PDF417 Fonts & Encoder 5
User Manual

PDF417 Fonts & Encoder 5 User Manual

Publication date: March 2019
Revision: 10881

Technical Support
Phone: (905) 752-0226
Fax: (905) 752-0355
Email: support@morovia.com
Web: http://www.morovia.com

For more information about Morovia products, visit http://www.morovia.com.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

PDF417 Fonts & Encoder 5 is the ultimate tool box to print PDF417 symbols. PDF417 is a multi-row, variable-length two dimensional symbology with high data capacity and error correction capability.

This release supports creating PDF417, Macro PDF417 and compact PDF417[1] symbols.

Both fonts and encoder functions are updated in this release, based on the feedback received from customers using previous version. Especially, this release is updated to accommodate the most recent standard - *ISO/IEC 15438: PDF417 bar code symbology specification*, second release, published in June 2006. Both 32-bit and 64-bit encoder DLLs and applications are included in the package. Windows 2000 or above is required to run run encoder GUI, or to call the encoder DLL. True type fonts can be used in other platforms such as OS/X or Linux.

This package includes the following contents:

- Four true type fonts targeting laser printers and bar code printers - `mrvpdf417n2.ttf`, `mrvpdf417n3.ttf`. `mrvpdf417n4.ttf`, `mrvpdf417n6.ttf`.
- Scalable PCL fonts to be used on PCL-compatible printers.
- The user manual, which you are reading on.
- PDF417 Encoder GUI, a GUI program to create barcode strings based on data entered. The program can export barcode images in a variety of formats, such as *PNG*, *EMF*, *SVG* and *EPS*.
- A Windows native *DLL* that allows you to add PDF417 printing to your own application.
- A Crystal Reports extension DLL that adds PDF417 Code printing functionality to Crystal Reports 9.0 and above.
- An ActiveX Control that can be inserted into Microsoft Office programs or integrated into your custom application.
- Examples demonstrating how to add PDF417 printing functionality to your applications in a variety of programming environments, such as Access, VB6, and .Net.

What's New

This release

- Crystal Reports UFL is modified to work with 64-bit Crystal runtime.
- Support of SSRS (SQL Server Reporting Service) is added.

Backward compatibility

The fonts and encoder included in this release are fully backward compatible with Version 4 release. However, the new UFL requires the function name change in existing reports.

Installing Morovia PDF417 Fonts & Encoder 5

To Install From a CD

[1] Previously referred as Truncated PDF417.

1. Insert the program CD into your CD drive. The setup starts automatically. Or if the auto-run feature isn't enabled on your system, click the Windows Start button and choose the Run command. Type **`D:\Setup.exe`** in the dialog box and click the OK button (Note that `D` represents the letter assigned to your CD-ROM drive. If your drive is assigned to a different letter, use it instead of `D`).
2. Follow the on-screen instructions.
3. Your will be prompted to enter the `License To` and `Registration Code`. The `License To` and `Registration Code` information are found on the back of the CD case.

To Install from Direct Download

1. Click the Download link to start the download.
2. When the browser prompts, do one of the following: A. To run setup immediately, click `Open` or `Run This Program from Its Current Location`. B. If you decide to run the setup at a later time, click `Save` or `Save This Program to Disk`.
3. If you chose `Save This Program to Disk` in Step 2, locate the file where you saved it, and double click on it.
4. Follow the setup instructions.
5. Your will be prompted to enter the `License To`/ `Registration Code`. The `License To` and `Registration Code` information can be found in the email we send to you after order completes.

Limitations of Trial Version

A trial version is provided on our web site that can be downloaded freely. In the trial version, whenever an encoder function is called, a reminder dialog appears on the screen. Barcodes created by the trial version have extra “DEMO” added at the end. To remove the limitation, purchase the full version.

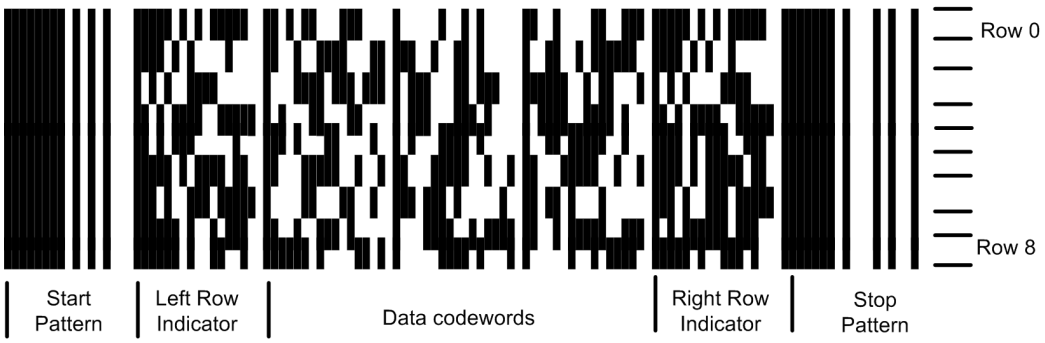
Chapter 2. Overview

This chapter briefly reviews the symbologies (a.k.a. barcode formats) supported by PDF417 Fontware. PDF417 Fontware & Writer SDK supports normal PDF417, truncated PDF417 and Macro PDF417.

What is PDF417?

PDF417 is a multi-row, variable-length symbology with high data capacity and error-correction capability. PDF417 offers some unique features which make it the widely used 2D symbology. A *PDF417* symbol can be read by linear scanners, laser scanners or two-dimensional scanners. PDF417 is capable of encoding more than 1100 bytes, 1800 text characters or 2710 digits. Large data files can be encoded into a series of linked PDF417 symbols using a standard methodology referred to as *Macro PDF417*.

Figure 2.1. Anatomy of a PDF417 symbol



A PDF417 symbol is composed of several key elements. These elements are module, codeword, start pattern, stop pattern, row indicator, row and column. Each key element is explained below.

module

A *module* is the smallest unit of a PDF417 symbol. The individual row height in a symbol is equal to the module height. The width of a module is commonly referred as *X dimension*. The ratio of the module height vs. width is called *y-height*. For better decodability, y-height should be no less than 3.0.

