain as is.

SetInputBookmark

Description

Sets the name value for the top-level bookmark generated when AddBookmarks is equal to 1.

Syntax

```
object.SetInputBookmark = value
```

The SetInputBookmark property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	String	The name of the top-level bookmark generated when AddBookmarks is equal to 1.

SpecialFlags

Description

Sets flags used to calculate TIFF data stream size versus posted size when using PrintTIFF, SetHeaderTIFF or TIFFToPDF. SpecialFlags can also be used when re-encoding a TIFF as a JPEG.

Return type

Long

Description
The flag assigned to the field.

Syntax

object.**SpecialFlags** = *value*

The SpecialFlags property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	Long	256 = Calculate TIFF data on stream size versus table posted size. 4096 = Re-encode TIFF as JPEG.

Subject

Description

Returns the PDF subject data.

NOTE: Must be called after GetPDFInfo.

Return type

String

Description
The data contained in the Subject Information.

Syntax

```
value = object.Subject = value
```

The Subject property has these parts:

Part	Description
Object	An expression of the Toolkit object.

SubsetFonts

Description

Sets or returns whether or not embedded fonts will be subset in the output PDF.

Return type

Variant Bool

Return Value	**Description**
False	Fonts are not subset.
True	Fonts are subset.

Syntax

*value = object.***SubsetFonts** *= value*

The SubsetFonts property has these required parts:

Part	**Value Type**	**Description**
Object		An expression of the Toolkit object.
Value	Variant_Bool	False = Fonts will not be subset. (Default) True = Fonts will be subset.

Title

Description

Returns the PDF Title data.

NOTE: Must be called after GetPDFInfo.

Return type

String

Description
The data contained in the Title Information.

Syntax

value = object.**Title**

The Title property has these parts:

Part	Description
Object	An expression of the Toolkit object.

ToolkitVersion

Description

Returns the version of Toolkit used to run the current script.

Return type

String

Description
The version of Toolkit being used.

Syntax

*value = object.***ToolkitVersion**

The ToolkitVersion property has these parts:

Part	Description
Object	An expression of the Toolkit object.

Example

```
'ToolkitVersion Example
Set TK = CreateObject("APToolkit.Object")

'Set the version of the PDF being created
strTKVer = TK.ToolkitVersion
msgbox strTKVer

Set TK = Nothing
```

WordSpacing

Description

Manually sets the spacing between words. This can be useful when the font definition is missing information such as kerning.

NOTE: Use CharSpacing to set the space between characters manually.

Return type

Float

Description
The spacing between words in PDF Units.

Syntax

object.**WordSpacing** = *value*

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	Float	The spacing between words in PDF Units.

Example

```
'WordSpacing Example
Set TK = CreateObject("APToolkit.Object")

r = TK.OpenOutputFile("WordSpacing.pdf")

      'Set the spacing between characters
      TK.WordSpacing = 25

      TK.SetFont "Helvetica", 20, 0
      TK.PrintText 30, 740, "Hello World", 0

TK.CloseOutputFile

Set TK = Nothing
```

WordWrapBuffer

Description

Sets or returns the buffer used on all form fields when using SetFormFieldData. The buffer specified is placed between the right side of the form field and the right most text.

Return type

Float

Description
The buffer specified in PDF Units.

Syntax

object.**WordWrapBuffer** = *value*

The WordWrapBuffer property has these parts:

Part	Value Type	Description
Object		An expression of the Toolkit object.
Value	Float	The buffer. Specified in PDF Units.

PDFFieldInfo

PDFFieldInfo is a subobject of the Toolkit object, which sets and retrieves a predefined set of methods or properties about a specified form field.

This section includes the following:

- Instantiating the PDFFieldInfo subobject.
- Method.
- Properties.

Instantiating the PDFFieldInfo Subobject

The PDFFieldInfo subobject is created by passing a field name and field instance to the Toolkit object FieldInfo method.

```
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set FIO = TK.FieldInfo("test", 1)
```

Method

The PDFFieldInfo subobject contains the ListItems method.

ListItems

Description

Passes an instance to the property and a ListItems object is returned for the current input PDF.

Return type

Object

Description
The ListItems object for the current input PDF.

Syntax

*value = object.***`ListItems`**

The ListItems method has this part:

Part	Description
Object	An expression of the PDFFieldInfo subobject.

Properties

The PDFFieldInfo subobject has the following properties:

ActionScript

Description

Returns any JavaScript code set to activate upon execution of an event for a particular form field.

Return type

String

Description
The JavaScript code.

Syntax

*value = object.***ActionScript**

The ActionScript property has this part:

Part	Description
Object	An expression of the PDFFieldInfo subobject.

Example

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set FIO = TK.FieldInfo("test", 1)
MsgBox FIO.ActionScript
Set FIO = Nothing
Set TK = Nothing
```

Alignment

Description

Sets and retrieves the alignment for the specified field.

Return type

Short

Return Value	**Description**
0	The current alignment is left.
1	The current alignment is center.
2	The current alignment is right.
3	The current alignment is full justified.

Syntax

*value = object.***Alignment** *= value*

The Alignment property has these required parts:

Part	**Value Type**	**Description**
Object		An expression of the PDFFieldInfo subobject.
Value	Short	0 = Left. 1 = Center. 2 = Right. 3 = Justified Full.

Remarks

If justified is selected for a multi-line field, the last line of the field will not be justified. If you require the last line to be justified, you can pass a carriage return.

Example

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set FIO = TK.FieldInfo("test", 1)
MsgBox FIO.Alignment
Set FIO = Nothing
Set TK = Nothing
```

AmountBlue

Description

Sets and retrieves the amount of blue in a field.

Return type

Short

Description
The amount of blue. The value ranges from 0 to 255 with 255 being true blue.

Syntax

value = *object*.**AmountBlue** = *value*

The AmountBlue property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	The amount of blue. The value ranges from 0 to 255 with 255 being true blue.

Remarks

You can use this in conjunction with the AmountGreen and AmountRed properties.

Example

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set FIO = TK.FieldInfo("test", 1)
MsgBox FIO.AmountBlue
Set FIO = Nothing
Set TK = Nothing
```

AmountGreen

Description

Sets and retrieves the amount of green in a field.

Return type

Short

Description
The amount of green. The value ranges from 0 to 255 with 255 being true green.

Syntax

*value = object.***AmountGreen** *= value*

The AmountGreen property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Short	The amount of green. The value ranges from 0 to 255 with 255 being true green.

Remarks

You can use this in conjunction with the AmountBlue and AmountRed properties.

Example

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set FIO = TK.FieldInfo("test", 1)
MsgBox FIO.AmountGreen
Set FIO = Nothing
Set TK = Nothing
```

AmountRed

Description

Sets and retrieves the amount of red in a field.

Return type

Short

Description
The amount of red. The value ranges from 0 to 255 with 255 being true red.

Syntax

*value = object.***AmountRed** *= value*

The AmountRed property has these parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Short	The amount of red. The value ranges from 0 to 255 with 255 being true red.

Remarks

You can use this in conjunction with the AmountBlue and AmountGreen properties.

Example

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set FIO = TK.FieldInfo("test", 1)
MsgBox FIO.AmountRed
Set FIO = Nothing
Set TK = Nothing
```

BackgroundColor

Description

Sets and retrieves the fill color of the specified field.

Return type

String

Description
The fill color for the specified field.

Syntax

`value = object.BackgroundColor = value`

The BackgroundColor property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	String	The fill color for the specified field.

BorderColor

Description

Sets and retrieves the border color for the specified field.

Return type

String

Description
The border color for the specified field.

Syntax

*value = object.***BorderColor** *= value*

The BorderColor property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	String	The border color for the specified field.

BorderWidth

Description

Sets and retrieves the border width for the specified field.

Return type

Short

Description
The border width for the specified field.

Syntax

value = *object*.**BorderWidth** = *value*

The BorderWidth property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Short	The border width for the specified field.

Bottom

Description

Sets and retrieves the lower Y coordinate for the specified field.

Return type

Float

Description
The lower Y coordinate. Uses the PDF Coordinate System.

Syntax

value = *object*.**Bottom** = *value*

The Bottom property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Float	The lower Y coordinate. Uses the PDF Coordinate System.

Remarks

You can use this in conjunction with the Height, Left, Top and Width properties.

ButtonTextAlternate

Description

Sets and retrieves the alternative text for current instance of a button field type.

Return type

String

Description
The alternative text.

Syntax

*value = object.***ButtonTextAlternate** *= value*

The ButtonTextAlternate property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	String	The alternative text.

ButtonTextNormal

Description

Sets and retrieves the text displayed on the current instance of a button field type.

Return type

String

Description
The text displayed.

Syntax

value = *object*.**ButtomTextNormal** = *value*

The ButtonTextNormal property has these required parts:

Part	**Value Type**	**Description**
Object		An expression of the PDFFieldInfo subobject.
Value	String	The text displayed.

ButtonTextRollover

Description

Sets and retrieves the text displayed when the mouse moves over the current instance of a button field type.

Return type

String

Description
The text displayed.

Syntax

*value = object.***ButtonTextRollover** *= value*

The ButtonTextRollover property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	String	The text displayed.

CheckboxStyle

Description

Sets or retrieves the check box style for the specified field.

Return type

Long

Return Value	**Description**
1	Check (default for checkbox).
2	Circle.
3	Cross.
4	Diamond.
5	Square.
6	Star.

Syntax

*value = object.***CheckboxStyle** *= value*

The CheckboxStyle property has this part:

Part	**Value Type**	**Description**
Object		An expression of the PDFFieldInfo subobject.
Value	Long	1 = check (default for checkbox) 2 = circle 3 = cross 4 = diamond 5 = square 6 = star

Color

Description

Sets and returns the text color of the specified field.

Return type

String

Description
The text color of the specified field.

Syntax

*value = object.***Color** *= value*

The Color property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	String	The text color of the specified field.

Comb

Description

Divides the field into equally spaced combs or sections based on the `MaxLen` entry of the field dictionary. Returns the current comb status of the field.

NOTE: If the `MaxLen` entry is not present, this property will not work.

Return type

Variant Bool

Return Value	Description
True	The property is turned on for the current form field instance.
False	The property is turned off for the current form field instance.

Syntax

value = *object*.**Comb** = *value*

The Comb property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	True = Turn the property on. False = Turn the property off. (Default)

CommitOnChange

Description

If set, the value entered into the form field will be saved immediately. If turned off, the value will be saved only when the user tabs to the next form field.

Return type

Variant Bool

Return Value	Description
True	The property is turned on for the current form field instance.
False	The property is turned off for the current form field instance.

Syntax

*value = object.***CommitOnChange** *= Value*

The CommitOnChange property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	True = Turn the property on. False = Turn the property off. (Default)

Description

Description

Sets and retrieves the short description set for the specified field.

Return type

String

Description
The short description.

Syntax

*value = object.***`Description`** *= value*

The Description property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	String	The short description.

DefaultValue

Description

Sets and retrieves the default value for the specified field instance.

Return type

String

Description
The default value for the specified field.

Syntax

*value = object.***`DefaultValue`** *= value*

The DefaultValue property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	String	The default value for the specified field.

DoNotScroll

Description

Enables or disables the scroll feature for the specified field.

Return type

Variant Bool

Return Value	Description
True	The property is turned on for the current form field instance.
False	The property is turned off for the current form field instance.

Syntax

*value = object.***DoNotScroll** *= value*

The DoNotScroll property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	True = Turn the property on. False = Turn the property off. (Default)

DoNotSpellCheck

Description

Enables or disables the spell check feature for the specified field.

Return type

Variant Bool

Return Value	Description
True	The property is turned on for the current form field instance.
False	The property is turned off for the current form field instance.

Syntax

*value = object.***DoNotSpellCheck** *= value*

The DoNotSpellCheck property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	True = Turn the property on. False = Turn the property off. (Default)

DisplayOffset

Description

Returns the distance the item is offset from the edge of the field based on alignment.

Return type

String

Description
The distance the first character is from the left side of the field in PDF Units.

Syntax

*value = object.***DisplayOffset**

The DisplayOffset property has this part:

Part	Description
Object	An expression of the PDFFieldInfo subobject.

Example

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set FIO = TK.FieldInfo("test", 1)
MsgBox FIO.DisplayOffset
Set FIO = Nothing
Set TK = Nothing
```

EditCombo

Description

Sets or removes the ability to edit a list for the current form field instance. If set, the list field can be used as a combo box.

Return type

Variant Bool

Return Value	Description
True	The property is turned on for the current form field instance.
False	The property is turned off for the current form field instance.

Syntax

*value = object.***EditCombo** *= value*

The EditCombo property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	True = Turn the property on. False = Turn the property off. (Default)

ExportValue

Description

Sets or retrieves the export value for the specified field.

Return type

String

Description
The export value for the specified field.

Syntax

*value = object.***ExportValue** *= value*

The ExportValue property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	String	The export value for the specified field.

FieldType

Description

Returns the type of field for the specified field.

Return type

String

Return Value	Description
/Btn	Push buttons, radio buttons and checkboxes.
/Tx	Single or multi-line text fields.
/Ch	List boxes and combo boxes.
/Sig	Digital signature field.

Syntax

*value = object.***FieldType**

The FieldType property has this part:

Part	Description
Object	An expression of the PDFFieldInfo subobject.

Example

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set FIO = TK.FieldInfo("test", 1)
MsgBox FIO.FieldType
Set FIO = Nothing
```

FileSelect

Description

If set, the specified field will display the path name to a file. The contents of the file represent the contents of the field.

Return type

Variant Bool

Return Value	Description
True	The property is turned on for the current form field instance.
False	The property is turned off for the current form field instance.

Syntax

*value = object.***FileSelect** *= value*

The FileSelect property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	True = Turn the property on. False = Turn the property off. (Default)

Flags

Description

Sets and retrieves the bitwise flags set for the specified field.

Return type

Long

Description
The flags set for the specified field.

Syntax

*value = object.***Flags** *= value*

The Flags property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Long	A series of flags that can be combined via "or" statements: -4096 = All bits will be cleared (set to 0). You can 'OR' 4096 with other bits to achieve the desired effect. (This affects the line on which it is called.) -998 = Flatten field and reset font, color and rotation information to field defaults. (You must use -998 on the line prior to the line you wish to reset.) -997 = Flatten field and do not reset font, color and rotation information. -996 = Flatten field using an image file as named in field data. The image type is auto-determined. -995 = Flatten field as a known JPEG using an image file as named in field data. -994 = Flatten field as a known TIFF using an image file as named in field data. 0 = Read Only. 1 = "As is". All attributes of the field remain unchanged. 2 = Hidden. 4 = Enable Printing. 8 = Disable Zoom. 16 = Disable Rotation.

		32 = The field will print, but cannot be viewed. 64 = The field will be hidden and read only.

Remarks

The flags can be “OR’ed” together.

Example

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Output.pdf")
Set FIO = TK.FieldInfo("test", 1)
MsgBox FIO.Flags
Set FIO = Nothing
Set TK = Nothing
```

FontName

Description

Sets and retrieves the font name used in the specified field.

Return type

String

Description
The font name used in the specified field.

Syntax

*value = object.***FontName** *= value*

The FontName property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	String	The font name used in the specified field.

Example

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set FIO = TK.FieldInfo("test", 1)
MsgBox FIO.FontName
Set FIO = Nothing
Set TK = Nothing
```

FontSize

Description

Sets and retrieves the font size used in the specified field.

Return type

Float

Description
The font size used in the specified field.

Syntax

*value = object.***FontSize** *= value*

The FontSize property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Float	The font size to use in the current field.

Example

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set FIO = TK.FieldInfo("test", 1)
MsgBox FIO.FontSize
Set FIO = Nothing
Set TK = Nothing
```

FormatString

Description

Sets and retrieves the format string used to display characters in the specified field.

Return type

String

Description
The format string used to display characters.

Syntax

*value = object.***FormatString** *= value*

The FormatString property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	String	The format string used to display characters.

Example

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set FIO = TK.FieldInfo("test", 1)
MsgBox FIO.FormatString
Set FIO = Nothing
Set TK = Nothing
```

Height

Description

Sets and retrieves the height of the current form field.

Float

Description
The height of the current form field instance in PDF Units.

Syntax

*value = object.***Height** *= value*

The Height property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Float	The height of the current form field instance in PDF Units.

Remarks

You can use this in conjunction with the Bottom, Left, Top and Width properties.

Example

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set FIO = TK.FieldInfo("test", 1)
MsgBox FIO.Height
Set FIO = Nothing
Set TK = Nothing
```

Hidden

Description

Sets or removes the hidden setting for the specified field. If hidden, the field will not display in the PDF or print.

Return type

Variant Bool

Return Value	Description
True	The property is turned on for the current form field instance.
False	The property is turned off for the current form field instance.

Syntax

*value = object.***Hidden** *= value*

The Hidden property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	True = Turn the property on. False = Turn the property off. (Default)

JavaScript

Description

Returns any custom calculation script set for the current form field instance.

Return type

String

Description
The custom calculation script for the specified field.

Syntax

*value = object.***JavaScript**

The JavaScript property has this part:

Part	Description
Object	An expression of the PDFFieldInfo subobject.

Example

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set FIO = TK.FieldInfo("test", 1)
MsgBox FIO.JavaScript
Set FIO = Nothing
Set TK = Nothing
```

KeyDownFormat

Description

Sets and retrieves the text displayed when the button is pressed.

Return type

String

Description
The text displayed.

Syntax

*value = object.***KeyDownFormat** *= value*

The KeyDownFormat property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	String	The text displayed.

Left

Description

Sets and retrieves the lower-left X coordinate for the current field.

Return type

Float

Description
The lower-left X coordinate. Uses the PDF Coordinate System.

Syntax

value = *object*.**Left** = *value*

The Left property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Float	The lower-left X coordinate. Uses the PDF Coordinate System.

Remarks

You can use this in conjunction with the Bottom, Height, Top and Width properties.

Example

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set FIO = TK.FieldInfo("test", 1)
MsgBox FIO.Left
Set FIO = Nothing
Set TK = Nothing
```

ListMultiSelect

Description

If set, the user will be able to select multiple items in a list box by pressing the **Ctrl** key during selection.

Return type

Variant Bool

Return Value	Description
True	The property is turned on for the current form field instance.
False	The property is turned off for the current form field instance.

Syntax

*value = object.***ListMultiSelect** *= value*

The ListMultiSelect property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	True = Turn the property on. False = Turn the property off. (Default)

ListSort

Description

Sets or determines if a list will be sorted alphabetically.

Return type

Variant Bool

Return Value	Description
True	The property is turned on for the current form field instance.
False	The property is turned off for the current form field instance.

Syntax

*value = object.***ListSort** *= value*

The ListSort property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	True = Turn the property on. False = Turn the property off. (Default)

Locked

Description

Adds or removes the locked status for the current form field instance. When locked, no changes can be made to the form field properties.

Return type

Variant Bool

Return Value	Description
True	The property is turned on for the current form field instance.
False	The property is turned off for the current form field instance.

Syntax

*value = object.***Locked** *= value*

The Locked property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	True = Turn the property on. False = Turn the property off. (Default)

MaxLength

Description

Sets and retrieves any specified allowed character limit for the specified field.

Return type

Short

Description
The character limit for the specified field.

Syntax

*value = object.***MaxLength** *= value*

The MaxLength property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Short	The character limit for the specified field.

