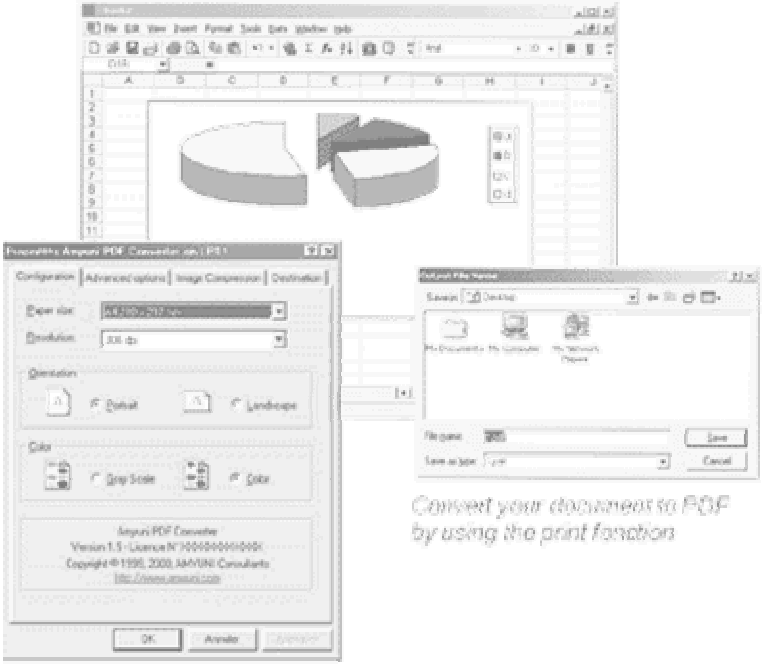
Amyuni PDF Converter

Version 2.05
Version 2.05 Professional
Updated March 15, 2002

User's Manual



Main property page

Amyuni Consultants – Amyuni Technologies
www.amyuni.com

Contents

Legal Information

Acknowledgments

Special thanks to:

Jean-loup Gailly (jloup@gzip.org)
Mark Adler (madler@alumni.caltech.edu)

for their work on the deflate compression.

This software is also based in part on the work of the Independent JPEG Group and on parts of the FreeType library.

What's new in version 2

- Generated PDF documents can now be *secured* to prevent them from being viewed, modified or even printed (Professional versions of the PDF Converter only).
- PDF documents can be optimized for downloading through the web, also known as PDF Linearization (Professional versions of the PDF Converter only).
- PDF documents can be emailed directly using any MAPI compliant mailing system as opposed to being saved on disk for later emailing.
- *Multi-national character sets* are now supported. These include eastern-european, middle-eastern and far-eastern languages.
- PDF *documents can be watermarked* using either a simple text string or another PDF file that can be used as background for generated PDF files. Watermarking can also be completely defined from the user interface.
- *JPEG compression level* can be adjusted to produce smaller files with reduced quality pictures or larger files with high-quality pictures.
- *Custom paper-sizes* are supported and can be pre-defined in the user interface.
- PDF *document properties* such as the Author can now be entered via the user interface.
- *Hyperlinks* can now be programmatically inserted in the PDF file while it is being created.
- *Improved programmer interface* (CDI ActiveX control) to give programmers the ability to post-process generated files.

For a complete version history, please refer to the Versions.txt file that comes with this product.

Installation

Before installing and/or using this product, please make sure you carefully read the copyright notice and agree on all its terms.

The install procedure described here applies to the standard version of the PDF Converter. For installing and using the developer version, please go to Appendix at the end of this document.

Automatic Installation Procedure

An automatic installation procedure has been created to easily install the PDF printer on your machine. This procedure is only available for Windows 95, 98, Me, NT, 2000 and later operating systems. For Windows 3.1, refer to the section on manual installation.

If you have received the product by e-mail, copy the file that you received in an empty directory, unzip the file and launch “install.exe”.

If you have received the product by diskette or on CD-ROM, you can launch “install.exe” directly from the diskette or CD-ROM.

The installation procedure automatically detects your operating system; copies needed files to your system directory and installs the PDF printer in your system.

“Install.exe” can be launched with the following switches:
/s or -s : Silent mode. No dialog box is displayed in this case. This is useful for automating the install procedure.
/p or -p : No Spooler. The default installation stops and restarts the spooler. This may not be allowed under some situations. If you are getting “Spooler Error” messages when installing the PDF Converter, then you should use this option.
/u or -u : Uninstall. This options removes the PDF printer and all associated files and registry entries from the system.

A specific printer name can also be specified by executing Install “My PDF Printer”. In this case the PDF printer will appear in the printers control panel as “My PDF Printer”.

A log file named “install.log” is generated in the source directory. If the install procedure is launched from a read-only media such as CD-ROM, the log file will be

generated in your system’s temporary directory. This file is required by the support staff to analyse problems that may occur during installation.

Changes from previous version

The installation tool for versions 1 and 1.5 copied a file named PDFINTF.DLL to the system directory. Version 2 now copies the file CDINTF.DLL and registers it in the system to be used as an ActiveX control from other applications.

Starting with version 2.02, the PDF printer is now attached to the LPT1:. The PDF: port is not used anymore.

Manual Installation Procedure

To manually install the PDF printer into your system, we recommend that you follow the instructions in your operating system's manual.

The following options should be chosen to have the PDF driver correctly installed:
Output port LPT1: for all Windows versions.
Spooler Off (or direct printing On)

Usage

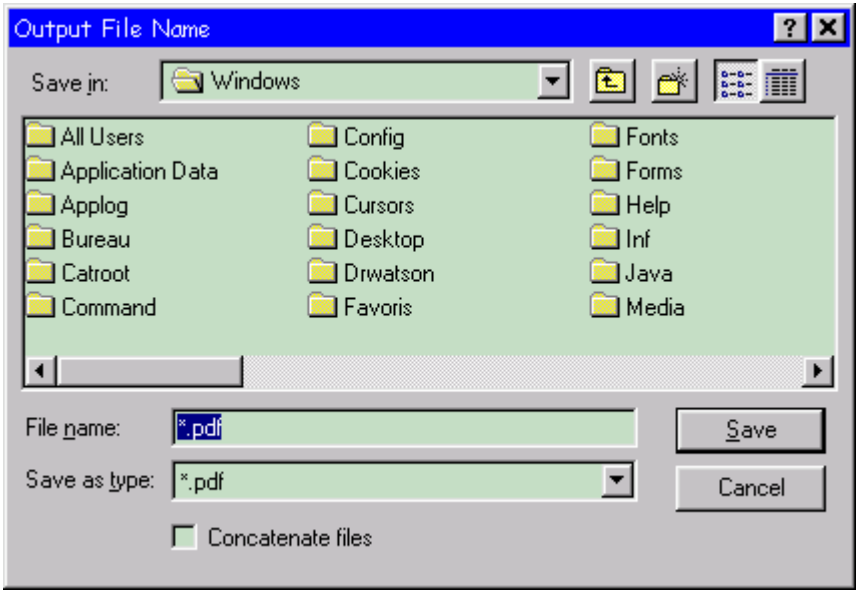
The PDF Converter works like any standard Windows® printer driver.