All other elements, such as start pattern, stop pattern and codewords, are composed of modules.

start pattern

A unique pattern of light and dark elements which indicates the leftmost part of a PDF417 symbol. All the rows start with the same pattern, with the result that the entire leftmost part of the symbol appears as a set of vertical bars and spaces.

codeword

A codeword contains four bars and four spaces, totaling 17 module widths and 1 module height. Each codeword starts with a bar and ends with a space. PDF417 encodes user data into codewords. Each codeword represent an integer value from 0 and 928.

stop pattern

A unique pattern of light and dark elements which indicates the rightmost part of a PDF417 symbol. Same as start pattern, all rows of PDF417 symbols share the same stop pattern.

row indicators

The two code words adjacent to the start or stop pattern in a row are *row indicators*. They encode information about the structure of the PDF417 symbol in terms of the row identification, total number of rows and columns and the error correction level.

row

A set of elements made up of a start pattern, row indicators, data codewords and a stop pattern. A PDF417 symbol must have at least 3 rows and no more than 90.

column

A column refers to the set of codewords vertically. It does not include start pattern, stop pattern and row indicators. A PDF41 symbol may have 1 to 30 columns.

In Figure 2.1, "Anatomy of a PDF417 symbol", there are 9 rows and 3 columns.

Creating PDF417 symbols can be thought as two distinct processes: *encoding* and *rendering*. Your application first calls a encoder function to convert the data into an integral representation. During this step you specify the parameters desired, such as number of rows columns, or aspect ratio. After encoding succeeded, you can one of rendering API to render the barcode in a variety of graphics formats supported, such as Windows metafile or PNG. If you are using fonts, you retrieve *barcode string* and format the barcode string with one of the fonts to turn them into a barcode.

Encoding Parameters

The minimum encoding parameters required are *data encoded* and *security level*. These two parameters decide the minimum number of codewords required. Once the number of minimum codewords is known, the program determines the size of the barcode based on other sizing parameters, such as number of rows and aspect ratio.

Input Format

PDF417 is capable of encoding all 256 values (bytes). Except the NUL character, which serves as the terminator for C language, you can enter all these characters as is. However, in some situations you may want to use the escape method, because your environment does not support entering control characters, or you are entering symbology specific features. For example:

- To specify advanced PDF417 features such as ECI and Macro PDF.
- To enter NUL character (ASCII code 0) and other control characters.
- To enter extended characters with pure ASCII.

The escape method described here is called *tilde method*. When creating PDF417 barcodes you can use the following tilde codes:

~dnnn

When `nnn` corresponds to a numeric value between 0 and `255`, the tilde code sequence represents a character with value equal to `nnn`. For example, `~d032` represents a space character.

~~

Represents a tilde (~) character.

~2

Indicates that a MacroPDF417 control block follows.

~7

Indicates the start of a ECI block. This escape sequence must be followed by exact 6 digits, which corresponds to the ECI value.

~X

Represents a character value from 0 to 26. Replace the X like in the following example ~@ means character ASCII 0, ~A means character 1, ~B means character 2, ~C means character 3 ...

For more information on entering ECI and Macro PDF417 block, see Appendix A, *Input Format for ECI and Macro PDF417*.

Security Level

PDF417 allows users selection of security level. While the default algorithm[1] is appropriate to most applications, you may want to increase the amount of error correction. Higher error correction level increases the size of the barcode, but is more resistant to various errors, such as printing error and symbol damage during the transport.

Internally, each symbol uses one of eight levels of error correction. Increasing the error correction by one level doubles the number of error correction codewords. In turn, it roughly doubles the amount of damage that the symbol can tolerate.

In our software, you can specify 9 to indicate that you want to use the default algorithm.

Since a PDF417 symbol must be rectangular, when the total number of data and security codewords does not make up a rectangle, padding codewords are added to fill the spaces. The padding codewords just fill the space, and do not increase the error resistance of the symbol[2] For this reason, our software consider the security level you requested as *minimum*, not the *exact* requirement. That is, if the symbol size required can't satisfy the security level requirement, the software reports an error. However, if the security level can be increased within the size specified, it selects a higher security level. The principle here is to reduce the number of useless padding codewords and increase the ECC codewords without increasing the overall symbol size. This arrangement makes the symbols produced more robust.

Sizing Parameters

After the minimum number of codewords are determined, the encoder selects a size that can fit these codewords. The encoder is quite flexible here: you can specify the exact number of rows and columns, or you can leave all the choices to the encoder.

The size of a printed PDF417 symbol is calculated as follows:

```
symbolwidth = (numCols*17+69)*moduleWidth
  symbolHeight = numRows * yHeight * moduleWidth
```

Specifying Exact Number of Rows and Columns

If you want all your symbols have identical sizes, you may specify the number of rows and columns to the ones required. In this scenario, you need to ensure that the size specified can hold the largest data possible, otherwise the encoder will report an error.

Specifying Either Number of Rows or Columns

Sometimes it is easier to specify one parameter and allow another parameter to grow if needed. This is useful when you occasionally have data that requires larger sizes.

For example, most of your data will fit a 11X7 rectangle. However, sometimes your data will require 14X7 rectangle. You can specify 7 as the number of columns, and leave the selection of rows to the encoder.

If another parameter, *AspectRatio* is not specified, the encoder will choose a combination that produces the minimum number of codewords. Otherwise, it selects a combination that produces a symbol with its aspect ratio closest match to the one specified.

---

[1]The algorithm is described in *Annex E, User selection of error correction level*, ISO/IEC 15438:2006.

[2]Worse, it might give you a false impression that your barcode is more error-resistant because it is larger.

Specifying Neither Number of Rows nor Columns

This scenario is not recommended, however supported by the encoder. When neither number of rows nor the number of columns is specified, the encoder will choose (1)if AspectRatio is not specified, the combination that requires the minimum number of codewords. (2) The combination that produces a symbol whose aspect ratio is the closest match to the value specified.

Because of the nature of PDF417 symbol, if you do not specify Aspect Ratio, the resulted barcode is normally a short and wide one, as it is always efficient to expand horizontally than vertically. If you do not like this behavior, set Aspect Ratio accordingly.

Impact of Aspect Ratio

Apsect Ratio does not affect the decision when you specify both the number of rows and columns. when you only specify one of them, or not at all, it kicks in. The aspect ratio of a symbol is the ratio of its overall height vs. its width. For example, if your aesthetics suggest that the symbol looks better when the width is approximately 3 times its height, the aspect ratio is 0.33.

The value of aspect ratio is considered *recommended* unlike other parameters. Other parameters entered are either considered *minimum* (such as security level) or *exact* (such as number of rows). It is not possible to achieve the exact value. It enables the encoder to compare all possible row and column combinations and select one that produces the closest match.

In order for the encoder to calculate the aspect ratio of the symbol, the encoder has to know another parameter, *yHeight*, which is the height of a module vs. its width. Because this parameter only affects Aspect Ratio, it impacts only when Aspect Ratio is considered during the selection process. If you specify both rows and columns, or aspect ratio is 0, it does not have any effect.

yHeight

The yHeight is the height of the module vs. it width. For better decodability, yHeight should be greater than 3.0. After number of rows and columns are determined, yHeight affects the final symbol size.