Example

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set FIO = TK.FieldInfo("test", 1)
MsgBox FIO.MaxLength
Set FIO = Nothing
Set TK = Nothing
```

MouseDownScript

Description

Sets or retrieves the script to execute when the mouse moves down the field for the specified field.

Return type

String

Description
The script to execute when moving the mouse down the field for the specified field.

Syntax

*value = object.***MouseDownScript** *= value*

The MouseDownScript property has these required parts:

Part	**Value Type**	**Description**
Object		An expression of the PDFFieldInfo subobject.
Value	String	The script to execute when moving the mouse down the field for the specified field.

MouseEntryScript

Description

Sets or retrieves the script to execute when the mouse enters the field for the specified field.

Return type

String

Description
The script to execute when the mouse enters the field for the specified field.

Syntax

*value = object.***MouseEntryScript** *= value*

The MouseEntryScript property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	String	The script to execute when the mouse enters the field for the specified field.

MouseExitScript

Description

Sets or retrieves the script to execute when the mouse exits the field for the specified field.

Return type

String

Description
The script to execute when the mouse exits the field for the specified field.

Syntax

*value = object.***MouseExitScript** *= value*

The MouseExitScript property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	String	The script to execute when the mouse exits the field for the specified field.

MouseUpScript

Description

Sets or retrieves the script to execute when the mouse moves up the specified field.

Return type

String

Description
The script to execute when moving the mouse up the field for the specified field.

Syntax

*value = object.***`MouseUpScript`** *= value*

The MouseUpScript property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	String	The script to execute when moving the mouse up the field for the specified field.

Multiline

Description

Adds or removes the multi-line setting to the current form field instance. If set, the text in the field will wrap to a new line when it reaches the end of the field.

Return type

Variant Bool

Return Value	Description
True	The property is turned on for the current form field instance.
False	The property is turned off for the current form field instance.

Syntax

*value = object.***`Multiline`** *= value*

The Multiline property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	True = Turn the property on. False = Turn the property off. (Default)

Name

Description

Sets and retrieves the name of the specified field.

Return type

String

Description
The name of the specified field.

Syntax

*value = object.***Name** *= value*

The Name property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	String	The name of the specified field.

NoRotate

Description

Sets or removes the rotate setting for the specified field. If turned on, the field will not rotate when the page is rotated.

Return type

Variant Bool

Return Value	Description
True	The property is turned on for the current form field instance.
False	The property is turned off for the current form field instance.

Syntax

*value = object.***NoRotate** *= value*

The NoRotate property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	True = Turn the property on. False = Turn the property off. (Default)

NoToggleOnOff

Description

Used with radio buttons. Requires one radio button must be selected at all times. Deselecting the button will not clear the field.

Return type

Variant Bool

Return Value	Description
True	The property is turned on for the current form field instance.
False	The property is turned off for the current form field instance.

Syntax

*value = object.***NoToggleOnOff** *= value*

The NoToggleOnOff property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	True = Turn the property on. False = Turn the property off. (Default)

NoView

Description

Sets or removes the no view setting for the specified field. If turned on, the field will not display, but may print depending on the Printable setting.

Return type

Variant Bool

Return Value	Description
True	The property is turned on for the current form field instance.
False	The property is turned off for the current form field instance.

Syntax

*value = object.***NoView** *= value*

The NoView property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	True = Turn the property on. False = Turn the property off. (Default)

NoZoom

Description

Sets or removes the zoom setting for the specified field. If turned on, the field will not change size when using the PDF zoom feature.

Return type

Variant Bool

Return Value	**Description**
True	The property is turned on for the current form field instance.
False	The property is turned off for the current form field instance.

Syntax

*value = object.***NoZoom** *= value*

The NoZoom property has these required parts:

Part	**Value Type**	**Description**
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	True = Turn the property on. False = Turn the property off. (Default)

OnBlurScript

Description

Sets or retrieves the script to execute when the focus moves from the specified field to another location.

Return type

String

Description
The script to execute on blur.

Syntax

value = *object*.**OnBlurScript** = *value*

The OnBlurScript property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	String	The script to execute on blur.

OnFocusScript

Description

Sets or retrieves the script to execute when the focus moves from another location to the specified field.

Return type

String

Description
The script to execute on focus.

Syntax

*value = object.***OnFocusScript** *= value*

The OnFocusScript property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	String	The script to execute on focus.

PageNumber

Description

Returns the page number in which the current field is placed.

Return type

Long

Description
The page number on which the specified field is placed.

Syntax

*value = object.***PageNumber**

The PageNumber property has this part:

Part	Description
Object	An expression of the PDFFieldInfo subobject.

Example

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set FIO = TK.FieldInfo("test", 1)
MsgBox FIO.PageNumber
Set FIO = Nothing
Set TK = Nothing
```

Password

Description

Sets or removes the password type setting for the current form field instance. If set, the characters in the field will be masked.

Return type

Variant Bool

Return Value	Description
True	The property is turned on for the current form field instance.
False	The property is turned off for the current form field instance.

Syntax

*value = object.***Password** *= value*

The Password property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	True = Turn the property on. False = Turn the property off. (Default)

Printable

Description

Sets or removes the print setting for the specified field. When turned off, the field will not print.

Return type

Variant Bool

Return Value	Description
True	The property is turned on for the current form field instance.
False	The property is turned off for the current form field instance.

Syntax

`*value* = *object*.**Printable** = *value*`

The Printable property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	True = Turn the property on. (Default) False = Turn the property off.

RadiosInUnison

Description

If set, all radio buttons with the same on value will be checked when one radio button is selected. If turned off, the buttons will be mutually exclusive.

Return type

Variant Bool

Return Value	Description
True	The property is turned on for the current form field instance.
False	The property is turned off for the current form field instance.

Syntax

*value = object.***RadiosInUnison** *= value*

The RadiosInUnison property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	True = Turn the property on. False = Turn the property off. (Default)

ReadOnly

Description

Adds or removes the read only status for the current form field instance. If read only, the field cannot be selected by the PDF viewer.

Return type

Variant Bool

Return Value	Description
True	The property is turned on for the current form field instance.
False	The property is turned off for the current form field instance.

Syntax

value = *object*.**ReadOnly** = *value*

The ReadOnly property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	True = Turn the property on. False = Turn the property off. (Default)

Required

Description

Sets or removes the required setting for the current form field instance. If set, filling the form field is required.

Return type

Variant Bool

Return Value	Description
True	The property is turned on for the current form field instance.
False	The property is turned off for the current form field instance.

Syntax

*value = object.***Required** *= value*

The Required property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Variant_Bool	True = Turn the property on. False = Turn the property off. (Default)

Rotation

Description

Sets and retrieves the rotation of the field.

Return type

Short

Description
The amount of counterclockwise rotation in degrees.

Syntax

*value = object.***Rotation** *= value*

The Rotation property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Short	The amount of counterclockwise rotation in degrees.

Example

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set FIO = TK.FieldInfo("test", 1)
MsgBox FIO.Rotation
Set FIO = Nothing
Set TK = Nothing
```

Top

Description

Sets and retrieves the top Y coordinate of the specified field.

Return type

Float

Description
The top Y coordinate of the field. Uses the PDF Coordinate System.

Syntax

*value = object.***Top** *= value*

The Top property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Float	The top Y coordinate of the field. Uses the PDF Coordinate System.

Remarks

You can use this in conjunction with the Bottom, Height, Left and Width properties.

Example

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set FIO = TK.FieldInfo("test", 1)
MsgBox FIO.Top
Set FIO = Nothing
Set TK = Nothing
```

Type

Description

Returns the field type of the specified field.

Return type

Long

Return Value	**Description**
1	Text box.
2	Signature.
3	Push button.
4	Checkbox.
5	Combo box.
6	List box.

Syntax

*value = object.***Type**

The Type property has this part:

Part	**Description**
Object	An expression of the PDFFieldInfo subobject.

ValidationScript

Description

Sets and retrieves the validation script for the specified field.

Return type

String

Description
The validation script for the specified field.

Syntax

*value = object.***`ValidationScript`** *= value*

The ValidationScript property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	String	The validation script for the specified field.

Value

Description

Sets and retrieves the value for the specified field.

Return type

String

Description
The value for the specified field.

Syntax

*value = object.***Value** *= value*

The Value property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	String	The value for the specified field.

Width

Description

Sets and retrieves the width of the current form field.

Return type

Float

Description
The width of the current form field instance in PDF Units.

Syntax

*value = object.***Width** *= value*

The Width property has these required parts:

Part	Value Type	Description
Object		An expression of the PDFFieldInfo subobject.
Value	Float	The width of the current form field instance in PDF Units.

Remarks

You can use this in conjunction with the Bottom, Height, Left and Top properties.

Example

```
strPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".") & "\"
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set FIO = TK.FieldInfo("test", 1)
MsgBox FIO.Width
Set FIO = Nothing
Set TK = Nothing
```

ListItems

ListItems is a subobject of the PDFFieldInfo object, which sets and retrieves a predefined set for a specified form field list item.

This section includes the following:

- Instantiating the ListItems subobject.
- Properties.

Instantiating the ListItems subobject

The ListItems subobject is created by passing a field name and field instance to the Toolkit object FieldInfo method, and then passing an instance to the PDFFieldInfo object ListItems method.

```
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set FIO = TK.FieldInfo("test", 1)
Set LST = TK.ListItems
```

Properties

The ListItems subobject contains the following properties:

- DisplayItem.
- ExportValue.

DisplayItem

Description

The DisplayItem property will set and retrieve the list items contained in a list field.

Return type

String

Description
The items contained in the list box

Syntax

*value = object.***DisplayItem** = *value*

The DisplayItem property has these required parts:

Part	Value Type	Description
Object		An expression of the ListItems subobject.
Value	String	The items to place in the list box.

ExportValue

Description

Set and retrieve the export value for the list field.

Return type

String

Description
The current export value for the list field.

Syntax

*value = object.***ExportValue** *= value*

The ExportValue property has these required parts:

Part	Value Type	Description
Object		An expression of the ListItems subobject.
Value	String	The export value for the list field.

Text2PDF

Text2PDF is a subobject of the Toolkit object, which converts a specified text file or text stream to PDF. A single Text2PDF subobject is associated with each control instance, which enables performance of simple Text conversions straight to PDF.

This section includes the following:

- Instantiating the Text2PDF Subobject.
- Text2PDF Extra Parameters.
- Method.
- Properties.

Instantiating the Text2PDF Subobject

To instantiate the Text2PDF subobject, you must instantiate the Toolkit object first.

```
Set Toolkit = CreateObject("APToolkit.Object")
Set Text2PDF = Toolkit.Text2PDFObject
```

Text2PDF Extra Parameters

The Text2PDF subobject uses additional parameters and instructions for defining additional items or inserting a page break. For additional information, refer to the following sections:

- Input Stream Extras.
- Inserting a Page Break.

Input Stream Extras

You can define additional items such as annotations in the Input Stream. These are defined on a line by themselves.

Item	Description
.action o file.pdf	This will generate a link to open a specified PDF file.
.action program.exe	This will generate a launch link to a specified program (.exe).
.stream rawpdfstream	This will insert a specified PDF stream into the file.
.line+n	This will add n lines to the line counter. You can use this when .stream changes the internal line counter.
.line=n	This sets the internal line counter to n. You can use this when .stream changes the internal line counter.
.line-n	This will subtract n lines from the line counter. You can use this when

	.stream changes the internal line.
.bs n	This line changes the current box style to n, where n is defined by the AnnotBoxStyle property.

Inserting a Page Break

You can Insert a char(12) (form feed) into the stream after a carriage return/LF combination.

Method

The Text2PDF subobject contains the Convert method.

Convert

Description

Generates the output file using either the input file or input stream.

Return type

Long

Return Value	Description
0	Success.
-1	No output file specified.
-2	No input file or stream specified.
-3	Not enough memory to allocate.
-4	Unable to open input file.
-5	Unable to generate output file.

Syntax

*`Value = object.`***`Convert`**

The Convert method has this part:

Part	Description
Object	An expression of the Text2PDF subobject.

Properties

The Text2PDF subobject contains the following properties:

AnnotBoxStyle

Description

Sets the style for the border of the annotation box generated when converting text to PDF.

Syntax

object.**AnnotBoxStyle** = *value*

The AnnotBoxStyle property has these parts:

Part	Value Type	Description
Object		An expression of the Text2PDF subobject.
Value	Short	The type of box. -1 = Invisible 0 = Black solid 1 = Red dashed 2 = Red solid 3 = Green dashed 4 = Green solid 5 = Blue dashed 6 = Blue solid

Border

Description

Sets the border width around the edges of the PDF.

Syntax

```
object.Border = value
```

The Border property has these parts:

Part	Value Type	Description
Object		An expression of the Text2PDF subobject.
Value	Short	The border printed around the edges of the PDF in PDF Units. (Default is 20.)

CompressContents

Description

Compresses the text contents of the resultant PDF.

Syntax

object.**CompressContents** = *value*

The CompressContents property has these parts:

Part	Value Type	Description
Object		An expression of the Text2PDF subobject.
Value	Long	1 = Compress text contents. (Default) 0 = Do not compress text contents.

DOSTranslation

Description

Converts certain DOS characters to characters supported by the PDF font set.

Syntax

object.**DOSTranslation** = *value*

The DOSTranslation property has these parts:

Part	Value Type	Description
Object		An expression of the Text2PDF subobject.
Value	Long	1 = DOS characters will be converted. 0 = DOS characters will not be converted. (Default)

FixedLength

Description

Sets the maximum length for the inserted text.

Return type

None

Syntax

object.**FixedLength** = *value*

The FixedLength property has these required parts:

Part	Value Type	Description
Object		An expression of the Text2PDF subobject.
Value	Short	The length specified in PDF Units.

Remarks

You can use this in conjunction with the WordWrap property to control how the text breaks.

FontName

Description

Sets the font name for the text.

Syntax

object.**FontName** = *value*

The FontName property has these parts:

Part	Value Type	Description
Object		An expression of the Text2PDF subobject.
Value	String	The specified font name must be one of the one of the base 14 fonts or match the font name exactly as it appears in the X:\Windows\fonts directory.

FontSize

Description

Sets the font size of the output.

Syntax

```
object.FontSize = value
```

The FontSize property has these parts:

Part	Value Type	Description
Object		An expression of the Text2PDF subobject.
Value	Double	The font size for the output. (Default is 12 pt.)

InfoString

Description

Sets the value of Title data.

Return type

None

Syntax

object.**InfoString** = *value*

The InfoString property has these required parts:

Part	Value Type	Description
Object		An expression of the Text2PDF subobject.
Value	String	The text for the Title value.

InputFile

Description

Sets the name of the input file to open.

NOTE: This will overwrite and erase the value of the InputStream property.

Syntax

object.**InputFile** = *value*

The InputFile property has these parts:

Part	Value Type	Description
Object		An expression of the Text2PDF subobject.
Value	String	The name of the input file to open.

InputStream

Description

Sets the text stream or string to be processed.

NOTE: InputStream overwrites the value specified by the OpenInputFile method.

Syntax

object.**InputStream** = *value*

The InputStream property has these parts:

Part	**Value Type**	**Description**
Object		An expression of the Text2PDF subobject.
Value	String	The text stream or string to be processed.

LineSpacing

Description

Sets the space to insert between each line of text.

Syntax

*object.***LineSpacing** = *value*

The LineSpacing property has these parts:

Part	Value Type	Description
Object		An expression of the Text2PDF subobject.
Value	Double	The space to insert between each line based on the specified font size. (Defaults to 1 line.)

OutputFile

Description

Sets the name of the output file.

Syntax

```
object.OutputFile = value
```

The OutputFile property has these parts:

Part	Value Type	Description
Object		An expression of the Text2PDF subobject.
Value	String	The name of the output file.

PageHeight

Description

Sets the page height of the PDF.

Syntax

```
object.PageHeight = value
```

The PageHeight property has these parts:

Part	Value Type	Description
Object		An expression of the Text2PDF subobject.
Value	Long	The height of the PDF, specified in PDF Units. (Default is 792)

PageWidth

Description

Sets the page width of the PDF.

Syntax

```
object.PageWidth = value
```

The PageWidth property has these parts:

Part	Value Type	Description
Object		An expression of the Text2PDF subobject.
Value	Long	The width of the PDF, specified in PDF Units. (Default is 612.)

PDFEncoding

Description

Sets PDF encoding for the character set.

Syntax

object.**PDFEncoding** = *value*

The PDFEncoding property has these parts:

Part	Value Type	Description
Object		An expression of the Text2PDF subobject.
Value	Long	1 = PDF encoding will be used. 0 = WinAnsiEncoding will be used. (Default)

SetRGB

Description

Sets an RGB color scheme for the text.

Syntax

object.**SetRGB** = *value*

The SetRGB property has these parts:

Part	**Value Type**	**Description**
Object		An expression of the Text2PDF subobject.
Value	Long	1 = RGB color scheme will be used. 0 = RGB color scheme will not be used. (Default.)

SpaceWidth

Description

Sets the width of spaces between words.

Syntax

object.**SpaceWidth** = *value*

The SpaceWidth property has these parts:

Part	Value Type	Description
Object		An expression of the Text2PDF subobject.
Value	Double	The width to use for a space, specified in PDF Units. (Default is 0.6)

TabNumSpaces

Description

Specifies the number of spaces used to replace a tab character once encountered.

NOTE: The width of a space is set using SpaceWidth.

Syntax

object.**TabNumSpaces** = *value*

The TabNumSpaces property has these parts:

Part	Value Type	Description
Object		An expression of the Text2PDF subobject.
Value	Short	The number of spaces to use when a tab is encountered. (Default is 3.)

TitleString

Description

Sets a one-line title to be displayed at the top of every page.

Syntax

*object.***TitleString** = *value*

The TitleString property has these parts:

Part	Value Type	Description
Object		An expression of the Text2PDF subobject.
Value	String	The text for the title.

Example

```
Set TK = CreateObject("APToolkit.Object")
Set Text2PDF = TK.Text2PDFObject
Text2PDF.Outputfile "output.pdf"
Text2PDF.InputFile "input.txt"
Text2PDF. TitleString = "This is my line title on each page"
Text2PDF.Convert
Set Text2PDF = Nothing
Set TK = Nothing
```

WordWrap

Description

Turns word wrapping on or off.

Syntax

object.**WordWrap** = *value*

The WordWrap property has these parts:

Part	Value Type	Description
Object		An expression of the Text2PDF subobject.
Value	Long	1 = turns word wrapping on. (Default) 0 = turns word wrapping off.

Example

```
Set TK = CreateObject("APToolkit.Object")
Set Text2PDF = TK.Text2PDFObject
Text2PDF.Outputfile "output.pdf"
Text2PDF.InputFile "input.txt"
Text2PDF.WordWrap = True
Text2PDF.Convert
Set Text2PDF = Nothing
Set TK = Nothing
```

Flash

The Flash object enables you to embed Flash files in your PDF. Using the Flash object, you can set and control various parameters related to the display and viewing of your flash file.

This section includes:

- Using the Flash Object.
- Method.
- Properties.

Using the Flash Object

The Flash object is not instantiated using the procedure common to the other Toolkit objects. You create the Flash object using a programmatic identifier or ProgID. The ProgID for the Barcode object is `APToolkit.Flash`.

To create the Flash object, use the following syntax:

```
   Object = CreateObject("APToolkit.Flash")
```

Method

The Flash object contains the AsString method.

AsString

Description

Returns an XML stream of the Flash file for use with the SetFormFieldData method.

Return type

String

Description
The XML stream of the flash file.

Syntax

```
value = object.AsString
```

The AsString property has this part:

Part	Description
Object	An expression of the Flash object.

Properties

The Flash object has the following properties:

Description

Description

This property sets the description that appears when moving the mouse over the flash file in the PDF.

Return type

None

Syntax

object.**Description** = *value*

The Description property has these required parts:

Part	Value Type	Description
Object		An expression of the Flash object.
Value	String	The description.

Filename

Description

Specifies the name and location of the Flash file to embed in your PDF document.

Return type

None

Syntax

object.**Filename** = *value*

The Filename property has these required parts:

Part	Value Type	Description
Object		An expression of the Flash object.
Value	String	The full path to the flash file on your local system.

Flags

Description

Sets flags that control the display of the flash file in the PDF document.

Return type

None

Syntax

object.**Flags** = *value*

The Flags property has these required parts:

Part	Value Type	Description
Object		An expression of the Flash object.
Value	Long	A series of flags that can be combined via "or" statements: 0 = Read Only. 1 = "As is". All attributes of the field remain unchanged. 2 = Hidden. 4 = Enable Printing. 8 = Disable Zoom. 16 = Disable Rotation. 32 = The movie will print, but cannot be viewed. 64 = The movie will be hidden and read only.