The main difference is that instead of directly printing to a printer, it generates a PDF 1.2 or 1.3 compatible file.

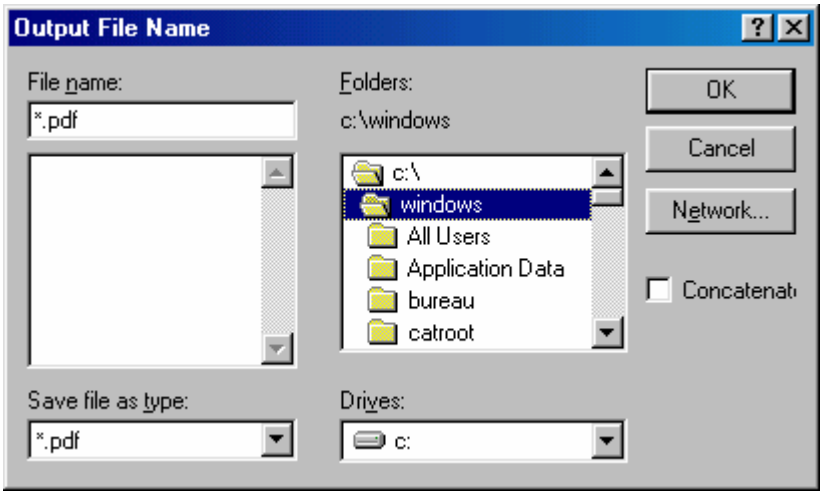
The PDF files generated by this driver can be viewed or printed to a physical printer by the Amyuni PDF Creator, the free Adobe® Acrobat® viewer versions 3 and upper, or by any other application capable of viewing standard PDF files.

When you use the “Print” menu of your application, you are prompted for the name of an output file:

Windows NT/2000/XP®



Windows® 3.1, 95, 98 or Me



Just enter any valid file name with .pdf extension. Other extensions can be used but the generated file will not be automatically recognized by the system as a PDF file.

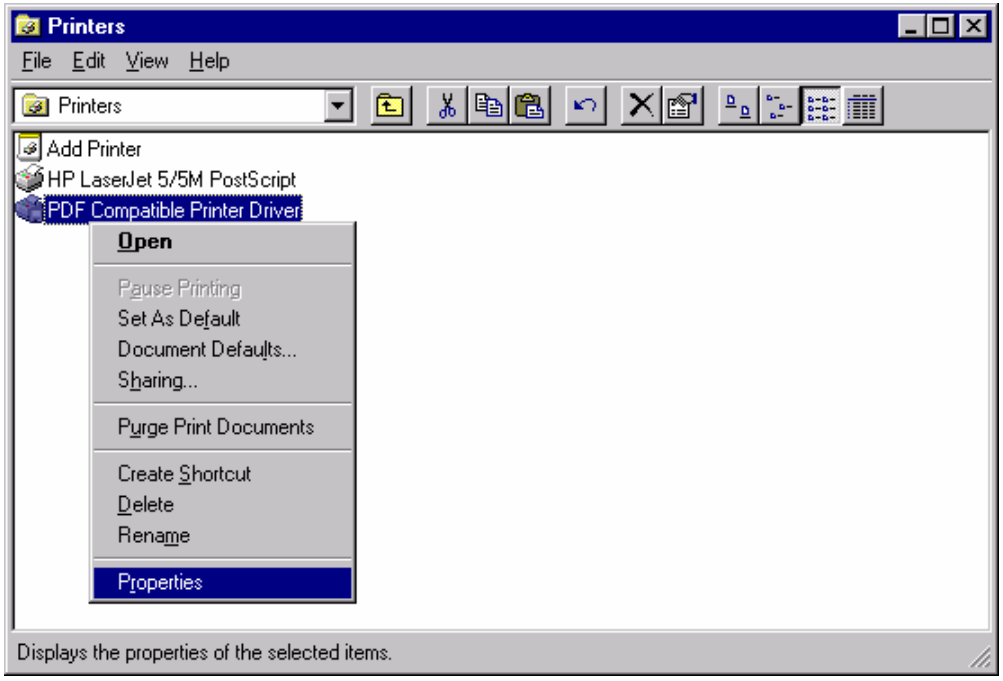
By default, the file is overwritten if it already exists. To append the document to an existing file instead of overwriting it, you can check the 'Concatenate' button.

Various options are available to automate the naming of the output file with or without user interaction. These options are described later in the manual under the **'Property page 4: Destination'** section.

Configuration

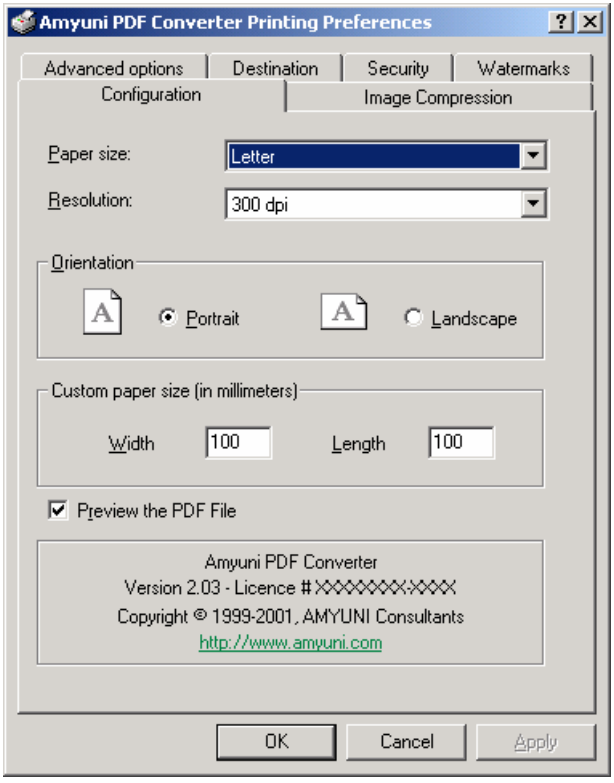
Information presented here only applies to the end-user version of the PDF Converter. The developer version does not have any user interface; all printer configuration should be done programmatically as explained in the samples and technical notes. The information presented here will however help the programmer have a better understanding of the various printer options made available by the PDF Converter.

The PDF printer properties are configurable using your system control panel or your application "Print" dialog box. The printer properties are available under "Document defaults" under Windows NT and "Printing Preferences" under Windows 2000.



The printer properties are divided into six property pages as follows.

Property Page 1: Configuration



Paper size

This is the default paper size for new documents. All standard paper sizes supported by Windows are also supported by the PDF Converter. This parameter is usually set by the client application. The value entered here is only the default for new documents. The default value for the Paper size is either A4 or Letter depending on the countries where the product is used.

Resolution

75, 150, 300, 600 or 1200 Dots per inch (DPI). This concerns mainly image resolution. Text precision can also be improved by increasing this value. The default value is 300 DPI.