If you are using font-based approach, note that yHeight is fixed per font. For example, `MRV PDF417N3` has a fixed yHeight of 3.1.

Module Width

The module width is the width of the module. It is an important parameter that affect overall symbol quality. We will explain its selection in the next section.

Compact PDF417

Compact PDF417 was referred as *Truncated PDF417* in previous standard. Compact PDF417 may be used where space is a primary concern and the symbol damage is unlikely, such as in an office environment. It removes the right row indicator, and reduces the stop pattern into a single bar.

Figure 2.2. Example Compact PDF417 symbol

Module Width

The module width, sometimes also referred as *X dimension*, is the smallest length in a PDF417 symbol. It must be printed correctly otherwise you will end up with a low quality, or even worse, an unreadable barcode.

Through the API provided by our software, you can create either vector graphic files or raster graphic files. Barcode string via font-based approach is a form of vector graphic. Regardless the graphic format to use, eventually images are converted into individual pixels on a printer. Therefore, the resolution of the printer has quite an impact on the dimension planning

On 600-dpi printers, a pixel is approximately 1.5 mils in width and height. This size is quite small compare to the resolution of a barcode scanner (> 5mils). Obviously loss or addition of a printer pixel does not affect the result, as the scanner is unable to find out. However, when you print on low resolution printers, such as thermal printers or fax machines, the selection of the lengths must be planned carefully. A typical thermal printer's resolution is 203 dpi, which means that the width of a printer pixel is about 5 mils, large enough to affect the printing quality.

when working on lower resolution printer, the module width should be set to a value that produces integral times of pixels on target printers. Note that images may undergo transform and its dimensions may alter during the process from that they are created to that they are finally rendered on the printer. A typical scenario is that an image is created but scaled up or down at a later time. The barcode image may look prefect to human eyes, but renders poorly on the printer.

When rendering vector graphic images, our encoder also asks for the target printer resolution parameter, to ensure that the lengths will render consistently on the target printer. When rendering raster graphic images, the encoder asks for the number of pixels per X dimension. Special care should be paid to avoid loss or addition of pixels during the rendering process.

When you are printing on a low resolution printer, such as a 203-dpi thermal printer, or 300-dpi ink-jet printer, make sure that the X dimension is integral times of the pixel size. Follow the steps below to determine the best X dimension for your printer:

1. First multiply nominal X dimension (in inches) by the printer resolution. Assuming that the nominal X dimension is 15 mils and you are printing on a 300-dpi printer. The result will be 0.015*300 = 4.5.
2. Round this value to the closet integer. In this case, the round integer is either 4 or 5. We use 4.
3. Divide the result by the printer resolution to get the actual X dimension. In our case, it is 4/300 = 0.013 inch, or 13 mils.
4. 13 mils is the optimal X dimension for this printer. From Table 2.1, “List of PDF417 fonts”, we know that `MRV PDF417 N3` produces PDF417 barcodes at 13 mils at 12 points. If we go with font-base solution, we use `MRV PDF417 N3` at 12 points.

Working with PDF417 Fontware & Writer SDK

Previous PDF417 fontware support creating PDF417 barcodes using font-based approach only. This version supports several methods creating barcodes. You should become familiar with each approach, and choose the one best fit for your working environment.

Font-Based

Creating barcodes using fonts involves two distinct processes: *encoding* and *rendering*. Rendering is to choose the appropriate font and font size and format the encoding results. Encoding is to convert the data into a special string, a.k.a. *Barcode String*, which becomes a barcode after being formatted with an appropriate font.

Warning PDF417 fonts are not intended to apply on the text encoded directly. You must generate the barcode string first, and format the string with the font to create a valid PDF417 barcode.

The Font-based approach is easy to understand, and in some occasions the only way to add barcode printing functionality, such as in Crystal Reports.

The drawing below illustrates how you create a PDF417 symbol using font-based approach. You first create the barcode string (which is an array of text lines). Format the barcode string with `MRV PDF417 N3` font you get the barcode.

```
FFFFFFFF0F0F0F000FED6D7973EBB15540FDF583B11EAB26050FFE5E1F1D8B773320FFFFFFF0F000F0F00F
FFFFFFFF0F0F0F000FDF5D2FAC56338840F99874FBE24EAB280FD694BFF577A51110FFFFFFF0F000F0F00F
EEEEEEEE0E0E0E000E2E028C6AA8886600E660600AEA04C4AA0E2C208E6200EC8880EEEEEEE0E000E0E00E
```

Encoding

To create a valid barcode, you need to call an encoder to get a special string, and format this string with our PDF417 font. To get this string, you need to call an encoder - you can run PDF417 Encoder GUI, and obtain the result from this GUI program. Or if you need to bulk generate the results, using the programming interface exposed from the encoder DLL.

PDF417 Encoder GUI

This GUI program allows you to quickly create barcode string and transfer it to other programs. You are able to enter the data encoded, and create the barcode on the fly. For more information, see Chapter 3, *Using Encoder GUI*.

MoroviaPDF417FontEncoder.dll

This DLL is a standard Windows DLL that can be called by many programming environments, including Microsoft Office, C++, FoxPro and .Net. Most programming environments support Windows DLL directly. We provide a VBA module and a C# class that wrap around the DLL functionality.

cruflPDF417V5.dll

This DLL is specifically designed for Crystal Reports. For more information on using the software with Crystal Reports, see Chapter 7, *Adding PDF417 Symbols to Crystal Reports*.

Font Characteristics

In this package four fonts are provided, as below:

Table 2.1. List of PDF417 fonts

Filename	Typeface	y-height	X-dim at 12 points
mrvpdf417n2.ttf	MRV PDF417 N2	yHeight: 2.1	20 mils
mrvpdf417n3.ttf	MRV PDF417 N3	yHeight: 3.1	13.35 mils
mrvpdf417n4.ttf	MRV PDF417 N4	yHeight 4.2	10 mils

Filename	Typeface	y-height	X-dim at 12 points
mrvpdf417n6.ttf	MRV PDF417 N6	yHeight 6.2	6.67 mils

Note that y-height is fixed per font. Generally speaking, font `MRV PDF417 N3` is recommended to use. `MRV PDF417 N2` should only be used when you have complete control on the full process, i.e. a closed system.

Using font you change the barcode size by increasing or decreasing font size. Font scales linearly on both directions.

Advantages and Disadvantages

Fonts are supported by many applications. It is often the most easiest and straightforward method to add barcode printing function to a document, report or custom program. The barcode string can be stored and transferred to another platform. Troubleshooting is straightforward by examining text strings.