Remarks

The flags can be “OR’ed” together.

Loop

Description

Sets the specified number of times the flash file will repeat once the movie has finished.

Return type

None

Syntax

object.**Loop** = *value*

The Loop property has these required parts:

Part	**Value Type**	**Description**
Object		An expression of the Flash object.
Value	Float	0 to n where 0 = continuous. (Default is 0.)

PlayCommand

Description

Sets actions to control when the flash file will begin playing.

Return type

None

Syntax

object.**PlayCommand** = *value*

The PlayCommand property has these required parts:

Part	Value Type	Description
Object		An expression of the Flash object.
Value	String	PO - Page open (default). PC - Page closed PV - Page visible PI - Page invisible D - Mouse down U - Mouse up

Rendition

Description

Specifies the name given to the flash file to be passed to the Flash object. This can be useful for JavaScript purposes.

NOTE: You must specify a rendition name.

Return type

None

Syntax

object.**Rendition** = *value*

The Rendition property has these required parts:

Part	Value Type	Description
Object		An expression of the Flash object.
Value	String	The rendition name.

Explorer

The Explorer object is a subobject of the Toolkit object. You can use the Explorer object to gain access to the underlying structure of your PDFs including reference objects, values, attributes, and entries.

You can use the Explorer object to locate problems in the PDF structure such as incorrect dictionary references, missing values and broken objects.

This section includes:

- Instantiating the Explorer Object.
- Methods.

NOTE: The Explorer object requires a strong understanding of PDF code. For additional reference, refer to the PDF Specification.

Instantiating the Explorer Object

The Explorer subobject is created by passing instance to the Toolkit object Explorer property.

```
Set TK = CreateObject("APToolkit.Object")
r = TK.OpenInputFile("Input.pdf")
Set PDFExplorer = TK.Explorer
```

Methods

The Explorer object has the following methods:

CountObjects

Description

Returns the number of objects in the current input PDF.

Return type

Long

Description
The number of objects in the current input PDF.

Syntax

```
value= object.CountObjects
```

The CountObjects property has this part:

Part	Description
Object	An expression of the Explorer subobject.

Dict_GetAttrVal

Description

Returns the attribute value of the specified dictionary entry

Return type

Long

Description
The attribute value.

Syntax

*value = object.***Dict_GetAttrVal** *t_dict, t_name*

The Dict_GetAttrVal method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
t_dict	Long	The specified dictionary entry.
t_name	String	The name of the value.

Dict_GetName

Description

Returns the name for the specified dictionary entry.

Return type

String

Description
The name of the dictionary entry.

Syntax

*value = object.***Dict_GetName** *t_dict, tPos*

The Dict_GetName method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
t_dict	Long	The specified dictionary entry.
tPos	Long	A zero based index.

Dict_GetValue

Description

Returns the value for the specified dictionary entry.

Return type

Long

Description
The value for the specified dictionary entry.

Syntax

*value = object.***Dict_GetValue** *t_dict, tPos*

The Dict_GetValue method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
t_dict	Long	The specified dictionary object.
tPos	Long	A zero based index.

GetObject

Description

Retrieves the object specified by the object ID.

Return type

Long

Description
The object specified by the object ID

Syntax

*value = object.***GetObject** *objectID*

The GetObject method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
t_val	Long	The object ID.

GetPageNbrForObject

Description

Returns the page number for the specified object ID.

Return type

Long

Description
The page number where the specified object ID is located.

Syntax

*value = object.***GetPageNbrForObjects** *objectID*

The GetPageNbrForObject method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
t_val	Long	The object ID.

GetRootObject

Description

Returns the root object and references.

Return type

Long

Description
The root object and references.

Syntax

*value = object.***GetRootObject**

The GetRootObject method has this part:

Part	Description
Object	An expression of the Explorer subobject.

Obj_AttrVal

Description

Returns the attribute value for the specified object.

Return type

Long

Description
The attribute value.

Syntax

*value = object.***Obj_AttrVal** *t_object*

The Obj_AttrVal method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
t_object	Long	The specified object.

Val_DecodePDFString

Description

Decodes the PDF string of the specified value.

Return type

String

Description
The decoded PDF string.

Syntax

`value = object.`**`Val_DecodePDFString`**` t_val`

The Val_DecodePDFString method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
t_val	Long	The specified value.

Val_GetDict

Description

Returns the dictionary entry for the specified value.

Return type

Long

Description
The dictionary entry for the specified value.

Syntax

*value = object.***Val_GetDict** *t_val*

The Val_GetDict method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
t_val	Long	The specified value.

Val_GetFirstElem

Description

Returns the first element of the specified value.

Return type

Long

Description
The first element of the specified value.

Syntax

*value = object.***Val_GetFirstElem** *t_val*

The Val_GetFirstElem method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
t_val	Long	The specified value.

Val_GetFloatVal

Description

Returns a float value of the specified value.

Return type

Float

Description
The float value of the specified object value.

Syntax

*value = object.***`Val_GetFloatVal`** *t_val*

The Val_GetFloatVal method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
t_val	Long	The specified value.

Obj_GetKind

Description

Returns the type for the specified value.

Return type

Long

Description
The object type for the specified value.

Syntax

*value = object.***`Obj_GetKind`** *t_val*

The Obj_GetKind method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
t_val	Long	The specified value.

Val_GetNextElem

Description

Returns the next element for the specified value.

Return type

Long

Description
The next element for the specified value.

Syntax

*value = object.***Val_GetNextElem** *t_val*

The Val_GetNextElem method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
t_val	Long	The specified value.

Val_GetNumVal

Description

Returns the numerical value of the specified value.

Return type

Long

Description
The numerical value for the specified object value.

Syntax

*value = object.***Val_GetNumVal** *t_val*

The Val_GetNumVal method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
t_val	Long	The specified value.

Val_GetStreamChars

Description

Returns the character stream for the specified value.

Return type

String

Description
The character stream for the specified value.

Syntax

*value = object.***Val_GetStreamChars** *t_val*

The Val_GetStreamChars method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
t_val	Long	The specified value.

Val_GetStreamLength

Description

Returns the length of the stream for the specified value.

Return type

Long

Description
The length of the stream for the specified value.

Syntax

*value = object.***Val_GetStreamLength** *t_val*

The Val_GetStreamLength method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
t_val	Long	The specified value.

Val_GetString

Description

Returns the specified value as an XML string.

Return type

String

Description
The specified value as an XML string.

Syntax

*value = object.***`Val_GetString`** *m_val*

The Val_GetString method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
m_val	Long	The specified value.

Val_HasStream

Description

Determines if the specified value has a stream.

Return type

Long

Return Value	Description
1	The value has a stream,
0	The value does not have a stream.

Syntax

*value = object.***Val_HasStream** *t_val*

The Val_HasStream method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
t_val	Long	The specified value.

Val_IsArray

Description

Determines if the specified value is an array.

Return type

Long

Return Value	Description
1	The value is an array.
0	The value is not an array.

Syntax

*value = object.***`Val_IsArray`** *t_val*

The Val_IsArray method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
t_val	Long	The specified value.

Val_IsDict

Description

Determines if the specified value is a dictionary entry.

Return type

Long

Return Value	Description
1	The value is a dictionary entry.
0	The value is not a dictionary entry.

Syntax

*value = object.***Val_IsDict** *t_val*

The Val_IsDict method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
t_val	Long	The specified value.

Val_IsNum

Description

Determines if the specified value is a number.

Return type

Long

Return Value	Description
1	The value is a number,
0	The value is not a number.

Syntax

`value = object.`**`Val_IsNum`**` t_val`

The Val_IsNum method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
t_val	Long	The specified value.

Val_IsRef

Description

Determines if the specified value is a reference.

Return type

Long

Return Value	Description
1	The value is a reference entry.
0	The value is not a reference entry.

Syntax

*value = object.***Val_IsRef** *t_val*

The Val_IsRef method has these required parts:

Part	Value Type	Description
Object		An expression of the Explorer subobject.
t_val	Long	The specified value.

Val_IsString

Description

Determines if the specified value is a string.

Return type

Long

Return Value	**Description**
1	The value is a string,
0	The value is not a string.

Syntax

*`value = object.`***`Val_IsString`** *`t_val`*

The Val_IsString method has these required parts:

Part	**Value Type**	**Description**
Object		An expression of the Explorer subobject.
t_val	Long	The specified value.

Barcode

The Barcode object generates barcodes for embedding in your PDF document. Using the Barcode object, you can encode information into symbologies, accessible by your barcode readers.

This section covers the following:

- Using the Barcode Object.
- Methods.
- Properties.

For information regarding each symbology, refer to Appendix C: Symbologies.

Using the Barcode Object

The Barcode object is not instantiated using the procedure common to the other Toolkit objects. You create the Barcode object using a programmatic identifier or ProgID. The ProgID for the Barcode object is APToolkit.Barcode.

To create the Barcode object, use the following syntax:

```
Object = CreateObject("APToolkit.Barcode")
```

Example script

```
Set TK = CreateObject("APToolkit.Object")
Set barcode = CreateObject("APToolkit.Barcode")
```

Methods

The barcode object contains the following methods:

AsString

Description

Retrieve an XML stream of the generated barcode for use with the SetFormFieldData method.

Return type

String

Description
An XML stream of the generated barcode.

Syntax

value = object.**AsString**

The AsString method has this part:

Part	Description
Object	An expression of the Barcode object.

Remarks

Depending on the specified design mode, the barcode may not use the entire field space. We recommend that you generate your fields as close to the finished size of the barcode as possible.

CommentFont

Description

Sets the font and font style used for your comment text specified in the Comment property.

NOTE: CommentFont must be called prior to the Comment property.

Return type

None

Syntax

`Object``.`**`CommentFont`** *`FontName,fontSize,tBold,tItalic,tUnderline,tStrikethru, tCharset`*

The CommentFont method has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
FontName	String	The name of the font.
fontSize	Short	The font size.
tBold	Variant_Bool	True = The comment text will be bolded. False = The comment text weight will be normal. (Default)
tUnderline	Variant_Bool	True = The comment text will be underlined. False = The comment text will not be underlined.(Default)
tStrikethru	Variant_Bool	True = The comment text will be struck through. False = The comment text will not be struck through.(Default)
tCharset	Short	Default = 1

Remarks

By default, Toolkit generates the comment text in accordance with the barcode standards. If you are using the barcode for commercial purposes, we recommend that you do not change the comment font.

Font

Description

Set the font and font style for the human readable text.

Return type

None

Syntax

*object.***Font** *FontName, fontSize, tBold, tItalic, tUnderline, tStrikethru, tCharset*

The Font method has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
FontName	String	Name of the font installed on the local system. The font must be installed.
fontSize	Short	The size of the font.
tBold	Variant_Bool	True = font is bold. False = font is not bold. (Default)
tItalic	Variant_Bool	True = font is italic. False = font is not italic. (Default)
tUnderline	Variant_Bool	True = font is underlined. False = font is not underlined. (Default)
tStrikethru	Variant_Bool	True = The font will be struck through. False = The font will not be struck through.(Default)
tCharset	Short	Default = 0

Remarks

By default, Toolkit generates the human readable text in accordance with the barcode standards. If you are using the barcode for commercial purposes, we recommend that you do not change the font.

Properties

The Barcode object has the following properties:

AutoLabelSize

Description

Sets the design mode to Barcode Design Mode (BDM) or Label Design Mode (LDM).

Syntax

Object.**AutoLabelSize** = *value*

The AutoLabelSize property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.

Value	Variant_Bool	False = Label design mode (Default).
		True = Barcode design mode.

Remarks

In BDM, the barcode will auto size to the field where you instruct Toolkit to insert the barcode. Toolkit retains the barcode proportions.

In LDM, you must set the height and width of the label area.

BackColor

Description

Sets the background color of your barcode, used in conjunction with ForeColor.

Syntax

Object.**BackColor** = *value*

The BackColor property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	String	Use hex values. Default is white (0xFFFFFF).

Remarks

This changes the color of the label area. If your barcode reader does not support the use of alternative color, you should leave this property using default settings.

BarHeight

Description

Sets the bar height in the generated barcode.

Syntax

Object.**BarHeight** = *value*

The BarHeight property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	A number. Default is 1000. Units are specified with the Measurement property.

Remarks

The BarHeight property specifies the bar height for all linear symbologies except those whose height cannot be adjusted (such as POSTNET and RoyalMail). You can refer to Appendix C: Symbologies for a description of each barcode symbology.

BarWidthReduction

Description

Sets bar width reduction.

Syntax

Object.**BarWidthReduction** = *value*

The BarWidthReduction property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	99 to -99. Default is 0. Positive number indicates a width REDUCTION. Negative number indicates an INCREASE in width.

Remarks

Bar width can compensate for some printers, allowing for the ink spread or shrinkage.

BearerBars

Description

Sets bearer bars around the barcode.

Syntax

Object.**BearerBars** = *value*

The BearerBars property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.

Value	Variant_Bool	False = No bearer bars (Default).
		True = Include bearer bars.

Remarks

Bearer bars are horizontal bars printed across the top and bottom of some bar codes to help avoid partial reads if scanner moves off the top or bottom of the code. These are usually only required for certain types of barcodes, since the start and stop characters of most bar codes make bearer bars unnecessary.

The following symbologies support bearer bars

- Code39.
- Code93.
- Interleaved 2 of 5.
- Codabar.
- Code11.
- Code128.

This property does not affect barcode images for other symbologies.

You can refer to Appendix C: Symbologies for a description of each barcode symbology.

BorderColor

Description

Sets the color of the border.

Syntax

Object.**BorderColor** = *value*

The BorderColor property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	String	An RGB color value.

Remarks

If your barcode scanner does not support color, you should not change the color of the label border.

BorderStyle

Description

Sets the style of the border.

Syntax

Object.BorderStyle = value

The BorderStyle property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	0 = No border. 1 = Solid line. 2 = Dashed line. 3 = Dotted line. 4 = Dash-dot line. 5 = Dash-dot-dot line.

Remarks

The size of the border width affects the display of the selected border style.

BorderWidth

Description

Sets the value of the border width.

Syntax

Object.**BorderWidth** = *value*

The BorderWidth property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	0 = no border (Default). Units are specified with the Measurement property.

Remarks

The size of the border width affects the display of the selected border style.

Code25OptionalCheckDigit

Description

Determines if a checkdigit will be appended to your generated Code25 barcode.

Syntax

Object.**Code25OptionalCheckDigit** = *value*

The Code25OptionalCheckDigit property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Variant_Bool	False = No check digit is included. (Default) True = A check digit is calculated using the modulo 10 method and appended to the encoded data.

Remarks

For details on using the checkdigit, you can refer to Appendix C: Symbologies for a description of each barcode symbology.

Code39OptionalCheckDigit

Description

Determines if a checkdigit will be appended to your generated Code39 barcode.

Syntax

Object.**Code39OptionalCheckDigit** = *value*

The Code39OptionalCheckDigit property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Variant_Bool	False = No check digit is included. (Default) True = A check digit is calculated using the modulo 43 method and appended to the encoded data.

Remarks

For details on using the checkdigit, you can refer to Appendix C: Symbologies for a description of each barcode symbology.

Code39StartStopChars

Description

Determines if the start and stop characters will appear in the human readable text for the Code39 barcode.

Syntax

Object.**Code39StartStopChars** = *value*

The Code39StartStopChars property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Variant_Bool	False = No check digit is included. (Default) True = A check digit is calculated using the modulo 10 method and appended to the encoded data.

Remarks

This property affects all three Code 39 symbologies – Code 39, Code39 Mod 43 and Code39 Full ASCII. You can refer to Appendix C: Symbologies for a description of each barcode symbology.

Comment

Description

Sets the text to display as the barcode comment.

Return type

None

Syntax

```
Object.Comment = value
```

The Comment property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	String	Comment text to be printed.

Remarks

You can extend a longer comment to two lines by setting the comment margins. For additional information, refer to Appendix C: Symbologies for a description of each barcode symbology.

CommentAlignment

Description

Sets the text alignment for the comment.

Return type

None

Syntax

`Object``.`**`CommentAlignment`**` = `*`value`*

The CommentAlignment property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	0 = Left alignment (default). Align the text with left edge of the comment box. 1 = Right alignment. Align the text with the right edge of the comment box. 2 = Center alignment. Align the text with the center of the comment box. 3 = Justify alignment. Align the text to both edges of the comment box.

Remarks

The comment margins are calculated different depending on the alignment and current design mode. In BDM, Toolkit measures the horizontal margins from the edge of the comment to the edge of the generated barcode.

In LDM, Toolkit measures the horizontal margins from the edge of the comment to the label boundary. Vertical comment margins are not affected.

For additional details, refer to the CommentMarginBottom, CommentMarginLeft, CommentMarginRight, CommentMarginTop properties.

CommentMarginBottom

Description

Sets the width for the bottom comment margin.

Return type

None

Syntax

Object.**CommentMarginBottom** = *value*

The CommentMarginBottom property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	A positive number. The unit of measurement determined by the Measurement property.

Remarks

The comment margins are calculated different depending on the alignment and current design mode. In BDM, Toolkit measures the horizontal margins from the edge of the comment to the edge of the generated barcode.

In LDM, Toolkit measures the horizontal margins from the edge of the comment to the label boundary. Vertical comment margins are not affected.

For additional details, refer to the CommentAlignment, CommentMarginLeft, CommentMarginRight, CommentMarginTop properties.

CommentMarginLeft

Description

Sets the width for the left comment margin.

Return type

None

Syntax

Object.**CommentMarginLeft** = *value*

The CommentMarginLeft property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	A positive number. The unit of measurement determined by the Measurement property.

Remarks

The comment margins are calculated different depending on the alignment and current design mode. In BDM, Toolkit measures the horizontal margins from the edge of the comment to the edge of the generated barcode.

In LDM, Toolkit measures the horizontal margins from the edge of the comment to the label boundary. Vertical comment margins are not affected.

For additional details, refer to the CommentAlignment, CommentMarginBottom, CommentMarginRight, CommentMarginTop properties.

CommentMarginRight

Description

Sets the width for the right comment margin.

Return type

None

Syntax

Object.**CommentMarginRight** = *value*

The CommentMarginRight property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	A positive number. The unit of measurement determined by the Measurement property.

Remarks

The comment margins are calculated different depending on the alignment and current design mode. In BDM, Toolkit measures the horizontal margins from the edge of the comment to the edge of the generated barcode.

In LDM, Toolkit measures the horizontal margins from the edge of the comment to the label boundary. Vertical comment margins are not affected.

For additional details, refer to the CommentAlignment, CommentMarginBottom, CommentMarginLeft, CommentMarginTop properties.

CommentMarginTop

Description

Sets the width for the top comment margin.

Return type

None

Syntax

Object.**CommentMarginTop** = *value*

The CommentMarginTop property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	A positive number. The unit of measurement determined by the Measurement property.

Remarks

The comment margins are calculated different depending on the alignment and current design mode. In BDM, Toolkit measures the horizontal margins from the edge of the comment to the edge of the generated barcode.

In LDM, Toolkit measures the horizontal margins from the edge of the comment to the label boundary. Vertical comment margins are not affected.

For additional details, refer to the CommentAlignment, CommentMarginBottom, CommentMarginLeft, CommentMarginRight properties.

CommentOnTop

Description

Instructs Toolkit to display the comment text above the barcode symbol when generated.

Return type

None

Syntax

`object``.`**`CommentOnTop`**` = `*`value`*

The CommentOnTop property has these required parts:

Part	**Value Type**	**Description**
Object		An expression of the Barcode object.
Value	Variant_Bool	True = Comment is placed above the barcode symbol. False = Comment is placed below the barcode symbol.

Remarks

You can refer to Appendix C: Symbologies for a description of each barcode symbology.