Orientation

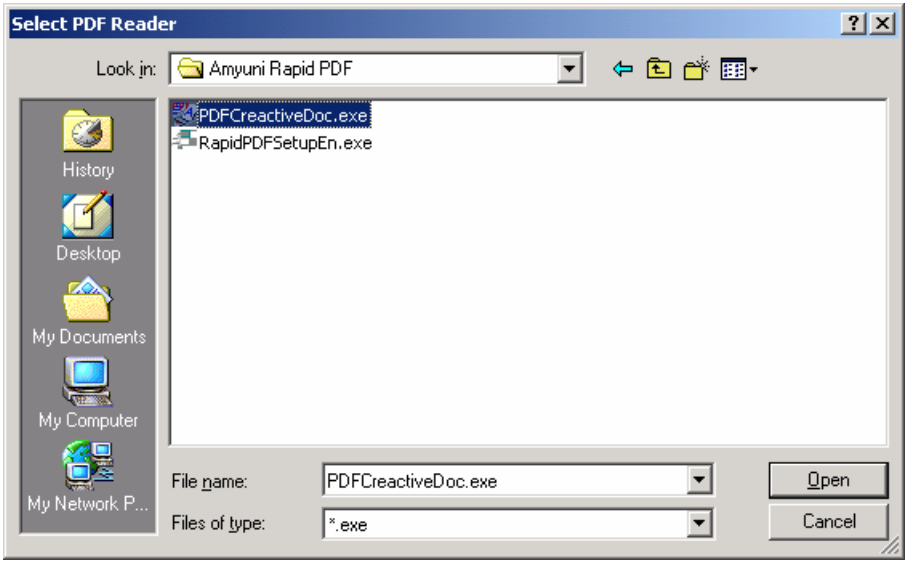
Portrait or Landscape. This is also a default value used for new documents only. Applications usually set this parameter internally. The default value is Portrait.

Custom paper size

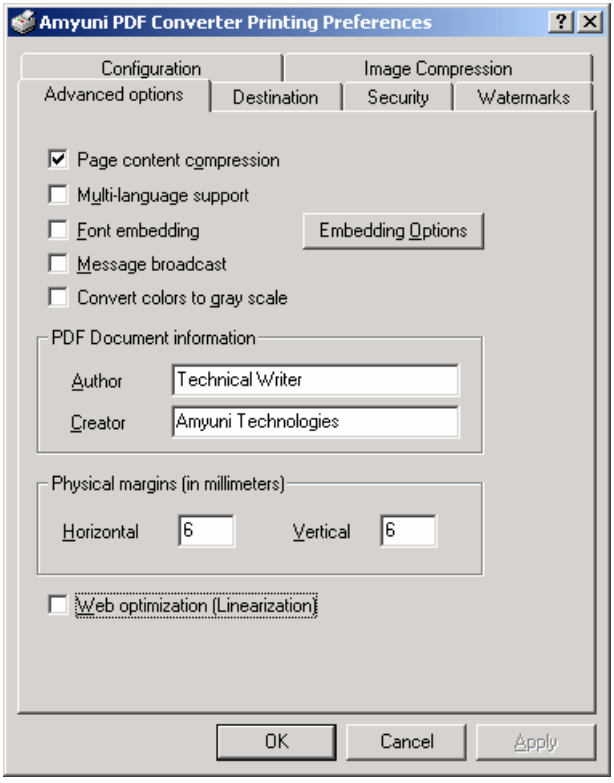
Users can define here a custom paper size that they often use for their documents. Custom paper size applies when 'Custom' is selected in the Paper size selection box. The default values for custom paper sizes are 100 mm width and length.

Preview the PDF File

This option enables you to view the PDF after it has been generated without having to launch the PDF reader and open the created PDF file. When you click on this checkbox, you will be prompted to choose the application needed to view the PDF file, this can be either Adobe Acrobat Reader®, Amyuni PDF Creator or any other PDF viewer.



Property Page 2: Advanced Options



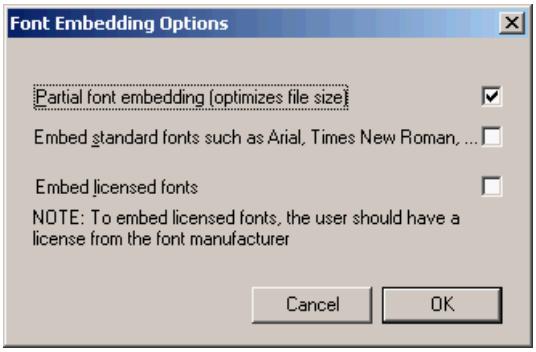
Page Content Compression

On or Off. This option enables page compression using the zlib/deflate compression algorithm. Removing this option can be useful for debugging or for applications that analyse the generated PDF file. The default value is On.

Multi-language support

When this option is checked, the PDF Converter will also convert characters that are not in the western-european and US character sets (ex: eastern european and far eastern characters) . The conversion in this case is however less efficient and files tend to be larger in size. Font embedding is recommended in this case to have some PDF viewers correctly display all the fonts.

Font embedding
On or Off. Setting this option to On will enable the driver to include all True Type® fonts used by the document in the output PDF file. This results in larger files but ensures consistent look of the document on any platform. The default value is Off.
Embedding Options
This button opens the following dialog-box to enable the user to more accurately define which fonts to embed:



- Partial font embedding is active by default and instructs the PDF Converter to embed only portions of the font file that are really used by the source-document.
- Embed standard fonts is disabled by default. Enabling it instructs the PDF Converter to embed Arial and Times New Roman fonts. These fonts are not embedded by default as they are provided with Acrobat® reader.
- Embed licensed fonts can be checked to embed fonts that require a license from their manufacturer to be embedded. This option is disabled by default and should be enabled only if the user is sure to have the license required to embed the fonts installed on his/her system.

Message Broadcasting
On or Off. When activated, messages are broadcasted for all running applications each time a document is printed to PDF format. The following events result in messages being broadcasted: Start of document, End of document, Start of page, End of page. Refer to appendix A for the format of these messages and how to intercept them. The default value is Off.

Convert colors to gray-scale
Using grey scale can decrease document size for archival or transmission purposes. The default value is Colour.

Vertical and horizontal margins

Most printers have a minimum physical margin below which they cannot generate any output. Applications use this value to prevent users from drawing outside the printable area. The default value is 6 millimetres.

Web optimization (Linearization)

This feature is available in the professional versions of the PDF Converter only

To view a PDF file located on a web server or another PC on the network, the user needs to download the whole file to the local PC before being able to open this file. When this option is activated, the PDF reader can download and view the document page by page without the need for downloading the full document. This results in the PDF document appearing on the client's screen in a nearly simultaneous manner.