On the other side, font-based approach relies on rendering software to turn the drawing command into pixel at the time of print. Some software may not rasterize fonts properly. PDF417 fonts require the line height properly calculated to avoid line gaps (must be 0), which is not always the case. Some software allow you to adjust the line gap, but many do not provide such an option.

Not all sizes will produce optimal size barcodes, especially when printed on low resolution printers. See the chapter followed on notes working on low resolution printers, such as a fax machine.

Font-based approach also relies on the PDF417 fonts, which provide limited choices of y-height as y-height in each font is fixed.

In case that you can't get the font work on an application, you have two other options - standard image files and ActiveX control.

Image File Based

You can export barcodes as standard image files to be consumed in other word processing and imaging programs. The PDF417 Encoder GUI supports exporting images in five formats via a GUI interface. The Encoder DLL and PDF417 ActiveX Control also provides programming interfaces to export images in those formats:

- PNG (Portable Network Graphics)
- BMP (Windows Bitmap File)
- EMF (Ehanced Meta File)
- SVG (Scalable Vector Graphics)
- EPS (Encapsulated PostScript)

Special attention should be paid when the target printer has low resolution, such as a thermal printer or fax machine. When generating vector graphics files, the software allows you to specify target dpi which you should set to the one that matches your printer. It is generally not recommended to scale the image down or up when working with low-resolution printers.

PDF417 Font Encoder DLL provides a method to retrieve a Windows enhanced metafile (EMF) handle that can be used to print to printers. The PDF417 ActiveX control provides a similar feature through its `Picture` property.

ActiveX Control

For environments that support ActiveX Controls (also called OLE controls), you can insert the PDF417 ActiveX Control into documents, spreadsheets, or invokes the control at the background in your program. With ActiveX Control you are able to change the barcode image by editing its properties.

Using ActiveX Control in Microsoft Office programs is straightforward. Programming knowledge is required in order to integrate the control with your custom programs.

Solution Quick Reference

Morovia PDF417 Fontware & Writer SDK 4.0 was conceived as a Swiss knife for adding PDF417 barcodes to a wide range of programming environments and applications. Use the following list to quickly locate the solution for your environment.

C++ Application

Both font-based and ActiveX-based solutions are supported in C++. ActiveX control can be invoked at background.

Example C++ code are included in this release. See the files under CSample folder in the program directory. The `PDF417EncodeC.cpp` contains code that dynamically loads the dll and calls the functions. The `PDF417EncodeC2.cpp` uses VC++ `pragrma` to link to the import library. If you have Visual C++ installed, you can run `build.bat` on command prompt to build the executable.

Microsoft Word

It is recommended to insert the barcode using ActiveX control so that you can change the barcode easily at later time. You can also invoke GUI encoder and paste the results in either RTF or EMF formats into Microsoft Word.

Microsoft Excel

You can use PDF417 ActiveX control in Excel and link data to be encoded with a cell. Refer to the section called "Excel (ActiveX Control)" for more information.

Mail Merge

If you currently store data in a database (typically Excel spreadsheet), and need to print labels with PDF417 barcodes, you can use Word Mail Merge to pull the data and print them out. the section called "Word Mail Merge" explains how to do so.

Microsoft Access

Both font and ActiveX control can be used in Microsoft Access reports. See the section called "Access (ActiveX Control)" for using PDF417 control, and the section called "Access (PDF417 Font)" for font-based solution.

Crystal Reports

PDF417 barcodes can be easily added to Crystal Reports 9.0 and above using UFL. See Chapter 7, *Adding PDF417 Symbols to Crystal Reports* for more information.

SQL Server Reporting Service (SSRS)

This product includes a SSRS custom assembly that adds PDF417 barcode printing functionality to Reporting Service. See Chapter 8, *Adding PDF417 Symbols to SQL Server Reporting Service* for details.

Visual Basic 6

Both font-based and ActiveX approaches can be used in Visual Basic 6. A sample project is included in the software.

.Net (All versions)

Both Font encoder DLL and ActiveX control can be used in .Net programming environment with the help of interoperability. The software includes examples written in C# on both approaches. Note: the software includes only 32-bit ActiveX control by default. Using font encoder dll is encourage because both 32-bit and 64-bit versions are included in the software.

Java

Java applications can access Encoder functions through Java Native Access(JNA)[3].

[3] https://jna.dev.java.net/

Delphi, Foxpro, Powerbuilder and more

Although we do not officially support those programming environments due to lack of software in our lab. However, based on our support experience, our software fully support these programming languages (both DLL and ActiveX control).

Currently, the software requires Windows 2000 or above to function. We are able to compile the encoder DLL on other platforms such as Mac OS/X and Linux. If you need a solution to work on those platforms, write to `support@morovia.com` for more information.

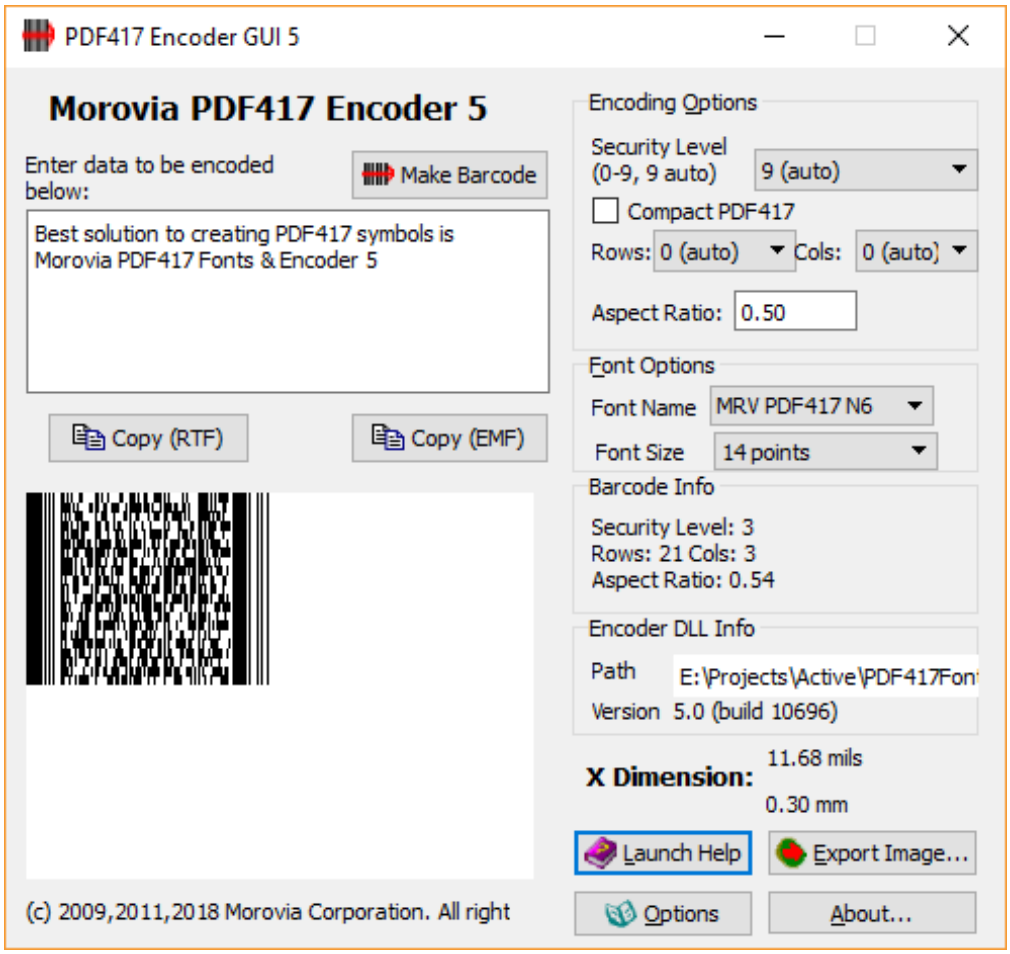
Could not locate your specific needs? You are welcome to email our support engineer at `support@morovia.com`. Before you do, you might want to visit our site first to see if your concern has been addressed. make sure that you concern has been visited by us. Search Morovia Knowledge Base[4] and Community Forums[5] first. It is possible that your question has been addressed.