DataMatrixModuleSize

Description

Determines the width and height of a single cell in the DataMatrix symbols generated by the Toolkit barcode object.

Return type

None

Syntax

object.**DataMatrixModuleSize** = *value*

The DataMatrixModuleSize property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	A number between 1 and 100. Default is 20.

Remarks

The DataMatrix symbol is comprised of squares. This property sets the size of the square. The value affects the overall symbol size. You can refer to Appendix C: Symbologies for a description of the DataMatrix barcode symbology.

DataMatrixTargetSizeID

Description

Determines the shape of the DataMatrix symbol generated by the Toolkit barcode object.

Return type

None

Syntax

object.**DataMatrixTargetSizeID** = *value*

The DataMatrixTargetSizeID property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	0= Auto select the number of rows and cols. 1 = Rectangle symbol of 12 by 12 modules. 2 = Rectangle symbol of 14 by 14 modules. 3 = Rectangle symbol of 16 by 16 modules. 4 = Rectangle symbol of 18 by 18 modules. 5 = Rectangle symbol of 20 by 20 modules. 6 = Rectangle symbol of 22 by 22 modules. 7 = Rectangle symbol of 24 by 24 modules. 8 = Rectangle symbol of 26 by 26 modules. 9 = Rectangle symbol of 32 by 32 modules. 10 = Rectangle symbol of 36 by 36 modules. 11 = Rectangle symbol of 40 by 40 modules. 12 = Rectangle symbol of 44 by 44 modules. 13 = Rectangle symbol of 48 by 48 modules. 14 = Rectangle symbol of 52 by 52 modules. 15 = Rectangle symbol of 64 by 64 modules. 16 = Rectangle symbol of 72 by 72 modules. 17 = Rectangle symbol of 80 by 80 modules. 18 = Rectangle symbol of 88 by 88 modules. 19 = Rectangle symbol of 96 by 96 modules. 20 = Rectangle symbol of 104 by 104 modules. 21 = Rectangle symbol of 120 by 120 modules.

		22 = Rectangle symbol of 132 by 132 modules. 23 = Rectangle symbol of 144 by 144 modules. 24 = Rectangle symbol of 8 by 18 modules. 25 = Rectangle symbol of 8 by 32 modules. 26 = Rectangle symbol of 12 by 26 modules. 27 = Rectangle symbol of 12 by 36 modules. 28 = Rectangle symbol of 16 by 36 modules. 29 = Rectangle symbol of 16 by 48 modules.

Remarks

You can refer to Appendix C: Symbologies for a description of the DataMatrix barcode symbology.

ForeColor

Description

Sets the foreground color of your barcode, used in conjunction with BackColor.

Return type

None

Syntax

Object.**ForeColor** = *value*

The ForeColor property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	String	Use hex values. Default is black (0x000000).

Remarks

This changes the color of the barcode, human readable text and comment text. If your barcode reader does not support the use of alternative color, you should leave this property using default settings.

I2of5OptionalCheckDigit

Description

Instructs Toolkit to calculate and append a checkdigit when generating an Interleaved 2 of 5 barcode.

Return type

None

Syntax

Object.**I2of5OptionalCheckDigit** = *value*

The I2of5OptionalCheckDigit property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Variant_Bool	False = No check digit is included. (Default) True = A check digit is calculated using the modulo 10 method and appended to the encoded data.

Remarks

For details on using the checkdigit, you can refer to Appendix C: Symbologies for a description of each barcode symbology.

LabelHeight

Description

Specifies the height of the label area.

Return type

None

Syntax

object.**LabelHeight** = *value*

The LabelHeight property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	A positive number. The unit of measurement determined by the Measurement property.

Remarks

In LDM, you will need to specify the height and width of the label area to prevent clipping. You can refer to Appendix C: Symbologies for a description of each barcode symbology.

LabelWidth

Description

Specifies the width of the label area.

Return type

None

Syntax

object.**LabelWidth** = *value*

The LabelWidth property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	A positive number. The unit of measurement determined by the Measurement property.

Remarks

In LDM, you will need to specify the height and width of the label area to prevent clipping. You can refer to Appendix C: Symbologies for a description of each barcode symbology.

MaxicodeClass

Description

Sets the class when generating a MaxiCode barcode in mode 2 or 3.

Return type

None

Syntax

object.**MaxicodeClass** = *value*

The MaxicodeClass property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	A 3-digit number. (Default is 1)

Remarks

You can refer to Appendix C: Symbologies for a description of each barcode symbology.

MaxicodeCountryCode

Description

Sets the country code when generating a MaxiCode barcode in mode 2 or 3.

Return type

None

Syntax

object.**MaxicodeCountryCode** = *value*

The MaxicodeCountryCode property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	3-digit number, which represents a country. The country codes are defined in ISO 3166.

Remarks

You can refer to Appendix C: Symbologies for a description of each barcode symbology.

MaxicodeMode

Description

Sets the mode when generating a MaxiCode barcode.

Return type

None

Syntax

object.**MaxicodeMode** = *value*

The MaxicodeMode property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Short	Modes 1 through 6 are available.

Remarks

You can refer to Appendix C: Symbologies for a description of each barcode symbology.

MaxicodeZipCode

Description

Sets the zip code when generating MaxiCode barcodes in mode 2 and 3.

Return type

None

Syntax

object.**MaxicodeZipCode** = *value*

The MaxicodeZipCode property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	String	Postal Code/Zip Code of the delivery address.

Remarks

You can refer to Appendix C: Symbologies for a description of each barcode symbology.

Measurement

Description

Sets the unit of measurement for properties that require units of length.

Return type

None

Syntax

object.**Measurement** = *value*

The Measurement property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	0 = Length is measured in mils (1/1000 inch). 1 =Length is measured in 1/1000th cm.

Remarks

Most of symbologies use English measurement units - based on mils (1/1000th inch). However, some symbologies use metric measurements or (1/1000th centimeter).

NOTE: It is recommended that you use the measurement unit specified by the standard to avoid float number calculation. You can refer to Appendix C: Symbologies for a description of each barcode symbology.

NarrowBarWidth

Description

Sets the ratio of a value for the width of the narrowest module in linear symbologies.

Return type

None

Syntax

object.**NarrowBarWidth** = *value*

The NarrowBarWidth property has these required parts:

Part	**Value Type**	**Description**
Object		An expression of the Barcode object.
Value	Double	A whole number between 0 and 1000. Default is 13. The unit of measurement determined by the Measurement property.

Remarks

This property is used in conjunction with the NarrowToWideRatio property.

NOTE: This property affects most linear symbologies. Height-modulated postal barcodes use fixed-pitch and will not be affected.

You can refer to Appendix C: Symbologies for a description of each barcode symbology.

NarrowToWideRatio

Description

Sets the ratio of the width of the wide module versus the narrow module.

NOTE: This property is valid only for Code 39, Code25, Code 11, Codabar and Interleaved 2 of 5 symbologies. All others ignore this property.

Return type

None

Syntax

object.**NarrowToWideRatio** = *value*

The NarrowToWideRatio property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Double	A value ranging from 2.0 to 3.0.

Remarks

Some linear symbologies can have two module widths. The width of the wide module width is a fixed multiple of the width determined by NarrowBarWidth.

NOTE: We do not recommend using a value below 2.5, as this may affect the readability of the barcode. We highly recommend you test barcode readability if you set this value below 2.5.

You can refer to Appendix C: Symbologies for a description of each barcode symbology.

PDFAspectRatio

Description

Sets the aspect ratio for a PDF417 barcode.

Return type

None

Syntax

object.**PDFAspectRatio** = *value*

The PDFAspectRatio property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Double	A number between 0 and n where n is undefined. n <1 = wide barcode. n = 1 = barcode is square. n > 1 = barcode is tall and thin.

Remarks

The PDFAspectRatio determines the overall shape of the PDF417 symbol and defines the overall height to width ratio. Higher values for the Aspect Ratio (greater than 1) produce tall, thin PDF417 bar codes and small values (greater than zero and less than 1) produce short, wide bar codes. A value of 1 produces approximately square bar codes. You can refer to Appendix C: Symbologies for a description of the PDF417 symbology.

PDFMaxCols

Description

Sets the maximum number of columns for a PDF417 barcode.

Return type

None

Syntax

object.**PDFMaxCols** = *value*

The PDFMaxCols property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	A number between 1 and 30.

Remarks

You can refer to Appendix C: Symbologies for a description of the PDF417 symbology.

PDFMaxRows

Description

Sets the maximum number of rows for a PDF417 barcode.

Return type

None

Syntax

object.**PDFMaxRows** = *value*

The PDFMaxRows property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	0 = Auto-select (Default). Valid range from 3 to 90.

Remarks

You can refer to Appendix C: Symbologies for a description of the PDF417 symbology.

PDFModuleHeight

Description

Sets the height of the smallest module for a PDF417 barcode.

Syntax

object.**PDFModuleHeight** = *value*

The PDFModuleHeight property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	A whole number between 1 and 100. Default is 30.

Remarks

You can refer to Appendix C: Symbologies for a description of the PDF417 symbology.

PDFModuleWidth

Description

Sets the width of the smallest module for a PDF417 barcode.

Syntax

object.**PDFSecurityLevel** = *value*

The PDFModuleWidth property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	A whole number between 1 and 100. Default is 13.

Remarks

You can refer to Appendix C: Symbologies for a description of the PDF417 symbology.

PDFSecurityLevel

Description

Sets the PDFSecurityLevel for a PDF417 barcode during generation.

Syntax

object.**PDFSecurityLevel** = *value*

The PDFSecurityLevel property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Short	Valid values range from 0 to 9. 9 = Automatic error correction. (Recommended)

Remarks

Each security level up to 8 incrementally adds overhead to a PDF417 barcode and thereby requires more symbol space. You can refer to Appendix C: Symbologies for a description of each barcode symbology.

PDFTruncatedSymbol

Description

Instructs Toolkit to truncate the PDF417 barcode during generation.

Syntax

object.**PDFTruncatedSymbol** = *value*

The PDFTruncatedSymbol property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Variant_Bool	True = produces a truncated version of all PDF417 barcodes. False = Does not product a truncated version. (Default)

Remarks

A truncated PDF417 symbol reduces the size of the PDF417 symbol by using a single termination bar for the stop pattern. The resultant barcode can encode the same information as a non-truncated PDF417, but is more susceptible to scanner errors caused by printing. You can refer to Appendix C: Symbologies for a description of each barcode symbology.

QuietZones

Description

Instructs Toolkit to include quiet zones when generating the barcode.

Syntax

object.**QuietZones** = *value*

The QuietZones property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Variant_Bool	True = include quiet zones. False = Quiet zones are not included (Default).

Remarks

Quiet zones help scanners determine where a barcode begins and ends. In a linear barcode, the quiet zones are the empty spaces that precede the start character and follow the stop character. In a 2-dimensional barcode, the quiet zone is the area around the barcode. The quiet zone added is 10 times the value of NarrowBarWidth for all linear symbologies, 2 times PDFModuleWidth value for PDF417 barcodes, 2 times of DataMatrixModuleSize value for DataMatrix barcodes and 1 element width for MaxiCode barcodes.

NOTE: Setting QuietZones to true substantially increase the barcode length for linear symbologies.

If you have set symbol margin properties you may safely set this property to False to better align the comment and human-readable text. You can refer to Appendix C: Symbologies for a description of each barcode symbology.

Rotation

Description

Instructs Toolkit to rotate the barcode to one of four 90-degree intervals.

Return type

None

Syntax

*object.*__Rotation__ = *value*

The Rotation property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	0 = (default) Zero degrees – no rotation. 1 = The working area is rotated 90 degrees counter-clockwise. 2 = The working area is rotated 180 degrees counter-clockwise. 3 = The working area is rotated 270 degrees counter-clockwise.

Remarks

Rotation affects label area.

ShowCheckDigit

Description

Instructs Toolkit to display the checkdigit in the human readable text.

Return type

None

Syntax

object.**ShowCheckDigit** = *value*

The ShowCheckDigit property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Variant_Bool	False = check digits are not shown. (Default) True = check digits are shown.

Remarks

Certain barcode formats require a checkdigit be contained in the barcode structure and will not be affected by the ShowCheckDigit property. These barcodes include UPC-A, UPC-E, EAN/JAN-13, EAN/JAN-8, Bookland and UCC/EAN 128. For details on using the checkdigit, you can refer to Appendix C: Symbologies for a description of each barcode symbology.

ShowComment

Description

Instructs Toolkit to display the comment text when generating barcodes.

Return type

None

Syntax

object.**ShowComment** = *value*

The ShowComment property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.

Value	Variant_Bool	True = comment is displayed (Default).
		False = comment is hidden.

Remarks

By default, Toolkit displays the comment text if it is included in your code. When displayed, the comment text margins are included in the size calculations.

NOTE: If you want to include the comment text margins in the calculation without displaying comment text, you will need to specify an empty string for the comment text string and ShowComment to True.

You can refer to Appendix C: Symbologies for a description of each barcode symbology.

ShowHRText

Description

Instructs Toolkit to display the human readable text when generating a barcode.

Return type

None

Syntax

object.**ShowHRText** = *value*

The ShowHRText property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Variant_Bool	True = The human readable portion is displayed. (Default) False = The human readable portion is hidden.

Remarks

By default, Toolkit generates human readable text with a barcode. Certain barcode formats do not require or accept human readable text. You can refer to Appendix C: Symbologies for a description of each barcode symbology.

SymbolMarginBottom

Description

Sets the width for the left symbol margin.

Syntax

object.**SymbolMarginBottom** = *value*

The SymbolMarginBottom property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	A positive number. The unit of measurement determined by the Measurement property.

Remarks

The symbol margin is the area between the label border and the area containing the barcode, human readable text, comment and comment margins. The chosen design mode may affect the use of the symbol margins. For additional details, refer to SymbolMarginLeft, SymbolMarginRight, and SymbolMarginTop.

SymbolMarginLeft

Description

Sets the width for the left symbol margin.

Syntax

object.**SymbolMarginLeft** = *value*

The SymbolMarginLeft property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	A positive number. The unit of measurement determined by the Measurement property.

Remarks

The symbol margin is the area between the label border and the area containing the barcode, human readable text, comment and comment margins. The chosen design mode may affect the use of the symbol margins. For additional details, refer to the SymbolMarginBottom, SymbolMarginRight and SymbolMarginTop properties.

SymbolMarginRight

Description

Sets the width for the right symbol margin.

Syntax

object.**SymbolMarginRight** = *value*

The SymbolMarginRight property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	A positive number. The unit of measurement determined by the Measurement property.

Remarks

The symbol margin is the area between the label border and the area containing the barcode, human readable text, comment and comment margins. The chosen design mode may affect the use of the symbol margins. For additional details, refer to the SymbolMarginBottom, SymbolMarginLeft and SymbolMarginTop properties.

SymbolMarginTop

Description

Sets the width for the top symbol margin.

Syntax

object.**SymbolMarginTop** = *value*

The SymbolMarginTop property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	A positive number. The unit of measurement determined by the Measurement property.

Remarks

The symbol margin is the area between the label border and the area containing the barcode, human readable text, comment and comment margins. The chosen design mode may affect the use of the symbol margins. For additional details, refer to the SymbolMarginBottom, SymbolMarginLeft and SymbolMarginRight properties.

Symbology

Description

Specifies the symbology or barcode format to generate.

Syntax

object.**Symbology** = *value*

The Symbology property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	0 = Code39. (Default) 1 = Code39 Full ASCII. 2 = Code39 Mod 43. 3 = Codabar. 4 = Code93. 5 = Code128. 6 = UCC/EAN 128. 7 = Interleaved 2 of 5 (ITF25). 8 = UPC-A. 9 = UPC-E. 10 = EAN/JAN-13. 11 = EAN/JAN-8. 12 = Bookland. 13 = Telepen. 14 = Telepen Numeric. 20 = PostNET. 21 = Planet. 22 = RoyalMail. 30 = MSI/Plessey. 31 = Code25. 32 = Code11. 40 = PDF417. 41 = DataMatrix. 42 = MaxiCode.

Remarks

The Toolkit Barcode object allows you to generate 24 different symbologies. For more information and usage requirements for each type, you can refer to Appendix C: Symbologies for a description of each barcode symbology.

TexAlignment

Description

If your barcode contains human readable text, you can specify the alignment for that text in relation to the barcode.

Syntax

object.**TexAlignment** = *value*

The TexAlignment property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Long	0 = Left alignment (default). Align the text with left edge of the comment box. 1 = Right alignment. Align the text with the right edge of the comment box. 2 = Center alignment. Align the text with the center of the comment box 3 = Justify alignment. Align the text to both edges of the comment box.

Remarks

Not all barcode scanners may support changing the alignment of the human readable text. For additional information, you can refer to Appendix C: Symbologies for a description of each barcode symbology.

TextOnTop

Description

Using this property, you can instruct Toolkit to place the human readable text above the barcode. The text will appear between the barcode and the top label border.

Syntax

object.**TextOnTop** = *value*

The TextOnTop property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	Variant_Bool	True = Text is placed above the symbol. False = Text is placed below the symbol (Default).

Remarks

Not all barcode scanners may support changing the placement of the human readable text. For additional information, you can refer to Appendix C: Symbologies for a description of each barcode symbology.

UccEanOptionalCheckDigit

Description

Determines if a checkdigit will be appended to your generated UCC barcode.

Return type

None

Syntax

object.**UccEanOptionalCheckDigit** = *value*

The UccEanOptionalCheckDigit property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.7
Value	Variant_Bool	True = check digit is included. False = check digit is not included (Default).

Remarks

For details on using the checkdigit, you can refer to Appendix C: Symbologies for a description of each barcode symbology.

Value

Description

The value property enables you to set the encoded value for the barcode.

Return type

None

Syntax

object.**Value** = *value*

The Value property has these required parts:

Part	Value Type	Description
Object		An expression of the Barcode object.
Value	String	The value to set.

Remarks

The barcode format selected in the Symbology property determines the information to encode. For example, if you were creating a POSTNET barcode, you might pass “92691” or “92691-001” to specify the postal code. You can refer to Appendix C: Symbologies for a description of each barcode symbology.

Appendix A: PDF Coordinate System

activePDF Toolkit contains many properties and methods that require an understanding of the PDF Coordinate System to determine space, location and the demarcation of dates. The following sections provide a brief overview of these measurements as they apply to Toolkit. For a complete overview of the PDF measurement system, you can refer to the PDF Specification Manual.

This section covers the following topics:

- PDF Units.
- PDF Document Coordinates.
- PDF Date Format.

PDF Units

The PDF specification defines space in a PDF file, where 72 PDF units is equal to 1 inch. For example, an 8 1/2" x 11" piece of paper would convert to 612 PDF units by 792 PDF units. You will need to round fractional sizes up to the nearest whole number.

PDF Document Coordinates

Before inserting headers or text onto a PDF page, you should become familiar the layout of the PDF page and the PDF coordinate system.

Defining the Space

Space in a PDF file, also known as user space, is measured in PDF Units. User space provides a way for PDF users and developers to maintain a single, easy to use system that a PDF Viewer automatically translates into device-dependent coordinates.

Origin and Positioning

In traditional programming languages, the coordinate system starts in the upper left corner (X and Y both increasing to the right and downward respectively). Conversely, the origin (0, 0) of a PDF document is in the lower-left corner, with Y increasing upward and X increasing to the right.

PDF Date Format

PDF documents use the internal date format: (D:YYYYMMDDHHmmSSOHH'mm'). The date format has these parts:

Part	Description
YYYY	The full four-digit year. (For example, 2004)
MM	The month from 01 to 12.
DD	The day from 01 to 31.
HH	The hour from 00 to 23.
mm	The minute from 00 to 59.
SS	The seconds from 00 to 59.
O	This is the relationship of local time to Universal Time (UT), denoted by one of the characters +, -, or Z.
HH'	The absolute value of the offset from UT in hours specified as 00 to 23.
mm'	The absolute value of the offset from UT in minutes specified as 00 to 59.