Property Page 3: Image Compression

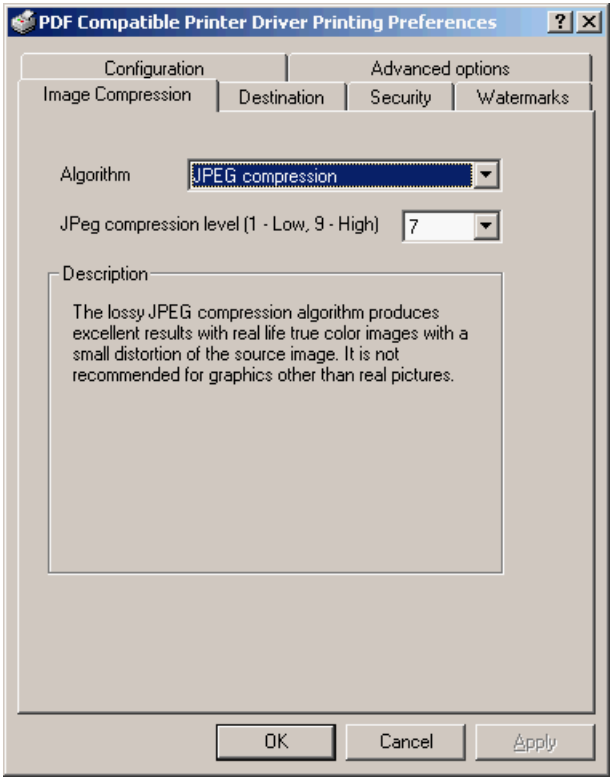


Image compression algorithm

Three image compression algorithms are implemented by the PDF Converter.

Default compression

The default image compression algorithm makes use of colour palettes to reduce the overall size of all images. This algorithm works well as long as the number of colours does not exceed 256.

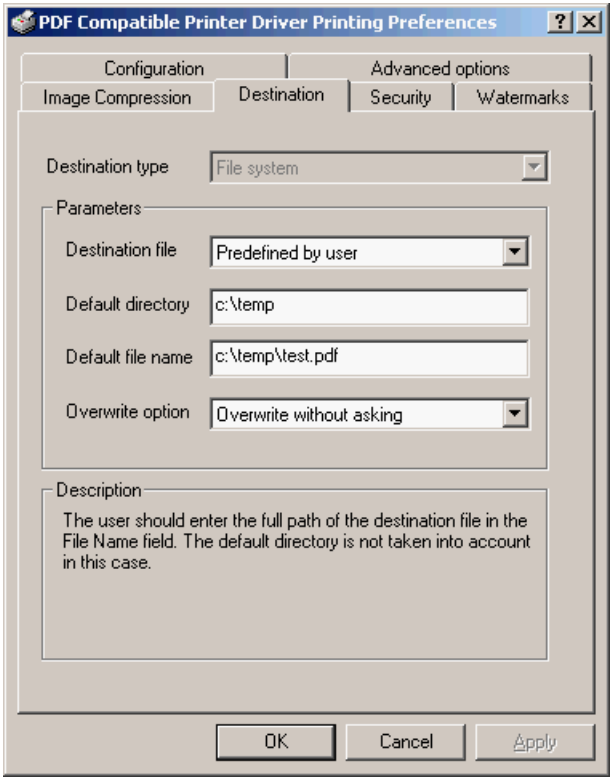
256-colour compression

The 256-colour compression algorithm computes the mostly used 256 colours in an image and replaces all other colour pixels by their closest match. This lossless compression algorithm works usually very well with most applications but is relatively slow on large images with a large number of colours.

JPEG compression
The lossy JPEG compression algorithm produces excellent results with real life true colour images with a small distortion of the source image. It is not recommended for graphics other than real pictures.

When JPEG compression is selected, the JPEG compression level selection box appears. JPEG compression levels vary from 1 to 9. Level 1 produces poor quality images with a very large compression ratio. Level 9 produces very good quality images with medium compression ratio. The default value is 7 which will produce compression levels and image quality suitable for most situations.

Property Page 4: Destination



Destination type

The PDF file can be either saved to the local or network drive (Destination type=FileSystem), or directly sent by email using the email software installed on the user's system (Destination type=Email system).

File System

The PDF file is saved to the local drive or on the network.

The destination file name can be set up in three different ways:

Prompt for file name

The user is prompted for an output file name each time a document is printed to the PDF printer. If a default directory is entered, the File Save dialog box is initialised with this value. The Default file name is not used.

Document title.pdf

The output file name is set automatically by the PDF Converter by concatenating the Default directory, the document title and .pdf extension. The document title depends on every client application. No additional user prompt is needed. The Default file name is not used.

Predefined by user
The output file name is predefined by the user in the “Default file name” field. The same file name is used for all generated PDF documents. No additional user prompt is needed. The Default directory is not used.

Overwrite option
This selection box defines the behaviour of the PDF printer when the output file already exists. The user can choose among one of the following options:

Overwrite without asking
The existing file is immediately overwritten by the new file without any confirmation. This is the default value that is compatible with previous versions of the product.

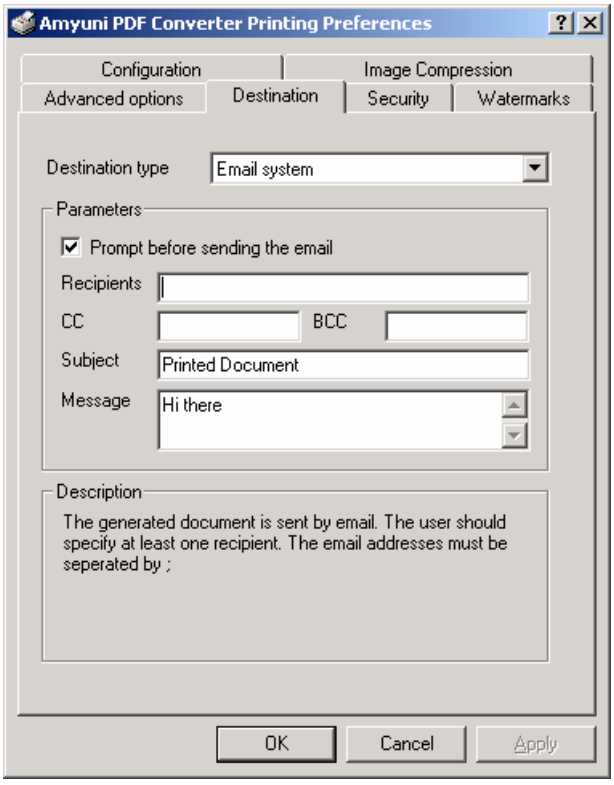
Confirm before overwriting
If the file already exists, the user is asked to confirm if he or she wishes to overwrite the file or cancel the operation.

Append to existing file
If the file already exists, the new document is automatically appended to it.

Add date and time to file name
The date and time are appended to the name of the file that the user gives through the user interface, or through the DefaultFileName function call. This is true whether the file existed or not. Ex: if the user or programmer enters test.pdf as file name, the resulting file will be named testddmmyyhhmmss.pdf, where ddmmyyhhmmss is the current date and time.

Add ID number to file name
An incremental ID number is appended to the name of the file that the user gives through the user interface, or through the DefaultFileName function call. This is true whether the file existed or not. Ex: if the user or programmer enters test.pdf as file name, the resulting file will be named testN.pdf, where N is an automatically generated ID number.

Email System



Prompt before sending the email
When this option is checked, the user is prompted with the various email parameters before the email is sent.

Recipients
The user can enter the list of recipients here. If more than one email address is entered, the addresses should be separated by a semi-colon (;).

BCC, CC
The user can enter the list of 'Carbon Copy' and 'Blind Carbon Copy' recipients here. If more than one email address is entered, the addresses should be separated by a semi-colon (;).

Subject, Message
The subject and message of the email that is to be sent with the PDF file attached.