[4] http://mdn.morovia.com/kb/
[5] http://forums.morovia.com/

Chapter 3. Using Encoder GUI

PDF417 Encoder GUI is a GUI program that runs on Windows 2000 and above. It provides a fast way to create PDF417 barcodes on the fly. From the program you can easily create barcodes either in barcode string format, or images files, and transfer the to a graphics designer or word processor.

Figure 3.1. PDF417 Encoder GUI Options



Make Barcode

Click on this button to refresh the barcode display.

Data to Be Encoded

This is the place to enter the data to be encoded. Multiple-line text is supported. To enter the next line, enter **Ctrl+Enter**.

Copy (RTF)

Place the barcode string in the clipboard, so that you can subsequently paste them into another application such as Microsoft Word.

PDF417 Encoder GUI places both text and RTF formats into the clipboard. In applications that are capable of processing *RTF* (Rich Text Format), you will see a barcode immediately after pressing the paste button.

Otherwise, you will see the text string instead. If this is the case, highlight the whole string and format with "MRV PDF417" font.

Copy (EMF)

Place the barcode image in the clipboard, so that you can subsequently paste it into another application such as Microsoft Word. The EMF created does not use font, so that you do not need to copy the font file when you view document from another computer.

Note that X dimension in the resulted barcode is determined by the Module Size in the Options dialog, as the EMF format does not use the font.

Barcode Info

This section displays the actual attributes of the barcode created, including error correction level, number of rows, number of columns and aspect ratio of the barcode.

Launch Help

Launch the HTML Help of this program.

Options

Pops up the Options dialog where you can specify additional option for exporting barcode images. See the section called "Program Options" for more information.

Export Image

Click on this button to export the barcode into standard image formats such as EMF, SVG, EPS and PNG.

About

Click on this button to pop up the About dialog.

X Dimension

This section displays the X-dimension of the barcode, in units of mils and mm.

Encoder DLL Info

This section displays the absolute path and the file version of the encoder DLL. This is useful if you have multiple DLLs in the computer.

Font Options

Select the font and size from this section.

Encoding Options

Specify the barcode options: security level (0-9, 9 automatic). Compact PDF417, Rows, Columns and Aspect Ratio.

Note If you leave all options to the default, the program selects a size that occupies the least space, often a short and wide stripe. To create a tall and narrow symbol, change Aspect Ratio to a value less than 1.0.

Exporting Images

PDF417 Encoder GUI supports exporting into several standard image formats. The images generated do not contain references to the fonts so that you can move around them without requiring the software to be installed.

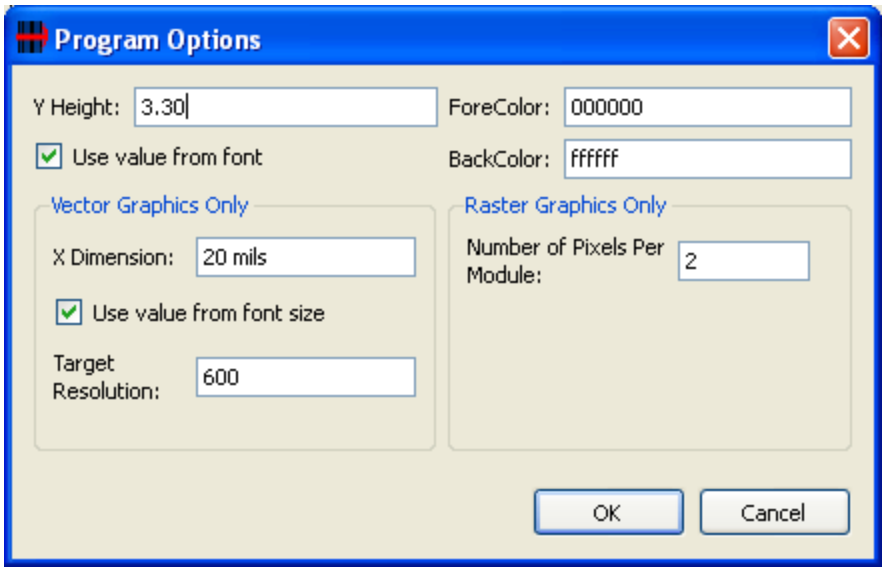
- *EMF*. This is the default vector graphics format used on Windows operating system.
- *PNG*. PNG is a bitmap image format widely supported by imaging software. It is the recommended image format for web graphics.

- *BMP*. BMP is the default bitmap format used on Windows platform.
- *EPS*. This format is widely used in graphics design industry. Recommended to use in conjunction with Adobe software such as Adobe Illustrator.
- *SVG*. SVG is an emerging vector graphics format. FireFox 3 supports it out of the box, as well as many drawing programs such as Adobe Illustrator.

Program Options

You can specify a couple of options applicable in exporting barcode images.

Figure 3.2. Export Options Dialog



Y Height

The ratio of the height of a module vs. its with. For better readability, the value should be greater than 3.0. This value affects graphics export only, and only effective when Use value from font is unchecked.

Use value from font

If this box is checked, the y-height will be taken from the current font selected when exporting barcodes to image files.

X Dimension

Apply on Vector image format only. You can specify a nominal size for the module width. You can enter something like "20 mils", or "0.02mm". If unit of measure is not specified, the unit of mils is assumed.