Appendix B: Runtime File Dependencies

A runtime file created using activePDF Toolkit requires specific files as part of the distribution.

Toolkit depends on the following to run properly:

- activePDF Toolkit Files.
- Runtime Files.
- System Files.

To request additional information or to purchase an activePDF Toolkit Runtime License, please contact activePDF Sales.

activePDF Toolkit Files

The following files are required for activePDF Toolkit to run properly:

- **APT352U.dll:** This is the main Toolkit .dll.
- **APTDBU.dll:** This component is only required if any database functions are used. (Requires the installation of MDAC 2.5.)
- **APTKIMGC.dll:** This is the image library, which is required if you are using the ImageToPDF, SetHeaderImage or PrintImage methods.
- **APToolkit.ocx:** This is the COM component. (Requires registration.)
- **APToolkitNET:** This is the .NET binding.
- **PVW32Cnv.dll:** This compliments the image library. PVW32Cnv.dll is required if you are using the ImageToPDF, SetHeaderImage or PrintImage methods.

Runtime Files

The following files are required for activePDF Toolkit to run properly:

- **MSVCRT.DLL:** This is the visual component lib.
- **MFC42.DLL:** This includes the foundation classes.

System Files

The following files are required for activePDF Toolkit to run properly:

- **KERNEL32.DLL:** This is the kernel API.
- **USER32.DLL:** This is the user context lib.
- **GDI32.DLL:** This is the graphics display interface.
- **OLEAUT32.DLL:** This is required for OLE Automation (App2App Communication).

Appendix C: Symbologies

Symbologies are systems of encoding data such that a scanner and/or a decoding system may together read and decode the data encoded in the barcode. Aside from the actual technique of encoding the bars and spaces, a number of technical specifications or characteristics define and separate one symbology from another. Each symbology represents a different barcode format.

Barcode technology is widely used across many industries. Most barcodes are machine-readable symbols that consist of vertical bars and spaces. The typical barcode also features quiet zones before and after, a start character, numerous data characters, numerous optional checkdigits and a stop character.

Each barcode format provides different capabilities when encoding your data as it defines the type of data. Toolkit generates the following three types:

- Numeric
- Alphanumeric
- 2-Dimensional (2D)

For a description of these types and additional terminology used with barcodes, refer to Barcode Terminology.

Barcode Terminology

This section is designed to provide an overview of the common terminology used to define barcode formats. These terms include:

- Character Set
- Label Area
- Symbol Margins
- Barcode
- Human Readable Text
- Comment
- Comment Margins
- Label Border
- Discrete
- Continuous
- Width
- Length
- Self-Checking
- Element
- Module
- Character
- Density
- Value X

Character Set

Character Set refers to what data a given barcode symbology can encode. Generally, there are three types of character sets:

- Numeric
- Alphanumeric
- Full ASCII

In general, a numeric character set produces the smallest barcode whereas a Full ASCII character set requires a larger space to encode the same data. However, Full ASCII gives you increased flexibility in encoding more types of information.

Numeric

A Numeric character set means the symbology can only encode numeric data from 0 through 9. Some additional characters may be encoded which are generally control features of the symbology, such as start/stop characters.

Alphanumeric

An Alphanumeric character set means the symbology can encode the digits 0 through 9 as well as alphabetic characters from A through Z. Some additional characters may be encoded which are generally control features of the symbology, such as start/stop characters.

Full ASCII

A Full ASCII character set is one that allows the encoding of the full ASCII character set. This symbology encodes any ASCII character, value 0 through 127.

Label Area

Label area describes the entire symbol, including the symbol margins, the barcode, the human readable text, the comment and comment margins. The label area does not include the label border.

Symbol Margins

The symbol margins define the area between the label border and the area composed of the barcode, human readable text, comment and comment margins.

Barcode

The barcode is the symbol you scan. This does not include the human readable text, except when the human readable text is required in the barcode.

Human Readable Text

Human readable text is the displayed barcode value. This can be the actual barcode value or a numerical representation and can appear above or below the barcode. Visible human readable text invalidates certain barcodes.

Comment

A comment is additional text displayed with the barcode, but not included in the encoded value. Comments appear above or below the barcode, within the label area. You can place comment text above or below the barcode and define separate comment margins.

Comment Margins

Comment margins are the defined area around the comment text. The comment margins are determined in relation to the type and size of barcode.

Label Border

The label border is the perimeter of the label area. Changes in the label border should not affect the label area or barcode.

Discrete

In a discrete symbology, the interpretation of each character encoded does not rely on the rest of the barcode. Such symbologies have characters that both start and end with a bar. Some amount of inter-character spacing or inter-character gap separates each character. The inter-character gap carries no information-the only duty of the inter-character spacing is to separate the characters.

Continuous

In a continuous symbology, the interpretation of each character encoded relies on the rest of the barcode. This is because characters start with a bar and end with a space. The final space is "terminated" by the starting bar of the next character. Individually, there is no way to know how wide the last space is without knowing where the next character begins. Continuous symbologies normally implement some kind of special termination bar or the termination bar such that the last space of the last data character terminates termination sequence.

Width

A Two-Width symbology has spaces and bars that are either wide or narrow. This has the benefit of simplicity-once you determine the width of a narrow bar, anything over a certain width is "wide". This allows for a large level of print tolerance in lower-quality printing conditions.

A Multiple-Width has bars and spaces that may be of three or more widths. The narrowest bar or space may be X in width, a medium-width space or bar may be 2X in width, and a wide bar may be 3X in width. Since there are more possible combinations available in a multiple-width symbology, data encoding is often more efficient and results in a tighter barcode. Multiple-width symbologies are usually continuous.

Length

A fixed-length symbology encodes a certain number of characters or digits. For example, a UPC-A barcode always encodes 12 digits of data. An application may not encode less or more than the pre-defined fixed-length of 12 characters. The symbology itself defines the length of data.

A variable-length symbology encodes a message of any length. For example, Code128 encodes any number of characters that can reasonably fit physically in the printed barcode. The symbology itself does not define how many characters to encode.

Self-Checking

A Self-checking symbology means a single printing or scanning error will not cause one of the component characters to convert into another valid character.

NOTE: Self-checking does not infer self-correcting.

Element

An element is any individual bar or space.

Module

A module is the amount of space a bar or space takes up measured in X's. For example, a narrow bar is 1X, thus the narrow bar takes up one module. A medium-size bar may be 2X in width, thus it would take up two modules. A wide bar may be 3X in width, thus it takes up three modules.

Character

A character is a sequence of elements (bars and spaces) which, taken together, encode a single logical value. Often, each character in a barcode is a fixed number of modules in length.

Density

Density is the number of characters encoded per inch given a certain X value. The smaller the X value, the more characters encoded in an inch of a barcode.

Value X

The value "X" is the "perfect" width of the barcode's narrowest element. The value of X must remain constant throughout a single barcode.

Numeric Symbologies

Toolkit can encode the following Numeric barcodes:

- Codabar
- Interleaved 2 of 5
- UPC-A
- UPC-E
- EAN/JAN-13
- EAN/JAN-8
- Bookland
- Telepen
- Telepen Numeric
- POSTNET
- Planet
- RoyalMail
- MSI/Plessey
- Code25
- Code 11

Codabar

A.K.A.

NW-7 (Japan, Narrow and Wide), JIS X 0503 (Japan), Rationalized Codabar, USD-4, 2 of 7 code

Overview

Developed in 1972, Codabar is discrete symbology , commonly used by the US blood banks and photo labs. FedEx® also uses a variation of Codabar for its Airbills.

Encoding

Codabar can encode the following 16 characters:

- **10 digits:** 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9
- **6 special characters:** -, $, :, /, .,+

Structure

Codabar has the following structure:

- A start character - one of (A, B, C, or D).
- Inter-character gap.
- Encoded value.
- A stop character - one of (A, B, C, or D).

Requirements

To ensure quality, the width of the inter-character gap should be equal to the width of the narrowest element (X). The minimum value of X is 7.5 mils. The wide-to narrow ration (N) must be between 2.0 and 3.0. N remains constant. If X is less than 20 mils, N must be greater than 2.2. The barcode height must be at least 15 percent of the barcode length or .25 inches. Use the greater of these measurements

Interleaved 2 of 5

A.K.A.

ITF, ITF-14, I 2 of 5

Overview

Interleaved 2 of 5 is a continuous, high-density, variable length symbology for encoding numeric values. Distribution industries are the primary users of Interleaved 2 of 5.

Encoding

Interleaved 2 of 5 can encode **10 digits:** 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

Structure

Interleaved 2 of 5 has the following structure:

- A start character with format bar space bar space.
- Encoded value.
- Checkdigit – Modulo 10 (Optional)
- A stop character with format bar bar space bar.

Requirements

Interleaved 2 of 5 encodes 2 characters in a unit of 5 bars and spaces. The symbology encodes even position characters as bars and odd position characters as spaces. As a result of the encoding, the total length of the digits to encode must be even in length. If a check digit is used, the total length is odd.

If the barcode does not meet the length requirement, Toolkit automatically appends a 0 (zero) to the encoded data.

Remarks

You can use the I2of5OptionalCheckDigit property to set or retrieve the checkdigit. Toolkit will automatically calculate and append the check digit in the correct location, based on the encoded value.

UPC-A

Overview

UPC A is one of the most common barcodes used in the United States, with variations appearing on most consumer goods and periodicals.

NOTE: For a UPC-A barcode to be valid, you must apply for a manufacturer code from the UCC.

Encoding

UPC-A can encode **10 digits**: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. You can use an additional pipe("|") to encode supplemental data. Refer to remarks below for additional information.

Structure

UPC-A has the following structure:

- Start guard bars (always bar+space+bar).
- Left half, 6 digits encoded using the encoding schema A or B.
- Center guard bars (space+bar+space+bar+space).
- Right half, 6 digits encoded using the encoding schema C.
- Stop guard bars (always with a pattern bar+space+bar).

The barcode structure expresses four main components, the number system, manufacturer code, product code and checkdigit.

Number system: In the human readable text, the number system appears to the left of the start guard bar. The number system adheres to the following values:

- **0** - Regular UPC code.
- **1** - Reserved.
- **2** - Weight Items.
- **3** - Drug/Health Items.
- **4** - In-store use on non-food items.
- **5** - Coupons.
- **6** - Reserved.
- **7** - Regular UPC code.
- **8** - Reserved.
- **9** – Reserved.

Manufacturer code: In the human readable text, the manufacturer code appears between the start and center guard bars. The manufacturer code is assigned by the UCC.

Product code: In the human readable text, the product code appears between the center and stop guard bars. Each manufacturer assigns the product code, which provides 99,999 different combinations. The UCC must approve a product code designation.

Checkdigit: In the human readable text, the checkdigit appears to the right of the stop guard bar. Toolkit automatically calculates and appends the checkdigit to your 11 digit encoding.

Requirements

A UPC-A barcode must contain 11 digits plus a checkdigit. Toolkit automatically calculates and appends the checkdigit based on your encoded value.

Remarks

The UPC-A barcode format allows for additional supplemental data, which you append to the end of the encode value using a pipe character ("|"). For example, if you specify "90123678812" as your 11-digit value and "02" for your 2-digit supplemental value, you would change the value to "90123678812|02"

For more information on supplemental barcodes, refer to the Supplemental Barcodes section.

UPC-E

Overview

UPC E is a truncated form of the UPC-A symbology with the zeros suppressed to reduce the length for smaller packaging.

NOTE: For viable compression, the parent UPC-A barcode must contain four zeros in the 11-digit value.

Encoding

UPC-E can encode **10 digits:** 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. You can use an additional pipe("|") to encode supplemental data. Refer to remarks below for additional information.

Structure

UPC-E has the following structure:

- Start guard bars (always with a pattern bar+space+bar).
- 5 digits calculated from the equivalent UPC number.
- Check digit.
- Stop guard bars (always with a pattern bar+space+bar).

Requirements

Each UPC-E barcode contains six digits with an implied number system of 0 (zero). For conversion to a UPC-E barcode, the parent UPC-A barcode must contain at least four zeros. If the parent UPC-A is valid, you can convert the value to UPC-E using following guidelines in order:

1. **Manufacturer codes ending in 000, 100 or 200** – The new barcode consists of the first two digits of the manufacturer code and the last three digits of the product code (must be equal to or between 000 and 999), followed by the third digit of the manufacturer code.
2. **Manufacturer codes ending in 00** – If the first guideline is not applicable, the new barcode consists of the first three digits of the manufacturers code and the last two digits of the product code (must be equal to or between 00 and 99), followed by the number 3 (three).
3. **Manufacturer codes ending in 0** – If the previous guidelines do not apply, the new barcode consists of the first four digits of the manufacturer code and the last digit of the product code (must be equal to or between 0 and 9), followed by the number 4 (four).
4. **Manufacturer codes ending in a non-zero digit** – If the previous guidelines do not apply, the new barcode consists of the manufacturer code and the last digit of the product code (must be equal to or between 5 and 9).

Toolkit automatically calculates and appends the checkdigit based on your encoded value.

Remarks

The UPC-E barcode format allows for additional supplemental data, which you append to the end of the encode value using a pipe character ("|"). For example, if you specify "425261" as your 1 value and "55999" for your 5-digit supplemental value, you would change the value to "425261|55999"

For more information on supplemental barcodes, refer to the Supplemental Barcodes section.

EAN/JAN-13

A.K.A.

JAN (Japanese Numbering Authority, based on numbering system)

Overview

EAN/JAN-13 is a superset of UPC-A implemented by the International Article Numbering Association (EAN) in Europe. Unlike UPC-A, EAN/JAN-13 includes country code specification for international use. As EAN-13 is a superset of UPC-A, any barcode software capable of reading EAN-13 can also read UPC-A with minor differences.

Encoding

EAN/JAN-13 can encode **10 digits:** 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. You can use an additional pipe ("|") to encode supplemental data. Refer to remarks below for additional information.

Structure

EAN/JAN-13 has the following structure:

- Start guard bars (always bar+space+bar).
- Left half - 7 digits encoded using the encoding schema A or B.
- Center guard bars (space+bar+space+bar+space).
- Right half - 6 digits encoded using the encoding schema C.
- Stop guard bars (always with a pattern bar+space+bar).

The barcode structure expresses four main components, the number system, manufacturer code, product code and checkdigit.

Number system: In the human readable text, the number system appears to the left of the start guard bar and continues into the first one or two digits after the guard bar. The number system indicates a numbering authority or region that assigns the manufacturer code. For a list of the current number system, refer to Appendix D: Barcode Tables.

Manufacturer code: In the human readable text, the manufacturer code appears between the start and center guard bars. The numbering authority assigns the manufacturer code.

Product code: In the human readable text, the product code appears between the center and stop guard bars. Each manufacturer assigns the product code, which provides 99,999 different combinations.

Checkdigit: In the human readable text, the checkdigit appears to the right of the stop guard bar. Toolkit automatically calculates and appends the checkdigit to your 12 digit encoding.

Requirements

A EAN/JAN-13 barcode must contain 12 digits plus a checkdigit. Toolkit automatically calculates and appends the checkdigit based on your encoded value.

The nominal X dimension is 13 mils and the printable X dimension ranges from 10.4 to 24 mils.

Remarks

The EAN/JAN-13 barcode format allows for additional supplemental data, which you append to the end of the encode value using a pipe character ("|"). For example, if you specify "901236788122" as your 12-digit value and "02" for your 2-digit supplemental value, you would change the value to "901236788122|02"

For more information on supplemental barcodes, refer to the Supplemental Barcodes section.

EAN/JAN-8

A.K.A.

JAN (Japanese Numbering Authority, based on numbering system)

Overview

EAN/JAN-8 is a truncated version of the EAN/JAN-13 symbology, where all eight digits are explicitly encoded. Unlike the relation between UPC-E and UPC-A, the EAN-8 is explicitly encoded. Additionally, EAN/JAN-8 enabled scanners may not read UPC-E barcodes.

Encoding

EAN/JAN-8 can encode **10 digits:** 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. You can use an additional pipe ("|") to encode supplemental data. Refer to remarks below for additional information.

Structure

EAN-8 has the following structure:

- Start guard bars (always with a pattern bar+space+bar).
- Two number system characters, encoded with character set A.
- The following two characters, encoded with character set A.
- Center guard bars (with a pattern space+bar+space+bar+space).
- Last three characters, encoded in character set C.
- Checkdigit
- Stop guard bars (always with a pattern bar+space+bar).

The barcode structure expresses three main components, the number system, the product code and checkdigit.

Number system: In the human readable text, the number system is the first two or three digits to the right of the start guard bar. The number system indicates the number authority or region that assigns the manufacturer code. For a list of the current number system, refer to Appendix D: Barcode Tables.

Product code: In the human readable text, the product code begins after the number system and continues past the center guard bar. The product code indicates the product.

Checkdigit: In the human readable text, the checkdigit appears to the right of the stop guard bar. Toolkit automatically calculates and appends the checkdigit to your 7 digit encoding.

Requirements

Each EAN/JAN-8 barcode contains seven digits of encoded value followed by a check digit. Toolkit automatically calculates and appends the checkdigit based on your encoded value.

Remarks

The EAN/JAN-8 barcode format allows for additional supplemental data, which you append to the end of the encode value using a pipe character ("|"). For example, if you specify "71245126" as your 12-digit value and "95000" for your 5-digit supplemental value, you would change the value to "71245126|95000"

For more information on supplemental barcodes, refer to the Supplemental Barcodes section.

Bookland

A.K.A.

ISBN (International Standard Book Number)

Overview

Bookland barcodes are EAN/JAN-13 barcodes that use a specific format exclusive to books. Generating a bookland barcode is the same as generating an EAN/JAN-13 barcode using the 978 numbering system.

Encoding

Bookland barcodes can encode **10 digits:** 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. You can use an additional pipe ("|") to encode supplemental data. Refer to remarks below for additional information.

Structure

Bookland has the following structure:

- Start guard bars (always bar+space+bar).
- Left half - 7 digits encoded using the encoding schema A or B.
- Center guard bars (space+bar+space+bar+space).
- Right half - 6 digits encoded using the encoding schema C.
- Stop guard bars (always with a pattern bar+space+bar).

The barcode structure expresses four main components, the number system, manufacturer code, product code and checkdigit.

Number system: In the human readable text, the number system appears to the left of the start guard bar and continues into the first one or two digits after the guard bar. The number system for Bookland is always 978. If you specify the Bookland barcode type, Toolkit automatically applies the 978 number system.

Manufacturer code: In the human readable text, the manufacturer code appears between the start and center guard bars. The numbering authority assigns the manufacturer code.

Product code: In the human readable text, the product code appears between the center and stop guard bars. Each manufacturer assigns the product code, which provides 99,999 different combinations.

Checkdigit: In the human readable text, the checkdigit appears to the right of the stop guard bar. Toolkit automatically calculates and appends the checkdigit to your 12 digit encoding.

Requirements

Bookland barcodes always use the 978 number system and the remainder of the barcode consists of specific parts from the book's ISBN number.

An ISBN is a 10-digit number preceded by the letters ISBN. Typically, an ISBN uses an OCR-A font. The number consists of four parts with variable length, separated by hyphens or spaces. The parts in order are the Group Identifier, Publisher Identifier, Title Identifier and checkdigit. Toolkit does not generate ISBN numbers. Regional agencies in every country assign ISBNs.

The Bookland barcode value consists of the 978 number system and the entire ISBN, minus the checkdigit plus a checkdigit generated by Toolkit. For example, if the ISBN of a book were "968-26-1240-3". The complete Bookland code generated by Toolkit would be "9789682612404".

Remarks

The Bookland barcode format allows for additional supplemental data, which you append to the end of the encode value using a pipe character ("|"). For example, if you specify "71245126" as your 12-digit value and "95000" for your 5-digit supplemental value, you would change the value to "71245126|95000"

Additionally, you will need to pay attention to the following values when generating 5-digit supplemental barcodes:

- **90000** – The value indicates a book has no suggested retail value.
- **99991** – The value indicates the book is a complimentary copy.
- **90001 to 98999** – Publishers use these values for internal purposes.
- **99990** – The value indicates the book is "Used" as dictated by the National Association of College Stores.