Property Page 5: Watermarks

Watermark option

This can be set to either 'No Watermarks', 'Watermarks below page content' and 'Watermarks above page content'.

Watermark source

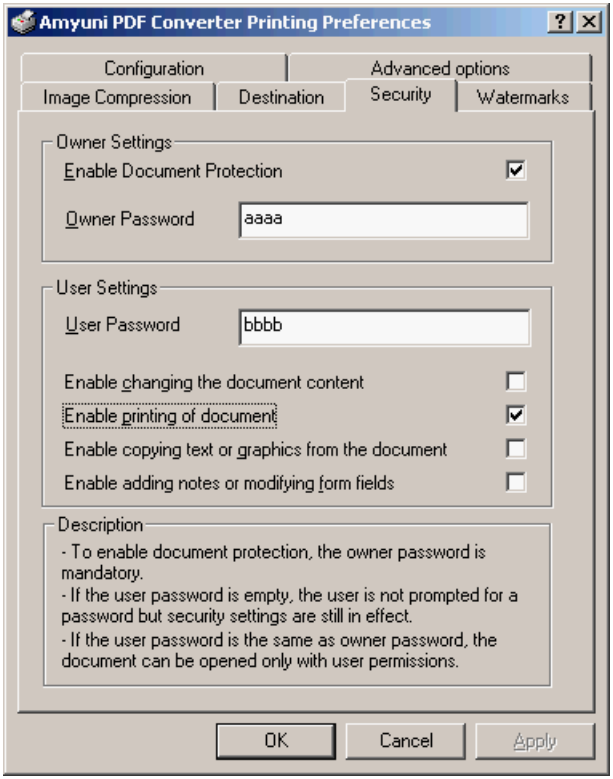
Watermarks can be added from two different sources:

- Simple text as entered in this dialog box
- Another PDF file that can be merged into the file that is currently being printed. This is useful to add a company or personal letter-head to all printed documents.

The other fields in this dialog-box define the various text properties used for drawing the watermark, or the file name for the other PDF file if that option was chosen.

Property Page 6: Security

This feature is available in the professional versions of the PDF Converter only



Enable document protection

When checked, this option enables documents to be encrypted and protected against illegal use or modification.

Owner and user passwords

Two passwords are associated to an encrypted PDF document. The owner password is for the author of the document, and the user password for the destinator or user of the document.

The owner password is mandatory and allows the author having this password to do any operation he/she wishes on this document, including modifying its security settings.

The user password is optional and can be one of the following:

- A blank password. In this case, the user is not prompted for a password when opening a document, but is restricted to the operations allowed by the author.
- The same password as the owner. In this case the user is prompted for a password and the author of the document will not be able to open this document as an owner to change its security settings.
- A password different from the owner. In this case, the user will not be able to open the document unless he/she enters a valid password. When a valid password is entered, the document can be viewed but its usage restricted to the operations allowed by the author.

User settings

- Enable changing the document content. When this option is cheked, the user is allowed to change the contents of the PDF document.
- Enable printing of document. The user cannot print the PDF document to any printer unless this option is checked.
- Enable copying text and graphics from the document. When this option is checked, the user can copy parts of the text of graphics from the PDF document.
- Enable addings notes or modifying form fields. The main body of the document cannot be changed but the user can add annotation or enter data in the form fields if there are any.

NOTE: These options are managed by the tool used to view the document and not by the PDF Converter. Once a valid password is entered, it is up to the viewer or editor to make sure that these security settings are respected.

Using the interface DLL (PDFIntf.dll)

This interface is not provided anymore with version 2 of the PDF Converter. The old version can still be used to maintain compatibility with older applications, but newer applications should always use the Common Driver Inteface (CDINTF.DLL) DLL to interface with the PDF Converter.

Appendix A: PDF Converter events format

When the message broadcast option is set, the PDF Converter broadcasts messages to all running applications each time one of the following events occur:

Start of document
Start of page
End of page
End of document

Reminder:
Each Windows messages is made of three parts:
A message Id. This is a 32-bit value.
A first parameter known as wParam. This is a 16-bit value under Windows 95/98 (Printer drivers under these systems are 16 bits), and a 32-bit value under Windows NT/2000.
A second parameter known as lParam. This is a 32-bit value.

The message Id generated by the PDF converter can be retrieved by calling the Windows API RegisterMessage function:

nPDFDriverMessage = RegisterWindowMessage(_T("PDF Driver Event")).

This is needed to intercept messages generated by the driver.

Associated with each printer event, is an event code which is given in the wParam parameter. The following event codes are defined by the Windows Device Driver Kit:

```
// printer driver events
#define DOCUMENTEVENT_STARTDOCPRE   5   // before StartDoc is handled
#define DOCUMENTEVENT_STARTDOCPOST 13   // after StartDoc is handled
#define DOCUMENTEVENT_STARTPAGE     6   // StartPage handled
#define DOCUMENTEVENT_ENDPAGE       7   // EndPage handled
#define DOCUMENTEVENT_ENDDOCPRE     8   // before EndDoc is handled
#define DOCUMENTEVENT_ENDDOCPOST   12   // after EndDoc is handled

// the following can be returned from the event handling function
#define DOCUMENTEVENT_SUCCESS     1
#define DOCUMENTEVENT_UNSUPPORTED 0
#define DOCUMENTEVENT_FAILURE     -1
```

Windows 95/98/Me

Event	wParam	lParam	
		LOWORD	HIWORD
Before StartDoc is handled	DOCUMENTEVENT_STARTDOCPRE	JobId	hDC
After StartDoc is handled	DOCUMENTEVENT_STARTDOCPOST	JobId	hDC
StartPage	DOCUMENTEVENT_STARTPAGE	JobId	hDC
EndPage	DOCUMENTEVENT_ENDPAGE	JobId	hDC
Before EndDoc is handled	DOCUMENTEVENT_ENDDOCPRE	JobId	hDC
After EndDoc is handled	DOCUMENTEVENT_ENDDOCPOST	JobId	hDC

Windows NT/2000/XP

Event	wParam		lParam
	LOWORD	HIWORD	
Before StartDoc is handled	DOCUMENTEVENT_STARTDOCPRE	JobId	hDC
After StartDoc is handled	DOCUMENTEVENT_STARTDOCPOST	JobId	hDC
StartPage	DOCUMENTEVENT_STARTPAGE	JobId	hDC
EndPage	DOCUMENTEVENT_ENDPAGE	JobId	hDC
Before EndDoc is handled	DOCUMENTEVENT_ENDDOCPRE	JobId	hDC
After EndDoc is handled	DOCUMENTEVENT_ENDDOCPOST	JobId	hDC