This parameter is taken only when Use value from font size is not checked.

Use value from font size

If this box is checked, the X dimension used when exporting vector graphics files will be taken from the PDF417 font and its font size, instead of the X dimension box. This will create a barcode closely resemble the one displayed on the screen. By default this box is checked.

Target Resolution

Apply on vector image format only. If you export a vector graphic images to a low resolution device such as screen or thermal printer, you should fill this field with the value of the resolution. For screen, use 96. This value makes sure that the length units are properly aligned to the edge of pixels when rasterized.

Number Pixels Per Module

This option applies on raster images only. Specify the number of pixels of a module (the real estate unit of a PDF417 barcode).

Foreground Color / Background Color

Specify the color for image background and foreground. Color values are specified using six hexidecimal digits, with the components in the order of red, green and blue. For exmaple 000000 is the value for black color, and ff0000 is the value for pure red.

Chapter 4. Working with Microsoft Office Programs

Some office programs, such as Excel and Access, support programming using VBA (Visual Basic for Application). Some programs support embedding ActiveX controls. All office programs seem to accept copy/paste of RTF and EMF objects. The table below summarizes the support options of various Office components (version 2007).

Table 4.1. Interoperability with Microsoft Office Programs

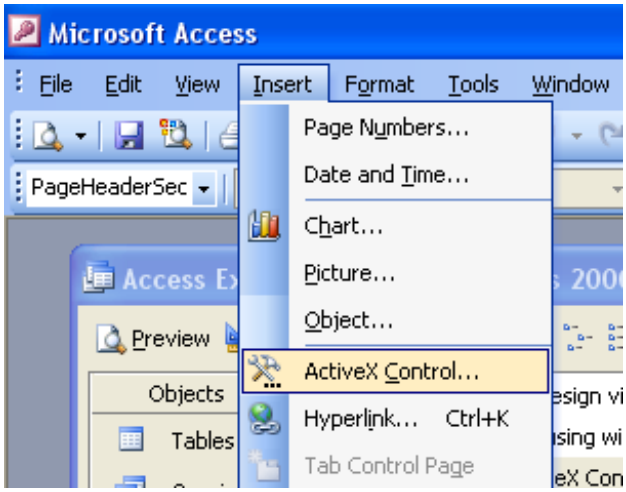
Application	RTF/EMF (from GUI encoder)	ActiveX control	Programmable
Word	both	Yes	No
Excel	EMF[a]	Yes	
Access	Both	Yes	Yes
PowerPoint	Both	Yes	No
OneNote	Both	No	No
Publisher	Both	No	No
Visio	Both	No	No

[a]Excel does not format line gaps properly. Consequently you can't put RTF in an Excel cell.

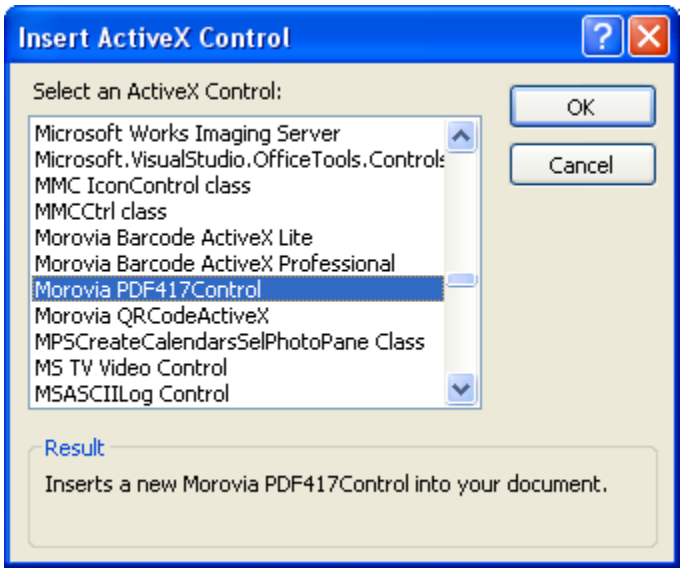
Access (ActiveX Control)

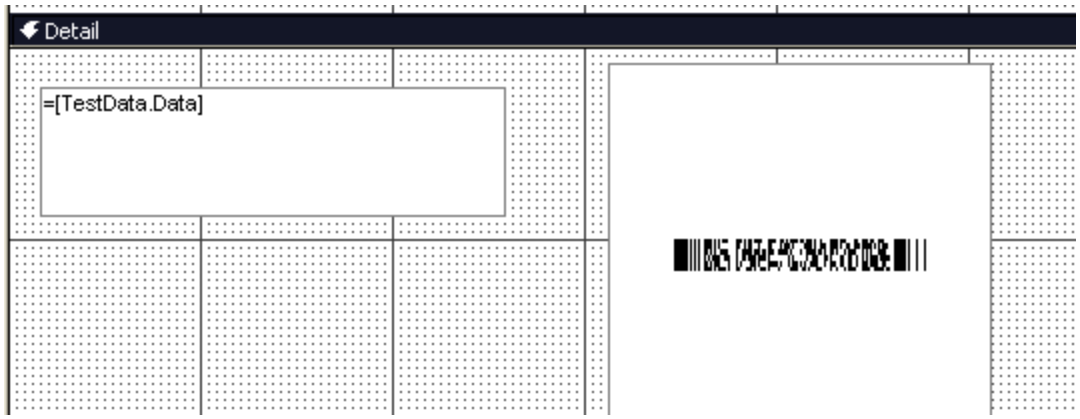
This section explains the steps to add PDF417 barcodes in a Microsoft Access report using Morovia PDF417 Control.

1. The PDF417 ActiveX control must be installed and registered on the computer.
2. Open a report in design view and choose Insert → ActiveX Control...

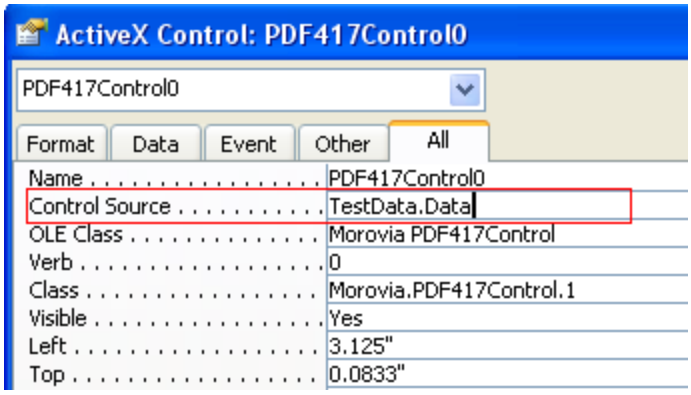


3. From the list of controls, select `Morovia PDF417 Control`.





4. After the control is placed on the report, right click on it and choose Properties...
5. Modify the Control Source property to point to the table and filed of the data you wish to encode into the barcode.