For more information on supplemental barcodes, refer to the Supplemental Barcodes section.

Telepen

Overview

Developed in 1972, Telepen barcode enable representation of all 128 standard ASCII character using limited space. The United Kingdom, universities and libraries use Telepen.

Encoding

Telepen can encode all 128 Standard ASCII values using a three-digit ASCII code, preceded by the \ character. For example, the carrot symbol ("^") corresponds to ASCII code \094. For a complete list of ASCII values, refer to Appendix D: Barcode Tables.

Structure

Telepen has the following structure:

- A start character.
- Encoded value.
- Check digit.- modulo 127
- Stop Character.

Requirements

To meet conventional printing requirements, Telepen barcodes have a wide to narrow bar ratio of 3:1. Telepen contains no inter-character gaps, but the specification allows them. The Telepen barcodes generated by Toolkit do not contain inter-character gaps. Toolkit automatically calculates and applies the checkdigit.

Telepen Numeric

Overview

Telepen Numeric is a variation of the Telepen barcode that allows for double density encoding of numeric values. This results in twice the packing density.

Encoding

Telepen Numeric encodes **10 digits:** 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

Structure

Telepen Numeric has the following structure:

- A start character.
- Message encoded.
- Checkdigit – modulo 127.
- Stop Character.

Requirements

To meet conventional printing requirements, Telepen barcodes have a wide to narrow bar ratio of 3:1. Telepen contains no inter-character gaps, but the specification allows them. The Telepen barcodes generated by Toolkit do not contain inter-character gaps. Toolkit automatically calculates and applies the checkdigit.

PostNET

Overview

Developed by the United States Post Office, PostNET (Postal Numeric Encoding Technique) is used to encode ZIP information.

NOTE: There are specific placement requirements when using PostNET barcodes. The complete details are available on the United States Postal Service website.

Encoding

PostNET can encode **10 digits:** 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

Structure

PostNET has the following structure:

- Frame bar (long).
- Encoded value - 5, 9, or 11 data characters.
- Check digit.
- Final frame bar (long).

Requirements

Each PostNET barcode consists of a series of five bars, constructed from shorts (encoded as 0) and longs (encoded as 1). The barcode is enclosed by two long frame bars surrounding the encoded address information and check digit. Toolkit automatically calculates and applies the checkdigit as needed. Depending on the form of encoding used, A PostNET barcode will have 32, 52, or 62 bars including the frame bars and check digit.

A US deliver address is encoded as a 5-digit ZIP (For example, 92691), 5-digit ZIP + 4 code (92691-6314), 11-digit delivery point code (92691-6314-05). If you specify a full address in your encoded value, Toolkit will ignore the non-numerical information. For example, if you specify "Mission Viejo, CA 92691-6314", the barcode will only encode using "92691-6314".

PostNET is different from other barcodes, as the information is encoded using the barcode height rather than the relation between the bars and spaces.

Toolkit generates PostNET barcodes according to the USPS standard requirements. By default, Toolkit displays the human readable text. You will need to set ShowHRText equal to false to comply with the standard requirements.

Planet

Overview

The United States Post Office developed PLANET barcode based on the POSTNET barcode format. PLANET barcodes use additional tracking numbers to enable the USPS confirmation services.

NOTE: There are specific placement requirements when using PLANET barcodes. The complete details are available on the United States Postal Service website.

Encoding

Planet can encode **10 digits:** 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

Structure

PLANET has the following structure:

- Frame bar (long).
- Service designation - (21=Origin Confirm or 22=Destination Confirm).
- Encoded value - 9 data characters (5-digit zip + 4-digit code).
- Check digit.
- Final frame bar (long).

Requirements

Each PLANET barcode consists of shorts (encoded as 0) and longs (encoded as 1). The barcode is enclosed by two long frame bars surrounding the service designation, encoded address information, and check digit. Toolkit automatically calculates and applies the checkdigit.

Toolkit generates PLANET barcodes according to the USPS standard requirements. By default, Toolkit displays the human readable text. You will need to set ShowHRText equal to false to comply with the standard requirements.

Remarks

The services can confirm that customers received the mail, allowing mailers to synchronize telemarketing programs with direct mail campaigns or augment other advertising media with their mailings.

RoyalMail

A.K.A.

RM4SCC (Royal Mail 4-state Customer Code), UK Royal Mail

Overview

RoyalMail barcodes are height-modulated symbologies used for automated mail sorting and processing.

NOTE: Detailed printing and placement requirements are available from the United Kingdom postal service.

Encoding

RoyalMail can express the following 36 characters:

- **26 upper-case letters:** A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, and Z.
- **10 digits:** 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

Structure

RoyalMail has the following structure:

- Start bar.
- Encoded value.
- Check digit.
- Stop bar.

Requirements

RoyalMail barcodes consist of a continuous string of characters with no space characters encoded as bars. The format has a density of 20 to 24 bars per 25.4 centimeters. Each bar can consist of Ascenders, Trackers and Descender. If you divide the individual bar height into thirds, the ascenders are the top third, the trackers are the middle and the descenders are the bottom. Each bar must contain a tracker.

The start and stop bars use a special encoding, consisting of a tracker and ascender or a full bar. The start and stop bars are unique, enabling the barcode to be read in any direction. Toolkit automatically encodes the start and stop bars. The quiet zone must be at least 2mm before the start bar and after the stop bar.

Toolkit automatically calculates and applies the checkdigit. Toolkit generates RoyalMail barcodes according to the standard requirements. By default, Toolkit displays the human readable text. You will need to set ShowHRText equal to false to comply with the standard requirements.

MSI/Plessey

A.K.A.

Plessey Code, MSI Code, Texlon Code, Anker Code, Modified Plessey

Overview

MSI is a modified form of the Plessey Code that has been widely used in the retail industry. The barcode format is not as prevalent as it once was and remains for software compatibility.

Encoding

MSI/Plessey encodes **10 digits:** 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

Structure

MSI/Plessey has the following structure:

- Start character.
- Encoded value.
- Check digit(s).
- Stop character.

Requirements

MSI/Plessey barcodes have a fixed-length and do not perform self-checking. A start, stop, and check digit character are required in every barcode.

MSI/Plessey can make use of two check digits, but the choice to implement or read the digit depends on the application. Toolkit automatically calculates and applies the checkdigit.

Code 25

A.K.A.

Code 2 of 5, Industry 25, Industry 2 of 5, Industrial 25, Industrial 2 of 5, Standard 25, Standard 2 of 5

Overview

Invented in the early 1960s, Code 25 has been widely adopted for use in the warehouse, photo finishing and airline industries. The barcode format is not as prevalent as it once was and remains for software compatibility.

Encoding

Code 25 encodes the following **10 digits:** 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

Structure

Code 25 has the following structure:

- Start character.
- Encoded value.
- Check digit(s).
- Stop character.

Requirements

Each character in a Code 25 barcode contains 5 bars with 2 bars being wide. The data encodes in the bar width resulting in a very low density.

Toolkit automatically calculates and applies the checkdigit.

Code 11

A.K.A.

USD -8

Overview

Developed in 1977, Code 11 is a discrete symbology that uses high-density numerical encoding. The most common implementation of the barcode is for the labeling of telecommunications components. The barcode format is not as prevalent as it once was and remains for software compatibility.

Encoding

Code 11 can encode the following 10 characters and 1 special character:

- **10 digits:** 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.
- **1 special character:** -(dash)

Structure

Code 11 has the following structure:

- Start character.
- Encoded value.
- Checkdigit C.
- Checkdigit K (required if the encoded value is longer than 10 digits).
- Stop character.

Requirements

The height of the barcode must be at least 0.15 times the length of the barcode or 0.25 inches. You will need to choose the greater of the two measurements, whichever is greater.

Toolkit will automatically calculate and apply one or two check digits, based on the amount of digits in the encoded value. The barcode is not self-checking, as printing defects can alter one character into another character very easily.

Alpha-numeric Symbologies

Toolkit can encode the following alphanumeric barcodes:

- Code128
- UCC/EAN 128
- Code 39
- Code39 Full ASCII
- Code39 Mod 43
- Code93

Code 128

A.K.A.

USS 128, C-128

Overview

Introduced in the early 1980's, Code 128 is a high-density alphanumeric symbology. Many industries quickly adopted the format for its size and encoding abilities.

Encoding

Code128 can encode all 128 Standard ASCII values using a three-digit ASCII code, preceded by the \ character. For example, the carrot symbol ("^") corresponds to ASCII code \094. For a complete list of ASCII values, refer to Appendix D: Barcode Tables.

Structure

Code 128 has the following structure:

- A start character.
- Message encoded.
- Checkdigit.
- Stop Character.
- Termination bar (bar+space+bar).

Requirements

Code 128 is a continuous, variable length barcode with multiple element widths and a checkdigit. All characters contain three bars and spaces for a total of 11 modules. Toolkit automatically calculates and applies the checkdigit.

For an open system, the minimum value of the narrowest element (X) is 7.5 mils and the barcode height is 15 percent of the length or 0.25 inches. You must use the greater of the two measurements. The leading and trailing quiet zone must be at least 0.25 inches.

Code128 makes use of three shift characters, A, B and C. These shift characters enable different meanings for each encoded character – increasing the amount of information you can encode in the barcode. Toolkit automatically applies the shift characters based on the encoded value.

UCC/EAN 128

A.K.A.

UCC/EAN128, UCC 128 (Uniform Code Council), EAN 128 (European Article Numbering)

Overview

Derived from Code128, UCC/EAN 128 allows for the encoding of data and the meaning of the data. UCC/EAN 128 is not a true barcode - it is a standard for defining data and formatting using Code128.

NOTE: Refer to the UCC/EAN website for guidelines on UCC/EAN 128 usage requirements.

Encoding

UCC/EAN 128 can encode all 128 Standard ASCII values using a three-digit ASCII code, preceded by the \ character. For example, the carrot symbol ("^") corresponds to ASCII code \094. For a complete list of ASCII values, refer to Appendix D: Barcode Tables.

In addition, UCC/EAN 128 uses **2 Special Characters:** (,). These enclose the Application Identifier in the encoded value, but do not encode into the barcode.

Structure

UCC/EAN 128 has the following structure:

- A code128 start character (START-A, START-B or START-C).
- A code128 FNC1 character.
- Application Identifier (AI).
- Encoded Value.
- Checkdigit.
- Stop Character.
- Termination bar.

The barcode structure expresses three main components, the Application Identifier, the encoded value and the checkdigit.

Application Identifier: In the human readable text, the AI appears enclosed in parentheses. The AI signifies a specific meaning for the data that follows the identifier. You can encode multiple application identifiers in a single barcode. Subsequent AIs become field identifiers. For a complete list of identifiers and meaning, refer to Application Identifier Values.

Encoded value: In the human readable text, the encoded value in a UCC/EAN 128 barcode appears directly after the enclosed AI. The value encoded and the digit requirements are subject to the AI. For a complete list of requirements, refer to Application Identifier Values.

Checkdigit: In the human readable text, the checkdigit appears as the last digit in the barcode. When UccEanOptionalCheckDigit is equal to true, Toolkit automatically calculates and appends the checkdigit.

Requirements

UCC/EAN 128 is a continuous, variable length barcode with multiple element widths and a checkdigit. All characters contain three bars and spaces for a total of 11 modules.

For an open system, the minimum value of the narrowest element (X) is 7.5 mils and the barcode height is 15 percent of the length or 0.25 inches. You will need to use the greater measurement. The leading and trailing quiet zone must be at least 0.25 inches.

To encode information, you will need to use the following format in your value:

```
(AI) encoded value(FID1)encoded value 1(FID2)encoded value 2…(FIDn)encoded value n
```

For example, if you encode Article Number "19421123450011", Best Before Date "991231" and Batch Number "101234", you need the AI "01", FID "15" and FID2 "10". You would specify "(01)19421123450011(15)1991231(10)101234" for the value.

Code 39

A.K.A.

Code 3 of 9, AIAG (Automobile Industry Action Group), USS (Uniform Symbol Specification) code 39, USD-3

Overview

Developed for non-retail environments, Code 39 is a discrete symbology that encodes alphanumeric information such as model numbers.

Encoding

Code 39 encodes the following 43 characters:

- **26 upper-case letters:** A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, and Z.
- **10 digits:** 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.
- **7 special characters:** -, ., *, $, /, +, %, SPACE.

Structure

Code 39 has the following structure:

- A start character - asterisk(*).
- Encoded value.
- Checkdigit - optional
- A stop character - asterisk(*).

Requirements

Each character consists of up to five bars and four spaces, making nine elements where three out of nine elements are wide   The height of the bars must be at least 0.15 times the barcode's length or .25 inches. You must use the greater measurement. To ensure quality, the width of the inter-character gap should equal the width of the narrowest element (X). The leading and trailing quiet zones must be at least 10 times the length of the narrowest element (X) or .10 inches. You must use the greater measurement.

When Code39OptionalCheckDigit is equal to true, Toolkit automatically calculates and encodes the checkdigit.

Depending on the barcode scanner, the start and stop asterisks must be included in the human readable text. Consult your scanner manual for more information. By default, Toolkit includes the asterisks in the generated human readable text. You do not need to include the characters in your value. If you require the characters, set Code39StartStopChars equal to false.

Code 39 Full ASCII

A.K.A.

Code 39 extended, Code 39 Full

Overview

Code 39 Full ASCII is a variant of Code39 that enables encoding of all 128 characters in the ASCII table. Code 39 Full ASCII is discrete.

Encoding

Code 39 Full ASCII can encode all 128 Standard ASCII values using a three-digit ASCII code, preceded by the \ character. For example, the carrot symbol ("^") corresponds to ASCII code \094. For a complete list of ASCII values, refer to Appendix D: Barcode Tables.

Structure

Code 39 Full ASCII has the following structure:

- A start character - asterisk(*).
- Encoded value.
- Checkdigit - optional
- A stop character - asterisk(*).

Requirements

Each character consists of up to five bars and four spaces, making nine elements where three out of nine elements are wide The height of the bars must be at least 0.15 times the barcode's length or .25 inches. You must use the greater of the two measurements.

To ensure quality, the width of the inter-character gap should equal the width of the narrowest element (X). The leading and trailing quiet zones must be at least 10 times the length of the narrowest element (X) or .10 inches. You must use the greater of the two measurements.

When Code39OptionalCheckDigit is equal to true, Toolkit automatically calculates and encodes the checkdigit.

Remarks

You will need to adjust your barcode reader to extended mode to read Code 39 Full ASCII barcodes accurately.

Code 39 Mod 43

A.K.A.

HIBC (Health Industry Bar Code), LOGMARS (Logistics Applications of Automated Marking and Reading Symbols), HIBC Code 39

Overview

Code 39 Mod 43 is similar to Code 39 with the addition of a modulo 43 check digit appended to the last character. This enables very high level of accuracy. The health industry uses the Mod 43 version of Code39 frequently.

Structure

Code 39 Mod 43 has the following structure:

- A start character - plus (+).
- Encoded value.
- Checkdigit – required
- A stop character - asterisk (*).

Each character consists of up to five bars and four spaces, making nine elements where three out of nine elements are wide The height of the bars must be at least 0.15 times the barcode's length or .25 inches. You must use the greater of the two measurements.

To ensure quality, the width of the inter-character gap should equal the width of the narrowest element (X). The leading and trailing quiet zones must be at least 10 times the length of the narrowest element (X) or .10 inches. You must use the greater of the two measurements.

If you are creating a Code 39 Mode 43 barcode, you must set Code39OptionalCheckDigit equal to true.

Code 39 Mod 43 must use a plus (+) symbol for the leading character. Toolkit adds the plus sign automatically if the encoded value does not contain one.

Code 93

A.K.A.

USS-93. (Uniform Symbol Specification)

Overview

Code 93 is an alphanumeric, variable length symbology designed to offer higher density and data security than Code 39. The Canadian Postal Service uses code 93 when encoding supplemental delivery information.

Encoding

Code 93 can encode all 128 Standard ASCII values using a three-digit ASCII code, preceded by the \ character. For example, the carrot symbol ("^") corresponds to ASCII code \094. For a complete list of ASCII values, refer to Appendix D: Barcode Tables.

Structure

Code 93 has the following structure:

- A start character - asterisk(*).
- Encoded value.
- First checkdigit "C".
- Second checkdigit "K".
- Stop Character - asterisk(*).
- Termination bar.

Requirements

Each Code 93 character contains nine modules arranged into three bars with adjacent spaces.

For an open system, the minimum value of the narrowest element (X) is 7.5 mils and the barcode height is 15 percent of the length or 0.25 inches. You must use the greater measurement. All characters end with a space - the termination bar appears after the last character (stop character) to determine the width of the trailing quiet zone. The leading and trailing quiet zone must be at least 0.25 inches. Toolkit automatically applies the asterisks in the encoded value and the checkdigits.

2-Dimensional (2D) Symbologies

Toolkit can encode the following 2D Symbologies:

- PDF417
- DataMatrix
- MaxiCode

PDF 417

Overview

PDF417 barcodes are multi-row and variable-length, which allows for high data capacity and error-correction capabilities. The barcode format can encode over 1100 bytes, 1800 text characters or up to 2710 digits. Linear, laser and two-dimensional scanners can read PDF417 barcodes. Additionally, the barcode format supports bidirectional decoding.

Encoding

PDF417 can encode all 128 Standard ASCII values using a three-digit ASCII code, preceded by the \ character. For example, the carrot symbol ("^") corresponds to ASCII code \094. For a complete list of ASCII values, refer to Appendix D: Barcode Tables.

Structure

It is common for a PDF417 barcode to contain from 3 to 90 rows, which are 90 to 583x in width. Each row is read from left to right, and has the following structure:

- Leading quiet zone.
- Start Pattern.
- Left row indicator symbol character.
- 1 to 30 data symbol characters.
- Right row indicator symbol character.
- Stop pattern.
- Trailing quiet zone.

NOTE: Truncated PDF417 barcodes reduce the stop pattern to a single bar.

Requirements

The symbol characters in a PDF417 barcode are 17-module wide, which consists of 4 bars and 4 spaces. Each bar and space can be from 1 to 6 modules in length. Each set of 929 patterns is a character set, but PDF417 only uses cluster numbers 0, 3 and 6. The symbol characters represent a “codeword” value determined by a number from 0 to 928.   Certain codeword values indicate a switch in compaction mode for data encoding. The data encodes in one of the following compaction modes:

- **Text Compaction** - Encodes alpha -numeric characters and punctuations. (Default)
- **Binary Compaction** - Encodes all 8-bit characters.
- **Numeric Compaction** - Encodes all 10 digits for the highest density.

Toolkit automatically encodes the best codeword for the smallest barcode possible, based on the value.

Remarks

A PDF417 barcode has the following adjustable parameters:

- Aspect Ratio.
- Width and Height.
- Error correction level.

Aspect Ratio: The number or rows and columns used change the barcodes aspect ratio. When adjusting the number of rows and columns, the number of symbol characters will remain constant for all rows in a barcode to ensure the barcode retains a rectangular shape. Each row uses character patterns from a single cluster. Any adjacent row must use different clusters in the sequence where the cluster number equals the row number minus 1. Each row must have a left row indicator and a right row indicator. Toolkit calculates the row indicators based on row number, total number of rows, columns, and the error correction level. You can adjust the aspect ratio with the PDFAspectRatio, PDFMaxCols and PDFMaxRows properties.

Width and Height: When printing a PDF417 barcode, it is important to ensure the resolution is as close to the nominal width of the barcode. You will need to scale the bar/space to the exact pixel pitch of the printer. For the best results, the printer resolution should be set higher than 200 dpi. At this resolution, scanners will not have trouble interpreting the difference between two bars with the same width but a different number of pixels.