Sample message handling function

```
//////////////////////////////////////////////////////////////////////////
LONG CMessageTestDlg::OnPDFEvent( UINT wParam, LONG lParam )
//////////////////////////////////////////////////////////////////////////
{
// a PDF event has occured
//
       DWORD  jobID;
       HDC    hDC;
       WORD   eventId;

       // get event ID
       eventId = LOWORD( wParam );
```

```
        if ( m_bNT )
        {       // Windows NT
                jobId = HIWORD( wParam );
                hDC = lParam;
        }
        else
        {       // Windows 95/98
                jobId = LOWORD( lParam );
                hDC = HIWORD( lParam );
        }

        switch ( eventId )
        {
        case DOCUMENTEVENT_STARTDOCPRE:         // before StartDoc is handled
                m_log.AddString( "StartDoc called but not handled yet" );
                break;
        case DOCUMENTEVENT_STARTDOCPOST:        // after StartDoc is handled
                m_log.AddString( "StartDoc called and handled by the driver" );
                break;
        case DOCUMENTEVENT_STARTPAGE:           // StartPage handled
                m_log.AddString( "StartPage called and handled" );
                break;
        case DOCUMENTEVENT_ENDPAGE:             // EndPage handled
                m_log.AddString( "EndPage called and handled" );
                break;
        case DOCUMENTEVENT_ENDDOCPRE:           // before EndDoc is handled
                m_log.AddString( "EndDoc called but not handled yet" );
                break;
        case DOCUMENTEVENT_ENDDOCPOST:          // after EndDoc is handled
                m_log.AddString( "EndDoc called and handled" );
                break;
        default:
                return DOCUMENTEVENT_UNSUPPORTED;
        }
        return DOCUMENTEVENT_SUCCESS;
}
```

The drawback of this method of intercepting messages, is that the PDF driver will broadcast messages to all running applications which might result in slowing the whole system quite a bit. To avoid this message broadcast, the PDF driver can be programmed to send messages only to a specific application or window by calling te “SendMessageTo” function of the interface DLL (CDIntf.dll). This function takes as parameter the class of the window were the messages should be sent, ex:

```
        TCHAR  className[64];
        GetClassName( hWnd, className, sizeof(className) );
        SendMessagesTo( className );
```

Appendix B: Special operations available to programmers

Adding bookmarks to the PDF document

Version 1.5 of the PDF Converter, added the ability to easily include bookmarks in the resulting PDF document while it is being generated.

Bookmarks can be added programmatically, either by using Windows Escape function, or the SetBookmark CDI ("Common Driver Interface") function.

The following function which is implemented in the "Common Driver Interface" allows embedding of bookmarks:

```
////////////////////////////////////////////////////////////////////////////
long WINAPI SetBookmark( HDC hDC, long lParent, LPCCSTR szTitle )
////////////////////////////////////////////////////////////////////////////
{
// MAKE SURE COMPILER STRUCT ALIGNEMENT IS SET TO 1 OR 2.
//
       struct
       {
              long   lParent;
              char   szTitle[255];
       } Bookmark;

       Bookmark.lParent = lParent;
       lstrcpyn( Bookmark.szTitle, szTitle, 254 );

       return ExtEscape( hDC, 250, sizeof( DWORD ) + lstrlen( Bookmark.szTitle
                           ) + 1, (LPCSTR)&Bookmark, 0, NULL );
}
```

The bookmark will be inserted at the location where the last text drawing operation occurred. Example: if you draw text in the middle of page 3 of your document and call SetBookmark immediately after, the bookmark will point to the middle of page 3 of the PDF document.

PDF bookmarks are structured in a tree. The root has an id of 0. This is where the lParent parameter is used. Each time you insert a bookmark, the ExtEscape function returns the bookmark ID which can be used to insert other bookmarks as children.

VB sample for inserting bookmarks

```
Public pdf As New CDIntf.CDIntf

Private Sub Form_Load()

    ' initialize PDF printer and set it as default
    pdf.DriverInit "Amyuni PDF Converter"
    pdf.SetDefaultPrinter

    ' draw some text
    Printer.CurrentX = 200
    Printer.CurrentY = 400
    Printer.Print "Bookmark 1"
    ' set a bookmark on page 1
    pdf.SetBookmark Printer.hDC, 0, "Bookmark 1"

    ' go to next page
    Printer.NewPage

    ' draw some text and set a new bookmark
    Printer.CurrentX = 100
    Printer.CurrentY = 100
    Printer.Print "Bookmark 2"
    Parent = pdf.SetBookmark(Printer.hDC, 0, "Bookmark 2")

    ' set a bookmark as child of another bookmark
    Printer.CurrentX = 100
    Printer.CurrentY = 800
    Printer.Print "Submark 2-1"
    pdf.SetBookmark Printer.hDC, Parent, "Submark 2-1"

    Printer.EndDoc

    pdf.DriverEnd

End Sub
```

Adding watermarks across printed pages

This can be done through the common driver interface DLL (CDINTF or FLLINTF) in two steps:

1. Call the SetWatermark function to set various watermark properties
2. Add the EnableWatermarks (64) option to the FileNameOptions property (or SetFileNameOption call)

Here is a VB sample that prints a watermark for page 1 and removes it for page 2:

```
    Const NoPrompt As Integer = 1   ' do not prompt for file name
    Const UseFileName As Integer = 2    ' use file name set by
    SetDefaultFileName
```

```
Const EnableWatermarks = 64     ' enable watermarks on eaach page

Dim PDFPrinter As New CDIntf.CDIntf

' this attaches the CDIntf object to an existing printer
PDFPrinter.DriverInit "Amyuni PDF Converter"
' set default file name
'PDFPrinter.DefaultFileName = "c:\temp\test.pdf"
' set options
PDFPrinter.FileNameOptions = NoPrompt + UseFileName + EnableWatermarks

PDFPrinter.SetWatermark "This is a watermark", "Arial", 32, 450, &HFF,
100, -100, False

' print something on page 1
Printer.CurrentX = 200
Printer.CurrentY = 400
Printer.Print "Page 1"

' remove the watermark for page 2
Printer.NewPage
PDFPrinter.FileNameOptions = 0
' print something on page 2
Printer.CurrentX = 200
Printer.CurrentY = 400
Printer.Print "Page 2"
Printer.EndDoc

' reset all options before returning
PDFPrinter.FileNameOptions = 0
```

Parameters of the SetWatermark method:
SetWatermark(Watermark, FontName, FontSize, Orientation, Colour, HorzPos, VertPos, Foreground)

Return type	: long	
DLL call	: long SetWatermark(HANDLE hIntf, LPCSTR szWatermark, LPCSTR szFont, short fontSize, short Orientation, COLORREF colour, LONG xPos, LONG yPos, BOOL bForeground)	
FLL call	: SetWatermark(hIntf, szWatermark, szFont, fontSize, Orientation, colour, xPos, yPos, bForeground)	
Parameters	: hIntf	handle to the printer as returned by xxxDriverInit
	: Watermark	text to print on each page
	: FontName	font to be used to print text
	: FontSize	font size in 0.1 inch units
	: Orientation	watermark text orientation in 0.1 degree units
	: Colour	watermark colour in RGB format
	: HorzPos	horizontal position of text in 0.1 inch units

: VertPos vertical position of text in 0.1 inch units, this should be negative

: Foreground flag that indicates whether the watermark should be above or below the page content

Adding hyperlinks to a document while it is being generated

Version 1.58 of the PDF Converter, added the ability to easily include while the PDF document is being generated.

Hyperlinks can be added programmatically, either by using Windows Escape function, or the SetHyperlink CDI (“Common Driver Interface”) function.