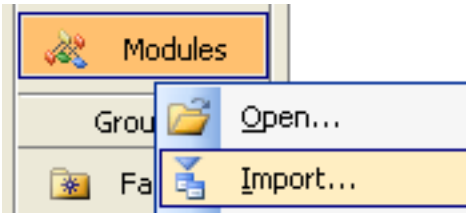


6. Modify other properties, such as `ModuleWidth`, `Cols`, `Rows` and `AspectRatio`. After you are satisfied with the result, close the property dialog.
7. Save the run the report. You should see the barcode appear in the report.

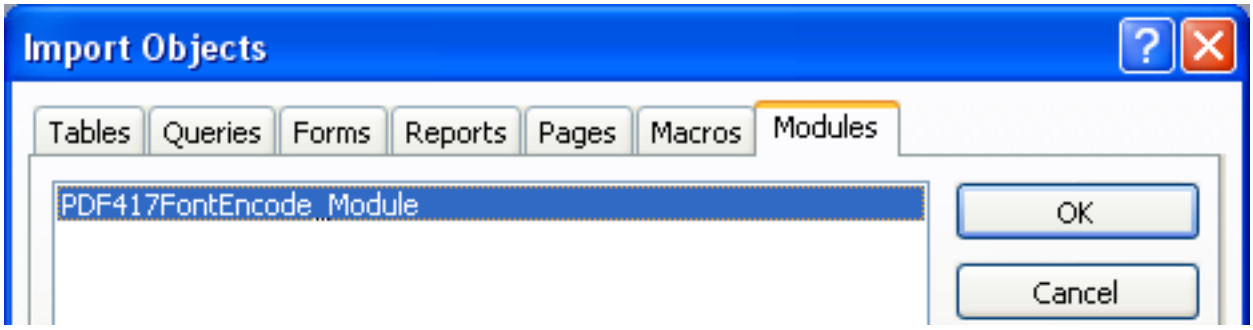
Access (PDF417 Font)

You can also add PDF417 barcodes to Access report using font-based solution.

1. Before creating barcodes in Microsoft Access, you must import the required module. This module adds VBA function so that you can put into the report. Choose Modules → Import and select `Access Example.mdb` file, located in the program folder.

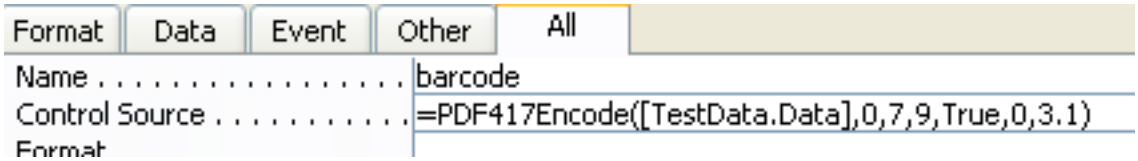


2. Choose `PDF417FontEncode_Module` from the other database. After it is properly imported, it will appear as one of the modules in the database.



3. Open a report in design view and add a text box to the report. The text box will be modified to contain a barcode.
4. Right click on the text box and choose properties.
5. Place the formula `=PDF417Encode([TestData.Data],0,7,9,True,0,3.1)` in the control source property of the text box where [TestData.data] is the field that contains the data to be encoded into the PDF417 barcode. The following parameters are `rows`, `cols`, `security level`, `full size PDF flag`,

`aspect ratio` and `y-height`. Note that the y-height value should match the one in the font to be used. In our case, the font we will use is `MRV PDF417 N3`, which has a fixed y-height of 3.1.



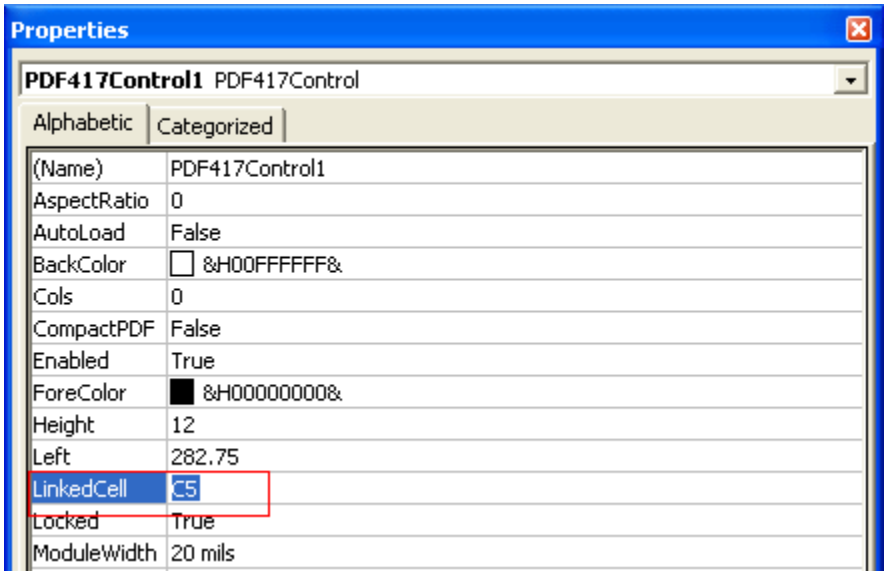
6. Run the report. You should see lines of hexadecimal characters appear in the place of the text box. This is the barcode string in the raw text form.
7. Go back to the design view and change the font of the text box. In our case, choose `MRV PDF417 N3` and 8 points. Adjust the size of the text box to fit the whole barcode.
8. Save and run your report. You should see the barcodes appear in the report.

Excel (ActiveX Control)

Excel has line gap issues with PDF417 fonts. You can paste EMF image from GUI encoder, or use ActiveX control as outlined below.

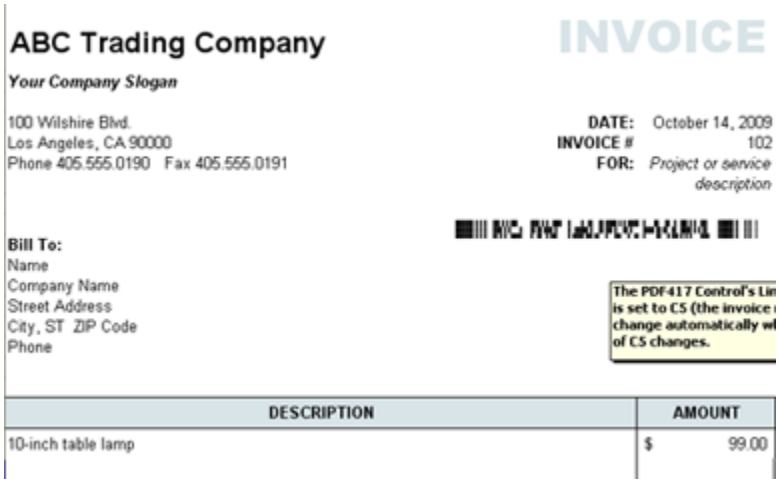
1. After you finished other parts of the spreadsheet, choose View → Toolbars → Control Toolbox
2. When Control Toolbox appears, click on More Controls button.