If you are viewing the barcode on a screen before printing, there are some important considerations. Generally, the barcode image on the screen will be 72 DPI. In this case, it is important to ensure the width and height are multiples of the pixel width. For example, using a 72 DPI resolution, the width of a pixel is 1000 divided by 72, which equals 13.88 pixels. The barcode width and height should be multiples of 13.88 such as 13.88 mils, 27.76 mils or 55.52 mils.

You can adjust the width and height with the PDFModuleHeight and PDFModuleWidth properties.

Error Correction: PDF417 barcodes contain 2 to 512 error correction code words, which correspond to an error correction level. The error correction codeword is calculated based on the Reed Solomon techniques. The number of error correction code words is per level is defined as follows:

- **0** - 2 code words.
- **1** - 4 code words.
- **2** - 8 code words.
- **3** - 16 code words.
- **4** - 32 code words.
- **5** - 64 code words.
- **6** - 128 code words.
- **7** - 256 code words.
- **8** - 512 code words.
- **9** - Automatic error correction based on data encoded.

NOTE: You can specify the error correction level using the PDFSecurityLevel.

DataMatrix

Overview

DataMatrix is a two-dimensional, variable length symbology used for encoding large amounts of data. Each barcode contains a series of data cells arranged in a unique perimeter pattern. DataMatrix barcodes are very efficient, allowing for a proper read even with 60% of the data area damaged.

NOTE: DataMatrix barcodes can encode over 2000 characters, but it is difficult for most scanners to read more than 800. If you require more data storage than 800 characters, you should use PDF417.

Encoding

DataMatrix can encode all 128 Standard ASCII values using a three-digit ASCII code, preceded by the \ character. For example, the carrot symbol ("^") corresponds to ASCII code \094. For a complete list of ASCII values, refer to Appendix D: Barcode Tables.

Structure

DataMatrix does not allow for the random choosing for the combination of rows and columns. For a list of available sizes, refer to the DataMatrixTargetSizeID.

Requirements

Each DataMatrix barcode consists of modules that influence the overall size. The overall shape of the barcode may be rectangular, but the data cell of the DataMatrix symbol is always square. You can set the module size with the DataMatrixModuleSize property.

Data encodes using any combination of 6 different DataMatrix encoding schemes:

- **ASCII** - Invoked prior to any other encoding scheme, all other schemes return to ASCII.
- **C40** - Primarily encodes upper-case alphanumeric data.
- **Text** - Primarily encodes lower-case alphanumeric data.
- **Base256** - Encodes all byte values from 0 to 255.
- **x12** - Primarily encodes ANSI X12 EDI data set.
- **EDIFACT** - Encodes ASCII characters from 32 to 94.

Toolkit will apply the encoding scheme that produces the shortest codeword stream.

MaxiCode

A.K.A.

USS-MaxiCode

Overview

Introduced by UPS in 1992, MaxiCode is a medium capacity two-dimensional barcode designed for high-speed scanning application of package sorting and tracking. The barcode's design allows it to scan in any direction.

Encoding

MaxiCode can encode all 128 Standard ASCII values using a three-digit ASCII code, preceded by the \ character. For example, the carrot symbol ("^") corresponds to ASCII code \094. For a complete list of ASCII values, refer to Appendix D: Barcode Tables.

Structure

MaxiCode has the following structural constants:

- Fixed barcode height and width (1.11 in. x 1.054 in. nominal).
- Center "Bulls eye" Finder Pattern.
- Data Modules arranged in a hexagonal array.
- 6 Orientation Clusters.
- 4 Mode bits.

The structure expresses the following components:

- Primary Message.
- Secondary Message.

Primary Message: The primary message encodes a postal code, 3-digit country code and 3-digit class of service code

Secondary Message: The secondary message encodes the remaining data.

Requirements

MaxiCode barcodes have a maximum data capacity of 93 characters. Each barcode contains 884 hexagonal modules arranged in 33 rows with each row containing up to 33 modules.

MaxiCode barcodes have 6 different encoding modes. Two of the modes, 0 and 1, are considered obsolete. Modes 2 and 3 respectively have succeeded them.

- **0** - Primary message is a Structured Carrier Message with a numeric postal code. The Secondary message encodes up to 84 uppercase characters, numeric or punctuation.
- **1** - Primary message plus secondary message encode up to 93 uppercase characters, numeric or punctuation.
- **2** - Primary message is a Structured Carrier Message with a numeric postal code. The Secondary message encodes up to 84 uppercase characters, numeric or punctuation.
- **3** - Primary message is a Structured Carrier Message with an alphanumeric postal code. The Secondary message encodes up to 84 characters.

- **4** - Primary plus secondary message encode up to 93 “characters”.
- **5** - Primary plus secondary message encode up to 77 “characters” with extended error correction throughout.
- **6** - Primary plus secondary message encode up to 93 “characters” for reader configuration purposes.

The Primary message contains four mode bits to indicate the mode. Depending on the mode, the Primary message also encodes the structured carrier message in 56 data bits, which contains the information for packing and sorting.

In Mode 0 and 1, the Secondary message consists of four independent sub-messages for error correction. In these modes, the fourth sub-message is prone to misreading.

In Mode 2 through 6, the Secondary message consists of into 2 error-corrected sub-messages, fully interleaved between even and odd characters. This limits the possibility of partial reads.

NOTE: If you specify Mode 2 or 3, the MaxicodeClass, MaxicodeCountryCode and MaxiZipCode properties are required.

Supplemental Barcodes

A supplemental barcode is a secondary barcode appearing to the right of the main symbol. Typically, the supplemental barcode is shorter than the main symbol. Toolkit automatically calculates the height. The human readable text appears above the barcode.

There are two types of supplemental barcodes:

- 2-Digit.
- 5-Digit.

The supplemental barcodes encode specific data relevant to the barcode creator, but commonly appears on printed material. Supplemental barcodes are only valid for UPC-A, UPC-E, EAN/JAN-13, EAN/JAN-8 and Bookland symbologies. For details on these symbologies, refer to the specific symbology type in Appendix C: Symbologies.

2-Digit Supplemental Barcode

The 2-digit supplemental barcode compliments the product code included with certain symbologies. The product code remains constant for all items produced by a single manufacturer. To allow for versioning, the 2-digit supplement increments with each new item in a series.

Magazines and other periodicals are common uses for the 2-digit supplement. The barcode remains constant, ensuring the retail seller does not have to enter a new item into their database each shipment. Combined with the 2-digit supplement, the retailer knows that this is a new issue of the same magazine. Additional uses of the 2-digit supplement are usually internal, allowing for sales tracking of a specific version of a product or inventory control.

5-Digit Supplemental Barcode

The UCC designed the 5-digit supplemental barcode for indication of the currency or industry and the price for products. The first digit of the supplemental barcode indicates the currency or industry by a pre-assigned UCC digit. The remaining four digits indicate the price for the product.

The most common use for the 5-digit supplemental barcode is the retail book industry. On a book, the first digit indicates the country currency code. For example, 5 indicates the U.S. Dollar, 0 indicates the British Pound and 4 for the Canadian Dollar. The rest of the code indicates the price up to 99.99 currency units. If the 5-digit supplement indicated "55999", the price in U.S. Dollars would be $59.99.

NOTE: The UCC assigns the values and may change them according to need. For the most current information and values, refer to the UCC guidelines.

Appendix D: Barcode Tables

Certain symbologies encode meaning beyond the encoded value. These symbologies encode information such as numbering authorities, usage information, special characters, dates, dimensions and more. To encode this information using Toolkit, you must use present values. This section includes values for the following:

- EAN Number System.
- ASCII Table.
- Application Identifier Values.

NOTE: For usage requirements, refer to the barcode format information in Appendix C: Symbologies.

EAN Number System

The structure of modern commerce requires barcodes to provide increasingly more detailed information about the goods to which they are applied. The EAN/UCC numbering system has become the worldwide standard for supplying this information. Using the EAN number system with certain barcodes enables you to specify data such as location numbers, type of goods, tracking and local numbering authorities. The following chart provides the codes as they apply to Toolkit.

For a complete list of uses for the numbering system, visit the EAN/UCC website.

Value	Numbering Authority
001 – 139	UCC (U.S. & Canada).
200 – 299	In-store numbers.
300 – 379	GENCOD-EAN France.
380	BCCI (Bulgaria).
383	EAN Slovenia.
385	EAN Croatia.
387	EAN-BIH (Bosnia-Herzegovina).
400 – 440	CCG (Germany).
450 -459 & 490 – 499	Distribution Code Center (DCC) Japan.
460 – 469	UNICSCAN - EAN Russia (Russian Federation).

470	EAN Kyrgyzstan.
471	EAN Taiwan.
474	EAN Eesti (Estonia).
475	EAN Latvia.
476	EAN Azerbaijan.
477	EAN Lithuania.
478	EAN Uzbekistan.
479	EAN Sri Lanka.
480	PANC (Philippines).
481	EAN Belarus.
482	EAN Ukraine.
484	EAN Moldavia.
485	EAN Armenia.
486	EAN Georgia.
487	EAN Kazakhstan.
489	HKANA (Hong Kong).
500 – 509	e.centre (UK).
520	HELLCAN-EAN HELLAS (Greece).
528	EAN Lebanon.
529	EAN Cyprus.
531	EAN-MAC (FYR Macedonia).
535	EAN Malta.
539	EAN Ireland.
540 – 549	EAN Belgium.Luxembourg.

560	CODIPOR (Portugal).
569	EAN Iceland.
570 – 579	EAN Denmark.
590	EAN Poland.
594	EAN Romania.
599	EAN Hungary.
600 – 601	EAN South Africa.
608	EAN Bahrain.
609	EAN Mauritius.
611	EAN Maroc (Morocco).
613	EAN Algeria.
616	EAN Kenya.
619	Tunicode (Tunisia).
621	EAN Syria.
622	EAN Egypt.
624	EAN Libya.
625	EAN Jordan.
626	EAN Iran.
627	EAN Kuwait.
628	EAN Saudi Arabia.
629	EAN Emirates.
640 – 649	EAN Finland.
690 – 695	Article Numbering Center of China - ANCC (China).
700 – 709	EAN Norge (Norway).

729	Israeli Bar Code Association - EAN Israel.
730 – 739	EAN Sweden.
740	EAN Guatemala.
741	EAN El Salvador.
742	EAN Honduras.
743	EAN Nicaragua.
744	EAN Costa Rica.
745	EAN Panama.
746	EAN Republic of Dominica.
750	AMECE (Mexico).
759	EAN Venezuela.
760 – 769	EAN (Swiss).
770	IAC (Colombia).
773	EAN Uruguay.
775	EAN Peru.
777	EAN Bolivia.
779	EAN Argentina.
780	EAN Chile.
784	EAN Paraguay.
786	ECOP (Ecuador).
789 – 790	EAN Brazil.
800 – 839	INDICOD (Italy).
840 – 849	AECOC (Spain).
850	Camera de Comercio de la Republica de Cuba (Cuba).

858	EAN Slovakia.
859	EAN Czech.
860	EAN YU (Yugoslavia).
865	GS1 Mongolia.
867	EAN DPR Korea (North Korea).
869	Union of Chambers of Commerce of Turkey - UCCE (Turkey).
870 -879	EAN Nederland (Netherlands).
880	EAN Korea (South Korea).
884	GS1 Cambodia (Cambodia).
885	EAN Thailand.
888	SANC (Singapore).
890	EAN India.
893	EAN Vietnam.
899	EAN Indonesia.
900 – 919	EAN Austria.
930 – 939	EAN Australia.
940 – 949	EAN New Zealand.
955	EAN Malaysia.
958	EAN Macau.
977	Serial Publications (ISSN).
978	Books & Paperbacks (ISBN).
979	Books & Paperbacks (ISBN) & Printed Sheet Music (ISMN).
980	Refund receipts.
981 – 982	Common Currency Coupons.

990 – 999	Coupons.

NOTE: Unassigned prefixes are reserved for future use.

ASCII Table

Using ASCII code, you can encode all 128 standard ASCII characters in your barcode. To use the ASCII characters, you need to supply the following ASCII codes for the desired characters in your encoded value. You must include the "\" character in your encoded value.

NOTE: For additional usage requirements, refer to the barcode format information in Appendix C: Symbologies.

ASCII CODE	ASCII VALUE	ASCII CODE	ASCII VALUE	ASCII CODE	ASCII VALUE
\000	NUL	**\044**	‘	**\088**	X
\001	SOH	**\045**	-	**\089**	Y
\002	STX	**\046**	.	**\090**	Z
\003	ETX	**\047**	/	**\091**	[
\004	EOT	**\048**	0	**\092**	\
\005	ENQ	**\049**	1	**\093**	]
\006	ACK	**\050**	2	**\094**	^
\007	BEL	**\051**	3	**\095**	_
\008	BS	**\052**	4	**\096**	`
\009	HT	**\053**	5	**\097**	a
\010	LT	**\054**	6	**\098**	b
\011	VT	**\055**	7	**\099**	c
\012	FF	**\056**	8	**\100**	d
\013	CR	**\057**	9	**\101**	e
\014	SO	**\058**	:	**\102**	f
\015	SI	**\059**	;	**\103**	g
\016	DLE	**\060**	<	**\104**	h

\017	DC1	**\061**	=	**\105**	i
\018	DC2	**\062**	>	**\106**	j
\019	DC3	**\063**	?	**\107**	k
\020	DC4	**\064**	@	**\108**	l
\021	NAK	**\065**	A	**\109**	m
\022	SYN	**\066**	B	**\110**	n
\023	ETB	**\067**	C	**\111**	o
\024	CAN	**\068**	D	**\112**	p
\025	EM	**\069**	E	**\113**	q
\026	SUB	**\070**	F	**\114**	r
\027	ESC	**\071**	G	**\115**	s
\028	FS	**\072**	H	**\116**	t
\029	GS	**\073**	I	**\117**	u
\030	RS	**\074**	J	**\118**	v
\031	US	**\075**	K	**\119**	w
\032	SPACE	**\076**	L	**\120**	x
\033	!	**\077**	M	**\121**	y
\034	“	**\078**	N	**\122**	z
\035	#	**\079**	O	**\123**	{
\036	$	**\080**	P	**\124**	|
\037	%	**\081**	Q	**\125**	}
\038	&	**\082**	R	**\126**	~
\039	'	**\083**	S	**\127**	DEL
\040	(	**\084**	T		

\041	)	**\085**	U		
\042	*	**\086**	V		
\043	+	**\087**	W		

Application Identifier Values

Application Identifiers encode meaning for the data that immediately follows in the encoding. Commonly, this includes dates, size, items contained, and destinations. Some AI numbers include an "x" or "y" following the numerical data. These special characters indicate that you must append a single digit Signifier.

The Signifier uses the following scheme for x and y:

- **x** – You must indicate the length of the data field, according to the "Length of Data" column.
- **y** – You must indicate where to place the decimal point in the encoded information. For example, if you are specifying ten and a half feet, the Signifier is 2 for "10.50".

NOTE: For usage requirements, refer to the barcode format information in Appendix C: Symbologies.

AI	Description	Length of data
0	Serial Shipping Container Code (SSCC18).	18 digits
1	Shipping Container Code (SSCC-14).	14 digits
2	Number of containers.	14 digits
10	Batch Number.	1-20 alphanumeric
11	Production Date.	6 digits: YYMMDD
13	Packaging Date.	6 digits: YYMMDD
15	Sell by Date (Quality Con-trol).	6 digits: YYMMDD
17	Expiration Date.	6 digits: YYMMDD
20	Product Variant.	6 digits: YYMMDD
21	Serial Number.	1-20 alphanumeric
22	HIBCC Quantity, Date, Batch and Link.	1-29 alphanumeric
23x	Lot Number.	1-19 alphanumeric
240	Additional Product Information.	1-30 alphanumeric
250	Second Serial Number.	1-30 alphanumeric
30	Quantity Each.	-
310y	Product Net Weight in kg.	6 digits

311y	Product Length/ 1st Dimension, in meters.	6 digits
312y	Product Width/Diameter/ 2nd Dimension, in meters.	6 digits
313y	Product Depth/Thickness/3rd Dimension, in meters.	6 digits
314y	Product Area, in square meters.	6 digits
315y	Product Volume, in liters.	6 digits
316y	Product Volume, in cubic meters.	6 digits
320y	Product Net Weight, in pounds.	6 digits
321y	Product Length/ 1st Dimension, in inches.	6 digits
322y	Product Length/ 1st Dimension, in feet.	6 digits
323y	Product Length,/ 1st Dimension, in yards.	6 digits
324y	Product Width/Diameter/2nd Dimension, in inches.	6 digits
325y	Product Width/Diameter/2 Dimension, in feet.	6 digits
326y	Product Width/2nd Dimension, in yards.	6 digits
327y	Product Depth/Thickness/3rd Dimension, in inches.	6 digits
328y	Product Depth/Thickness/3rd Dimension, in feet.	6 digits
329y	Product Depth/Thickness/3rd Dimension, in yards.	6 digits
330y	Container Gross Weight (Kg).	6 digits
331y	Container Length/1st Dimension (Meters).	6 digits
332y	Container Width/Diameter/2nd Dimension (Meters).	6 digits
333y	Container Depth/Thickness/3rd Dimension (Meters).	6 digits
334y	Container Area (Square Meters).	6 digits
335y	Container Gross Volume (Liters).	6 digits
336y	Container Gross Volume (Cubic Meters).	6 digits
340y	Container Gross Weight (Pounds).	6 digits

341y	Container Length/1st Dimension, in inches.	6 digits
342y	Container Length/1st Dimension, in feet.	6 digits
343y	Container Length/1st Dimension, in yards.	6 digits
344y	Container Width/Diameter/2nd Dimension, in inches.	6 digits
345y	Container Width/Diameter/2nd Dimension, in feet.	6 digits
346y	Container Width/Diameter/2nd Dimension, in yards.	6 digits
347y	Container Depth/Thickness/Height/3rd Dimension, in inches.	6 digits
348y	Container Depth/Thickness/Height/3rd Dimension, in feet.	6 digits
349y	Container Depth/Thickness/Height/3rd Dimension, in yards.	6 digits
350y	Product Area (Square Inches).	6 digits
351y	Product Area (Square Feet).	6 digits
352y	Product Area (Square Yards).	6 digits
353y	Net Weight (Troy Ounces).	6 digits
354y	Product Volume (Quarts).	6 digits
355y	Product Volume (Gallons).	6 digits
356y	Container Gross Volume (Cubic Inches).	6 digits
360y	Container Gross Volume (Cubic Feet).	6 digits
361y	Product Volume (Cubic Inches).	6 digits
362y	Product Volume (Cubic Feet).	6 digits
363y	Product Volume (Cubic Yards).	6 digits
364y	Container Gross Volume (Cubic Inches).	6 digits
365y	Container Gross Volume (Cubic Feet).	6 digits
366y	Container Gross Volume (Cubic Yards).	6 digits
367y	Number of Units Contained.	6 digits

368y	Customer Purchase Order Number.	6 digits
369y	Ship To/Deliver To Location Code (EAN13 or DUNS code).	6 digits
37	Bill To/Invoice Location Code (EAN13 or DUNS code).	1-8 digits
400	Purchase From Location Code (EAN13 or DUNS code).	1-29 alphanumeric
410	Ship To/Deliver To Postal Code (Single Postal Authority).	13 digits
411	Ship To/Deliver To Postal Code (Multiple Postal Authorities).	13 digits
412	Roll Products - Width/Length/Core Diameter.	13 digits
420	Electronic Serial Number (ESN) for Cellular Phone.	1-9 alphanumeric
421	UPC/EAN Serial Identification.	4-12 alphanumeric
8001	Price per Unit of Measure.	14 digits
8002	Coupon Extended Code: Number System, Offer, End of Offer.	1-20 alphanumeric
8003	Coupon Extended Code: Number System proceeded by 0.	14 Digit UPC + 1-16 alphanumeric Serial number
8004	Mutually Agreed Between Trading Partners.	1-30 alphanumeric
8005	USPS services.	6 digits
8100	Internal Company Codes.	6 digits
8101	Internal Company Codes.	10 digits
8102	Internal Company Codes.	2 digits
90	Internal Company Codes.	1-30 alphanumeric
91	Internal Company Codes.	2-digit service code, 9-digit customer ID, 8-digit package ID plus 1 Mod10 check digit
92	Internal Company Codes.	1-30 alphanumeric
93	Internal Company Codes.	1-30 alphanumeric

94	Internal Company Codes.	1-30 alphanumeric
95	Internal Company Codes.	1-30 alphanumeric
96	Internal Company Codes.	1-30 alphanumeric
97	Internal Company Codes.	1-30 alphanumeric
98	Internal Company Codes.	1-30 alphanumeric
99	Internal Company Codes.	1-30 alphanumeric

Appendix E: Supported Image Types

activePDF Toolkit supports these image types:

Image Type	Description
AWD	At Work Document, Microsoft Fax.
BMI	Zoner Bitmap.
BMP	Microsoft Windows Bitmap, including compressed and HiColor.
BMP	OS/2 Bitmap.
BW	Black-and White images in SGI Image file format.
CAL	CALS Raster Type 1.
CDR	CorelDraw! 2.0 - 9.0, preview and imported bitmaps only.
CDT	CorelDraw! 2.0 -9.0 Template, preview and imported bitmaps only.
CEL	AutoDesk Animator Pro animation.
CEL	AutoDesk Animator still picture.
CLP	Microsoft Windows Clipboard file.
CMX	Corel Metafile Exchange 5.0 - 9.0, preview and imported bitmaps only.
CPT	Corel PhotoPaint 6.0.
CUR	Microsoft Windows cursors.
CUT	Dr. Halo/Dr. Genius Clipboard format.
DCX	Intel's multipage fax-format.
DIB	Microsoft Widows Device Independent Bitmap, including compressed.
FLC	Autodesk Animator Pro Animation.
FLI	AutoDesk Animator Animation.
HAM	Amiga Interchange File Format in hardware modes (HAM).