The following function which is implemented in the “Common Driver Interface” allows embedding of hyperlinks:

```
////////////////////////////////////////////////////////////////////////
long WINAPI SetHyperLink( DWORD hDC, LPCSTR Destination )
////////////////////////////////////////////////////////////////////////
{
       return ExtEscape( (HDC)hDC, 252, lstrlen( Destination ) + 1,
                                        Destination, 0, NULL );
}
```

VB sample for inserting bookmarks and hyperlinks

```
Dim pdf As New CDIntf.CDIntf

' initialize PDF printer
pdf.DriverInit "Amyuni PDF Converter"

' set the output file
pdf.DefaultFileName = "c:\test.pdf"
pdf.FileNameOptions = NoPrompt + UseFileName

' draw some text
Printer.CurrentX = 200
Printer.CurrentY = 400
Printer.FontName = "Arial"
Printer.Print "Bookmark 1"
' set a bookmark on page 1
pdf.SetBookmark Printer.hDC, 0, "Bookmark 1"
' add a hyperlink to our web site
pdf.SetHyperLink Printer.hDC, "http://www.amyuni.com"

' go to next page
Printer.NewPage
```

```
    ' draw some text and set a new bookmark
    Printer.CurrentX = 100
    Printer.CurrentY = 100
    Printer.Print "Bookmark 2"
    Parent = pdf.SetBookmark(Printer.hDC, 0, "Bookmark 2")
    ' set a hyperlink to page 1
    pdf.SetHyperLink Printer.hDC, "#Bookmark 1"

    Printer.EndDoc
    pdf.DriverEnd

End Sub
```

The hyperlink will be on the last text or image that was drawn in the document. The hyperlink destination can either be an external URL such as “http://www.amyuni.com”, or a link to a bookmark inside the same document. In this case, the hyperlink destination should start by ‘#’ and point to a bookmark inside the document, ex: “#Bookmark 1”. Note that to be able to move backward and forward inside the document, the hyperlinks can be set before their destination bookmark.

Setting the new file generation options such as appending an ID or the date and time

Using the user interface, the user now has five new options for defining the output file name and the behaviour when the file already exists:

1. Overwrite existing file without confirmation. This is the default value.
2. Confirm before overwriting. If the file already exists, the user is asked to confirm if he or she wishes to overwrite the file or cancel the operation.
3. Append to existing file. If the file already exists, the new document is automatically appended to it.
4. Add date and time to file name. The date and time are appended to the name of the file that user gives through the user interface, or through the DefaultFileName function call. This is true whether the file existed or not. Ex: if the user or programmer enters test.pdf as file name, the resulting file will be named testddmmyyhhmmss.pdf, where ddmmyyhhmmss is the current date and time.
5. Add ID number to file name. An incremental ID number is appended to the name of the file that user gives through the user interface, or through the DefaultFileName function call. This is true whether the file existed or not. Ex: if the user or programmer enters test.pdf as file name, the resulting file will be named testN.pdf, where N is an automatically generated ID number.

These options can be programmatically set through the FileNameOptions function by adding these values to the file name option:

Overwrite existing file without confirmation	value = 0
Confirm before overwriting	value = 1000 (Hex)
Append to existing file	value = 2000 (Hex)
Add date and time to file name	value = 3000 (Hex)
Add ID number to file name	value = 4000 (Hex)

VB sample for appending ID numbers to the file name

```
    Const NoPrompt As Integer = 1       ' do not prompt for file name
    Const UseFileName As Integer = 2    ' use file name set by
SetDefaultFileName
    Const AppendID As Integer = &H4000  ' Add ID number to file name

    Dim pdf As New CDIntf.CDIntf

    ' initialize PDF printer
    pdf.DriverInit "Amyuni PDF Converter"

    ' set the output file
    pdf.DefaultFileName = "c:\test.pdf"
    pdf.FileNameOptions = NoPrompt + UseFileName + AppendID

    ' draw some text
    Printer.CurrentX = 200
    Printer.CurrentY = 400
    Printer.Print "My first PDF file"

    Printer.EndDoc
    pdf.DriverEnd

End Sub
```

Concatenating documents that had previously been generated with the PDF Converter

The function ConcatenateFiles has been added to the CDINTF DLL. This function enables you to concatenate PDF files after they have been generated, as opposed to when these files are being generated. By concatenation, we mean appending the pages of the second PDF file to the first one.

This function is declared as follows:

```
BOOL WINAPI ConcatenateFiles( LPCSTR file1, LPCSTR file2, LPCSTR file3 )
```

File1 and File2 are the input PDF files. File3 is the output PDF file and can be the same as File1 or File2.

This function returns FALSE if an error occurs, TRUE otherwise.

Concatenating multiple files can also be done using the Document ActiveX class that has been added to CDINTF.DLL. This class is described in the "Common Driver Interface" documentation.

Merging documents that had previously been generated with the PDF Converter

The function MergeFiles has been added to the CDINTF DLL. This function enables you to merge PDF files after they have been generated. By merging, we mean mixing the page contents of the second PDF file with the first one.

This function is declared as follows:

```
BOOL WINAPI MergeFiles(LPCSTR file1, LPCSTR file2, LPCSTR file3, BOOL bRepeat)
```

File1 and File2 are the input PDF files. File3 is the output PDF file and can be the same as File1 or File2.

This function returns FALSE if an error occurs, TRUE otherwise.

If the documents do not have the same number of pages, say file1 has N1 pages and file2 N2 pages where N1 < N2, then you can choose to:

- either merge file1 with the N1 pages of file2 and keep the remaining N2-N1 pages of file2 unchanged, in this case bRepeat should be set to False
- or merge the first block of N1 pages of file2 with the N1 pages of file1, merge the second block of N1 pages of file1 with the N1 pages of file1 and so on, in this case bRepeat should be set to True

Ex: if file1 contains the company's letterhead in PDF format as one page, file2 is a two page invoice in PDF format generated with the accounting package, you can call:

```
MergeFiles( file1.pdf, file2.pdf, file2.pdf, True )
```

to repeat the company's letterhead on all the invoice pages or

```
MergeFiles( file1.pdf, file2.pdf, file2.pdf, False )
```

to insert the company's letterhead on the first page only.

Appendix C: Using the developer version of the PDF Converter

Objective:
To help developers integrate the developer versions of the Amyuni Converters inside their applications.

The developer version of the Amyuni Converters is a special version of these products that can be distributed with the licensees' applications without paying any additional royalties to Amyuni Consultants.

By special version, we mean a version that:
- does not need to be pre-installed on the client system (no install utility)
- does not have any properties dialog box
- does not have any "File Save As" dialog box

The reference manual for the developer versions is the "Developer's manual" provided with each product. The user's manual also provided helps understand the overall operation of the products. The dialog boxes that are shown in this manual are not available in the developer version.

All printer configuration, file destination and options settings should be done programmatically by the main application.

The "Common Driver Interface" (CDINTF.DLL) provided with each of the mentioned products, is a DLL that contains both a standard DLL and an ActiveX interface to easily configure the printers.

Step by step procedure for using the Amyuni Converters, developer version

Step 1 – Copy all distributable files to the application's main directory
The application's main directory is usually where the executable file is located. The list of distributable files is given in the developer manual of each product.

Step 2 – Register the CDIntf ActiveX
If you are using the ActiveX interface, you need to register the ActiveX by calling: REGSVR32 CDINTF.DLL, from the directory where you copied the printer DLLs. You can use CDIntf through the DLL convention without the need for registering or creating ActiveXs.

Step 3 – Initialise the printer at start-up of your application
An application needs to initialise the printer when it is launched. To initialise the printer, you need to call PDFDriverInit, HTMLDriverInit, RTFDriverInit or EMFDriverInit depending on which product you are using. Each of these functions take as parameter the name of the printer that will be added to the printers folder as long as the application is running. Ex: PDFDriverInit("MyCompany PDF Export") will add a printer named "MyCompany PDF Export" to the list of printers available in the system. To avoid problems, use a name that is likely to be unique on the system where your application is running. Adding your company's name or your application's name or both to the printer name will help ensure a unique name.

Step 4 – Export to the format of your choice by printing from your application
When the user chooses the export function of your application to generate a PDF, HTML, RTF or EMF file, you need to set up the output file name using SetDefaultFileName, the file generation options SetFileNameOptions(NoPrompt + UseFileName + …) and print to the "MyCompany PDF Export" as you would do when printing to any other printer.

Step 5 – Restore the printer to its previous setting
When printing is over, you need to call SetFileNameOptions(0) to prevent other applications or users from overwriting the file that has just been generated from your application.

Step 6 – Remove the printer before exiting
Before exiting the application, call DriverEnd to remove the printer from the list of available printers.

Frequently encountered problems / asked questions

Q – *My application is deployed on NT/2000 systems where users do not have permission to add or remove a printer. How can I use these products under this configuration ?*

A – To use the Amyuni Converters in situations where users do not have enough rights to add or remove a printer, the printer needs to be installed in these systems by an administrator at the same time as the application is installed or at some later stage. To install the printer, you can use one of two methods:
In your application's install procedure, call PDFDriverInit("MyCompany PDF Export") or any other xxxDriverInit function. If you are using the ActiveX interface, the printer will be removed as soon as you exit the install application, to avoid this call DriverInit("MyCompany PDF Export") immediately after PDFDriverInit("MyCompany PDF Export").
Use the install utility provided with the product while logged in as an administrator. You can specify a printer name by using (install "MyCompany PDF Export"). You can also avoid the display of any dialog box by activating silent mode with the /s switch.
Your application's code does not need to be changed under this situation.

Q – *When frequently using one the Converter products, the DriverInit function fails or succeeds but printing fails. This happens randomly with no apparent reason*.

A – This problem usually occurs when xxxDriverInit / DriverEnd are called each time a document is converted. It is very important that these functions be called only once at application initialization (for xxxDriverInit) and once before exiting (for DriverEnd)

Q – *What is the difference between the DriverInit function and one of the xxxDriverInit functions ?*

A – The DriverInit function only attaches to an existing printer. If the specified printer does not exist, it is not created and the function fails. The xxxDriverInit functions attach to the printer if it already exists; otherwise they attempt to create it. In both cases, xxxDriverInit functions cause the printer to be removed when DriverEnd is called., whereas DriverInit leaves the printer in the system.

Appendix D: FileNameOptions values

The following options are supported by the PDF printer through the call to the SetFileNameOption function of the interface DLL (CDIntf.dll). For more details about calling this function to modify the printer options, please refer the the samples in this manual or the description of this function in the "Common Driver Interface.pdf" manual.

Option	Value (Hex)	Description
NoPrompt	1	Prevents the display of the file name dialog box
UseFileName	2	Use the file name set by SetDefaultFileName as the output file name
Concatenate	4	Concatenate with existing file instead of overwriting. This is useful only if the NoPrompt option is set
DisableCompression	8	Disable deflate (zip) compression of the page's content
EmbedFonts	10	Enable embedding of fonts used in the source document
BroadcastMessages	20	Send notification messages for PDF generation progress to all running application
PrintWatermark	40	Watermarks are printed on all printed pages
MultilingualSupport	80	Add supports for international character sets
EncryptDocument	100	Encrypt resulting document
FullEmbed	200	Embed full fonts as opposed to embedding the fonts partially
UseTcpIpServer	400	Reserved
SendByEmail	800	Send the document by email
ConfirmOverwrite	1000	If the file exists, confirm before overwriting
AppendExisting	2000	If the file exists, append to existing file
AddDateTime	3000	If the file exists, add date and time to file name
AddIdNumber	4000	If the file exists, add ID number to file name
LinearizeForWeb	8000	Activate web optimisation (Linearization) of PDF document
PostProcessing	10000	Post process file using specified application. The application's full path should be in the registry

JpegLevelLow	20000	Low quality JPeg compression of 24-bit images. This is equivalent to level 2 in the user interface.
JpegLevelMedium	40000	Medium quality JPeg compression of 24-bit images; this is equivalent to level 7 in the user interface
JpegLevelHigh	60000	High quality JPeg compression of 24-bit images; this is equivalent to level 9 in the user interface
Colors2GrayScale	80000	Replaces all colors by an equivalent gray scale value
ConvertHyperlinks	100000	Convert text beginning with http or www to a hyperlink
EmbedStandardFonts	200000	Embed standard fonts such as Arial, Times, ...
EmbedLicensedFonts	400000	Embed fonts requiring a license
256ColorCompression	800000	Activate 256 color compression

Appendix E: Technical notes available through our web site (www.amyuni.com)

A number of technical notes are available through our web site to help developers integrating the PDF Converter into their applications. The following technical notes are currently available:

Title	Description	Link
Multitasking the AMYUNI printer drivers	This note describes how to correctly use the AMYUNI printer drivers in multi-threaded environments such as web servers	www.amyuni.com/downloads/tn01.zip
Setting up printer properties using the DEVMODE structure	This note describes how to use the standard Windows DEVMODE structure to set general and specific printer properties	www.amyuni.com/downloads/tn02.htm
Using the Amyuni Printer Drivers with Visual FoxPro(r)	This note provides Visual FoxPro users with technical information to interface with the Amyuni series of printer drivers	www.amyuni.com/downloads/tn03.zip
Using the Amyuni Converters with Microsoft Access(r)	This note provides Microsoft Access users with technical information to interface with the Amyuni series of printer drivers	www.amyuni.com/downloads/tn05.zip
Using the Amyuni Converters with Sybase(r) PowerBuilder(r)	This note provides Sybase(r) PowerBuilder(r) users with technical information to interface with the Amyuni series of printer drivers	www.amyuni.com/downloads/tn06.zip
Using the Amyuni Converters with Borland(r) Delphi(r)	This note provides Borland(r) Delphi(r) users with technical information to interface with the Amyuni series of printer drivers	www.amyuni.com/downloads/tn07.zip

Technical Support

You can obtain technical support on our web site by visiting the following page:

http://www.amyuni.com/support.htm

Before contacting technical support, please take a look at the technical notes for answers to common questions.

Technical notes are freely available to registered and non-registered users from the following location:
http://www.amyuni.com/support.htm#technot

You need to be a registered user of our products to obtain technical support.

To be able to service you in an effective manner, include when possible the original document where the problem occurred and the PDF file generated by our driver.