HMR	GeoTIFF, produce by SW by HMR Inc.
HRZ	Slow scan television.
ICN	Images from RIPTerm program.
ICO	Microsoft Windows Icons, including HiColor and TrueColor.
ICO	OS/2 Icons.
IFF	Amiga Interchange File Format.
IMG	GEM IMG.
IMG	IMG Software Set.
IMG	Vivid Raytracer.
JFF	JPEG File Format, including Progressive Mode.
JIF	JPEG File Interchange Format, including Progressive Mode.
JMX	Images from a Tetris game.
JPEG	JPEG, including Progressive Mode.
JPG	JPEG, including Progressive Mode.
LBM	Amiga Interchange File Format, Interleaved Bitmap.
MAC	MacPaint.
MIL	CALS Raster Type 1.
MSP	Microsoft Paint.
OFX	OLIFAX fax package.
PAN	SmoothMove Pan Viewer, preview only.
PAT	CorelDraw! 6.0 -9.0 patterns, preview only.
PBM	Portable Bitmap.
PC2	Degas EliteMedium Resolution.
PCD	Kodak PhotoCD - Base/16, Base/4 and Base only.

PCT	Macintosh PICT.
PCX	PC PaintBrush.
PGM	Portable GrayMap.
PIC	PC Paint/Pictor, including HiColor.
PIC	Dr. Halo/Dr. Genius.
PNG	Portable Network Graphics.
PNM	Portable Any Bitmap.
PPM	Portable PixelMap.
PSD	Adobe Photoshop 2.5 - 4.0, including CMYK.
PYX	Old Epson scanner format.
QFX	Quick Link II Fax file format.
RAS	Raster Sun Microsystems.
RGB	TrueColor (RGB) Images in SGI File Format.
RLE	Compressed Microsoft Windows BMP.
RLE	Compressed Images in SGI File Format.
RLE	Compressed Intergraph Raster images, bi-level only.
RLE	Utah Run Length Encoded.
SAM	Images in Text Documents from AmiPro.
SCx	ColorRIX.
SEP	CMYK Separated Images in TIFF 6.0 File Format.
SGI	SGI Image File Format.
ST	NeoPaint for DOS thumbnails.
STW	NeoPaint for Windows thumbnails.
SUN	Raster Sun Microsystems.

TGA	TrueVision Targa.
UDI	Ultimate Database Interface
WPG	WordPerfect Graphics.
ZBR	Zoner Zebra for Microsoft Windows 1.0 - 1.5, preview only.
ZMF	Zoner Metafile, preview only.

Appendix F: Supported Comment Colors

activePDF Toolkit supports the following colors when using the AddComment method:

Color Name	Code
Snow	#FFFAFA
GhostWhite	#F8F8FF
WhiteSmoke	#F5F5F5
Gainsboro	#DCDCDC
FloralWhite	#FFFAF0
OldLace	#FDF5E6
Linen	#FAF0E6
AntiqueWhite	#FAEBD7
PapayaWhip	#FFEFD5
BlanchedAlmond	#FFEBCD
Bisque	#FFE4C4
PeachPuff	#FFDAB9
NavajoWhite	#FFDEAD
Moccasin	#FFE4B5
Cornsilk	#FFF8DC
Ivory	#FFFFF0
LemonChiffon	#FFFACD
Seashell	#FFF5EE
Honeydew	#F0FFF0
MintCream	#F5FFFA

Azure	#F0FFFF
AliceBlue	#F0F8FF
lavender	#E6E6FA
LavenderBlush	#FFF0F5
MistyRose	#FFE4E1
White	#FFFFFF
Black	#000000
DarkSlateGray	#2F4F4F
DimGrey	#696969
SlateGrey	#708090
LightSlateGray	#778899
Grey	#BEBEBE
LightGray	#D3D3D3
MidnightBlue	#191970
NavyBlue	#000080
CornflowerBlue	#6495ED
DarkSlateBlue	#483D8B
SlateBlue	#6A5ACD
MediumSlateBlue	#7B68EE
LightSlateBlue	#8470FF
MediumBlue	#0000CD
RoyalBlue	#4169E1
Blue	#0000FF
DodgerBlue	#1E90FF

DeepSkyBlue	#00BFFF
SkyBlue	#87CEEB
LightSkyBlue	#87CEFA
SteelBlue	#4682B4
LightSteelBlue	#B0C4DE
LightBlue	#ADD8E6
PowderBlue	#B0E0E6
PaleTurquoise	#AFEEEE
DarkTurquoise	#00CED1
MediumTurquoise	#48D1CC
Turquoise	#40E0D0
Cyan	#00FFFF
LightCyan	#E0FFFF
CadetBlue	#5F9EA0
MediumAquamarine	#66CDAA
Aquamarine	#7FFFD4
DarkGreen	#006400
DarkOliveGreen	#556B2F
DarkSeaGreen	#8FBC8F
SeaGreen	#2E8B57
MediumSeaGreen	#3CB371
LightSeaGreen	#20B2AA
PaleGreen	#98FB98
SpringGreen	#00FF7F

LawnGreen	#7CFC00
Green	#00FF00
Chartreuse	#7FFF00
MedSpringGreen	#00FA9A
GreenYellow	#ADFF2F
LimeGreen	#32CD32
YellowGreen	#9ACD32
ForestGreen	#228B22
OliveDrab	#6B8E23
DarkKhaki	#BDB76B
PaleGoldenrod	#EEE8AA
LtGoldenrodYello	#FAFAD2
LightYellow	#FFFFE0
Yellow	#FFFF00
Gold	#FFD700
LightGoldenrod	#EEDD82
goldenrod	#DAA520
DarkGoldenrod	#B8860B
RosyBrown	#BC8F8F
IndianRed	#CD5C5C
SaddleBrown	#8B4513
Sienna	#A0522D
Peru	#CD853F
Burlywood	#DEB887

Beige	#F5F5DC
Wheat	#F5DEB3
SandyBrown	#F4A460
Tan	#D2B48C
Chocolate	#D2691E
Firebrick	#B22222
Brown	#A52A2A
DarkSalmon	#E9967A
Salmon	#FA8072
LightSalmon	#FFA07A
Orange	#FFA500
DarkOrange	#FF8C00
Coral	#FF7F50
LightCoral	#F08080
Tomato	#FF6347
OrangeRed	#FF4500
Red	#FF0000
HotPink	#FF69B4
DeepPink	#FF1493
Pink	#FFC0CB
LightPink	#FFB6C1
PaleVioletRed	#DB7093
Maroon	#B03060
MediumVioletRed	#C71585

VioletRed	#D02090
Magenta	#FF00FF
Violet	#EE82EE
Plum	#DDA0DD
Orchid	#DA70D6
MediumOrchid	#BA55D3
DarkOrchid	#9932CC
DarkViolet	#9400D3
BlueViolet	#8A2BE2
Purple	#A020F0
MediumPurple	#9370DB
Thistle	#D8BFD8
Snow1	#FFFAFA
Snow2	#EEE9E9
Snow3	#CDC9C9
Snow4	#8B8989
Seashell1	#FFF5EE
Seashell2	#EEE5DE
Seashell3	#CDC5BF
Seashell4	#8B8682
AntiqueWhite1	#FFEFDB
AntiqueWhite2	#EEDFCC
AntiqueWhite3	#CDC0B0
AntiqueWhite4	#8B8378

Bisque1	#FFE4C4
Bisque2	#EED5B7
Bisque3	#CDB79E
Bisque4	#8B7D6B
PeachPuff1	#FFDAB9
PeachPuff2	#EECBAD
PeachPuff3	#CDAF95
PeachPuff4	#8B7765
NavajoWhite1	#FFDEAD
NavajoWhite2	#EECFA1
NavajoWhite3	#CDB38B
NavajoWhite4	#8B795E
LemonChiffon1	#FFFACD
LemonChiffon2	#EEE9BF
LemonChiffon3	#CDC9A5
LemonChiffon4	#8B8970
Cornsilk1	#FFF8DC
Cornsilk2	#EEE8CD
Cornsilk3	#CDC8B1
Cornsilk4	#8B8878
Ivory1	#FFFFF0
Ivory2	#EEEEE0
Ivory3	#CDCDC1
Ivory4	#8B8B83

Honeydew1	#F0FFF0
Honeydew2	#E0EEE0
Honeydew3	#C1CDC1
Honeydew4	#838B83
LavenderBlush1	#FFF0F5
LavenderBlush2	#EEE0E5
LavenderBlush3	#CDC1C5
LavenderBlush4	#8B8386
MistyRose1	#FFE4E1
MistyRose2	#EED5D2
MistyRose3	#CDB7B5
MistyRose4	#8B7D7B
Azure1	#F0FFFF
Azure2	#E0EEEE
Azure3	#C1CDCD
Azure4	#838B8B
SlateBlue1	#836FFF
SlateBlue2	#7A67EE
SlateBlue3	#6959CD
SlateBlue4	#473C8B
RoyalBlue1	#4876FF
RoyalBlue2	#436EEE
RoyalBlue3	#3A5FCD
RoyalBlue4	#27408B

Blue1	#0000FF
Blue2	#0000EE
Blue3	#0000CD
Blue4	#00008B
DodgerBlue1	#1E90FF
DodgerBlue2	#1C86EE
DodgerBlue3	#1874CD
DodgerBlue4	#104E8B
SteelBlue1	#63B8FF
SteelBlue2	#5CACEE
SteelBlue3	#4F94CD
SteelBlue4	#36648B
DeepSkyBlue1	#00BFFF
DeepSkyBlue2	#00B2EE
DeepSkyBlue3	#009ACD
DeepSkyBlue4	#00688B
SkyBlue1	#87CEFF
SkyBlue2	#7EC0EE
SkyBlue3	#6CA6CD
SkyBlue4	#4A708B
LightSkyBlue1	#B0E2FF
LightSkyBlue2	#A4D3EE
LightSkyBlue3	#8DB6CD
LightSkyBlue4	#607B8B

SlateGray1	#C6E2FF
SlateGray2	#B9D3EE
SlateGray3	#9FB6CD
SlateGray4	#6C7B8B
LightSteelBlue1	#CAE1FF
LightSteelBlue2	#BCD2EE
LightSteelBlue3	#A2B5CD
LightSteelBlue4	#6E7B8B
LightBlue1	#BFEFFF
LightBlue2	#B2DFEE
LightBlue3	#9AC0CD
LightBlue4	#68838B
LightCyan1	#E0FFFF
LightCyan2	#D1EEEE
LightCyan3	#B4CDCD
LightCyan4	#7A8B8B
PaleTurquoise1	#BBFFFF
PaleTurquoise2	#AEEEEE
PaleTurquoise3	#96CDCD
PaleTurquoise4	#668B8B
CadetBlue1	#98F5FF
CadetBlue2	#8EE5EE
CadetBlue3	#7AC5CD
CadetBlue4	#53868B

Turquoise1	#00F5FF
Turquoise2	#00E5EE
Turquoise3	#00C5CD
Turquoise4	#00868B
Cyan1	#00FFFF
Cyan2	#00EEEE
Cyan3	#00CDCD
Cyan4	#008B8B
DarkSlateGray1	#97FFFF
DarkSlateGray2	#8DEEEE
DarkSlateGray3	#79CDCD
DarkSlateGray4	#528B8B
Aquamarine1	#7FFFD4
Aquamarine2	#76EEC6
Aquamarine3	#66CDAA
Aquamarine4	#458B74
DarkSeaGreen1	#C1FFC1
DarkSeaGreen2	#B4EEB4
DarkSeaGreen3	#9BCD9B
DarkSeaGreen4	#698B69
SeaGreen1	#54FF9F
SeaGreen2	#4EEE94
SeaGreen3	#43CD80
SeaGreen4	#2E8B57

PaleGreen1	#9AFF9A
PaleGreen2	#90EE90
PaleGreen3	#7CCD7C
PaleGreen4	#548B54
SpringGreen1	#00FF7F
SpringGreen2	#00EE76
SpringGreen3	#00CD66
SpringGreen4	#008B45
Green1	#00FF00
Green2	#00EE00
Green3	#00CD00
Green4	#008B00
Chartreuse1	#7FFF00
Chartreuse2	#76EE00
Chartreuse3	#66CD00
Chartreuse4	#458B00
OliveDrab1	#C0FF3E
OliveDrab2	#B3EE3A
OliveDrab3	#9ACD32
OliveDrab4	#698B22
DarkOliveGreen1	#CAFF70
DarkOliveGreen2	#BCEE68
DarkOliveGreen3	#A2CD5A
DarkOliveGreen4	#6E8B3D

Khaki1	#FFF68F
Khaki2	#EEE685
Khaki3	#CDC673
Khaki4	#8B864E
LightGoldenrod1	#FFEC8B
LightGoldenrod2	#EEDC82
LightGoldenrod3	#CDBE70
LightGoldenrod4	#8B814C
LightYellow1	#FFFFE0
LightYellow2	#EEEED1
LightYellow3	#CDCDB4
LightYellow4	#8B8B7A
Yellow1	#FFFF00
Yellow2	#EEEE00
Yellow3	#CDCD00
Yellow4	#8B8B00
Gold1	#FFD700
Gold2	#EEC900
Gold3	#CDAD00
Gold4	#8B7500
Goldenrod1	#FFC125
Goldenrod2	#EEB422
Goldenrod3	#CD9B1D
Goldenrod4	#8B6914

DarkGoldenrod1	#FFB90F
DarkGoldenrod2	#EEAD0E
DarkGoldenrod3	#CD950C
DarkGoldenrod4	#8B658B
RosyBrown1	#FFC1C1
RosyBrown2	#EEB4B4
RosyBrown3	#CD9B9B
RosyBrown4	#8B6969
IndianRed1	#FF6A6A
IndianRed2	#EE6363
IndianRed3	#CD5555
IndianRed4	#8B3A3A
Sienna1	#FF8247
Sienna2	#EE7942
Sienna3	#CD6839
Sienna4	#8B4726
Burlywood1	#FFD39B
Burlywood2	#EEC591
Burlywood3	#CDAA7D
Burlywood4	#8B7355
Wheat1	#FFE7BA
Wheat2	#EED8AE
Wheat3	#CDBA96
Wheat4	#8B7E66

Tan1	#FFA54F
Tan2	#EE9A49
Tan3	#CD853F
Tan4	#8B5A2B
Chocolate1	#FF7F24
Chocolate2	#EE7621
Chocolate3	#CD661D
Chocolate4	#8B4513
Firebrick1	#FF3030
Firebrick2	#EE2C2C
Firebrick3	#CD2626
Firebrick4	#8B1A1A
Brown1	#FF4040
Brown2	#EE3B3B
Brown3	#CD3333
Brown4	#8B2323
Salmon1	#FF8C69
Salmon2	#EE8262
Salmon3	#CD7054
Salmon4	#8B4C39
LightSalmon1	#FFA07A
LightSalmon2	#EE9572
LightSalmon3	#CD8162
LightSalmon4	#8B5742

Orange1	#FFA500
Orange2	#EE9A00
Orange3	#CD8500
Orange4	#8B5A00
DarkOrange1	#FF7F00
DarkOrange2	#EE7600
DarkOrange3	#CD6600
DarkOrange4	#8B4500
Coral1	#FF7256
Coral2	#EE6A50
Coral3	#CD5B45
Coral4	#8B3E2F
Tomato1	#FF6347
Tomato2	#EE5C42
Tomato3	#CD4F39
Tomato4	#8B3626
OrangeRed1	#FF4500
OrangeRed2	#EE4000
OrangeRed3	#CD3700
OrangeRed4	#8B2500
Red1	#FF0000
Red2	#EE0000
Red3	#CD0000
Red4	#8B0000

DeepPink1	#FF1493
DeepPink2	#EE1289
DeepPink3	#CD1076
DeepPink4	#8B0A50
HotPink1	#FF6EB4
HotPink2	#EE6AA7
HotPink3	#CD6090
HotPink4	#8B3A62
Pink1	#FFB5C5
Pink2	#EEA9B8
Pink3	#CD919E
Pink4	#8B636C
LightPink1	#FFAEB9
LightPink2	#EEA2AD
LightPink3	#CD8C95
LightPink4	#8B5F65
PaleVioletRed1	#FF82AB
PaleVioletRed2	#EE799F
PaleVioletRed3	#CD6889
PaleVioletRed4	#8B475D
Maroon1	#FF34B3
Maroon2	#EE30A7
Maroon3	#CD2990
Maroon4	#8B1C62

VioletRed1	#FF3E96
VioletRed2	#EE3A8C
VioletRed3	#CD3278
VioletRed4	#8B2252
Magenta1	#FF00FF
Magenta2	#EE00EE
Magenta3	#CD00CD
Magenta4	#8B008B
Orchid1	#FF83FA
Orchid2	#EE7AE9
Orchid3	#CD69C9
Orchid4	#8B4789
Plum1	#FFBBFF
Plum2	#EEAEEE
Plum3	#CD96CD
Plum4	#8B668B
MediumOrchid1	#E066FF
MediumOrchid2	#D15FEE
MediumOrchid3	#B452CD
MediumOrchid4	#7A378B
DarkOrchid1	#BF3EFF
DarkOrchid2	#B23AEE
DarkOrchid3	#9A32CD
DarkOrchid4	#68228B

Purple1	#9B30FF
Purple2	#912CEE
Purple3	#7D26CD
Purple4	#551A8B
MediumPurple1	#AB82FF
MediumPurple2	#9F79EE
MediumPurple3	#8968CD
MediumPurple4	#5D478B
Thistle1	#FFE1FF
Thistle2	#EED2EE
Thistle3	#CDB5CD
Thistle4	#8B7B8B
grey11	#1C1C1C
grey21	#363636
grey31	#4F4F4F
grey41	#696969
grey51	#828282
grey61	#9C9C9C
grey71	#B5B5B5
gray81	#CFCFCF
gray91	#E8E8E8
DarkGrey	#A9A9A9
DarkBlue	#00008B
DarkCyan	#008B8B

DarkMagenta	#8B008B
DarkRed	#8B0000
LightGreen	#90EE90