3. From the list of controls presented, choose Morovia PDF417 Control.
4. Select the area to place the control in the spreadsheet.
5. Right click on the control, choose Properties and change the Linked Cell property to the name of the cell that contains the data you wish to encode.



6. Change other properties as necessary such as `Rows`, `Cols` and `AspectRatio` to adjust the size of the barcode.
7. After editing the properties, click on Exit Design Mode button to exit design mode. The barcode will appear in the spreadsheet.

8. The barcode is now bound to the cell. Change the data of the linked cell, the barcode will change accordingly.



Note: to subsequently modify or delete the barcode control, Excel must enter `Design Mode`. This can be done by pressing the Design Mode button on the Control Toolbox.

Microsoft Word

Using PDF417 control in Microsoft Word is similar to the one in Excel, except that Word does not provide a way for data binding.

1. choose View → Toolbars → Control Toolbox.
2. In the toolbox, choose the more controls button.
3. Select Morovia PDF417Control from the list of available ActiveX controls. After selecting it, the control will appear in the document, the control may be sized as necessary. To change the properties of the control right click on the control and choose Properties.
4. When finished, exit the design mode by choosing the design mode button.
5. To edit the properties of the control the program must be in design mode. If there are problems editing the properties of the control, press the design mode button to enable it.

Word Mail Merge

This tutorial uses Excel file `Word Mail Merge DataSource.xls` as mail merge data source. The data looks like this:

	A	B	C	D	E	F	G	H
1	ID	Company	Address	City	State	ZIP	First Name	Last Name
2	87456321	P. PHOENIX DESIGNS (INC)	2260 MAIN ST	WASHINGTON	DC	20008	JAYNE A.	PHOENIX
3	76543219	ZACHARY'S JEWELRY ETC., INC	1501 LOMAS BLVD NW	NEW YORK	NY	10036-123	SACHARY	SCHMIDT
4	65432198	CAPITAL JEWELERS, LLC	6101 MENAUL BLVD NE	NEW YORK	NY	10036-987	DAVID	CAPITAL
5	54321987	TOM JONES & CO. JEWELERS	1160 EL PASEO D-3	NEW YORK	NY	10036-654	HAL W.	JONES
6	43219873	AUSTIN POWER JEWELRY	230 E IDAHO AVE	NEW YORK	NY	10036-543	KEITH	AUSTIN
7	32198765	THE JEWELERS WORKBENCH	609 N MAIN ST	NEW YORK	NY	10036-943	JOHN	LUSK

We want to print address information as well as a PDF417 barcode that encodes such information in Avery label paper 5163. The Print Preview looks like the one below:

1. Before we start, we need to import a module into Excel. To do that, open Visual Basic Editor. In Excel 2007, this is done by selecting Developer → Visual Basic.

   In Visual Basic Editor, choose File → Import File. Navigate to the PDF417 Fontware installation folder, and select the `Morovia.PDF417FontDLL.bas`.

   Close Visual Basic Editor.

2. Add a new column in the spreadsheet that will hold the barcode string. In our case, we use column H. In cell H2, enter the definition as below:

```
=PDF417Encode(CONCATENATE(G2, " ", H2, CHAR(10), D2, ", ", E2, " ", F2), 0, 4, 9, TRUE, 0, 3.1)
```

	C	D	E	F	G	H	I
	Address	City	State	ZIP	First Name	Last Name	Barcode
NS (INC)	2260 MAIN ST	WASHINGTON	DC	20008	JAYNE A.	PHOENIX	FFFFFFFF0F0F0F000FFCFC795B(
RY ETC., INC	1501 LOMAS BLVD NW	NEW YORK	NY	10036-123	SACHARY	SCHMIDT	FFFFFFFF0F0F0F000FFCFC795B(

   After hitting **Enter**, you should see a hexadecimal string result show up. If not, examine the formula you entered.

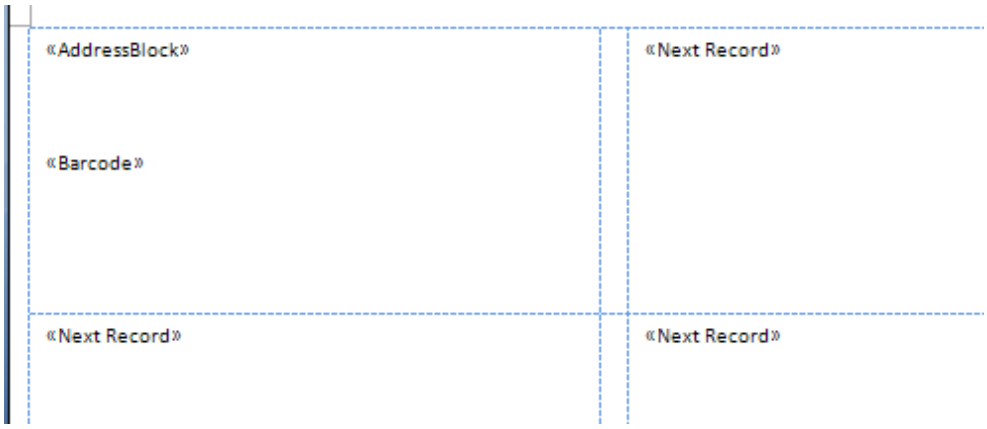
   Note the use of Excel function `CONCATENATE` here. This function is used to combine several fields, as well as line return characters.

   See the DLL API for the meaning of each field. Here we require the number of columns as 4.

3. Copy the formula to other cells of the same column. This can be done by selecting cell H2, highlighting the cells that the formula is copied, and select Paste.

   Close the Excel file and start Microsoft Word.

4. In Microsoft Word, choose Mailing → Start Mail Merge → Labels. Select Avery 5163 as the label we will work on.
5. Select Select Recipients Select Existing List. In the file dialog, navigate to the spreadsheet we just created. If it asks for Select Table, choose Sheet1$.
6. Click on Address Block. This is for the address line. Microsoft Word has the intelligence to select the address block.
7. Hit **Enter** once to move the cursor below, and click on Insert Merge Field. Select Barcode as the merge field. The document looks like the one below:



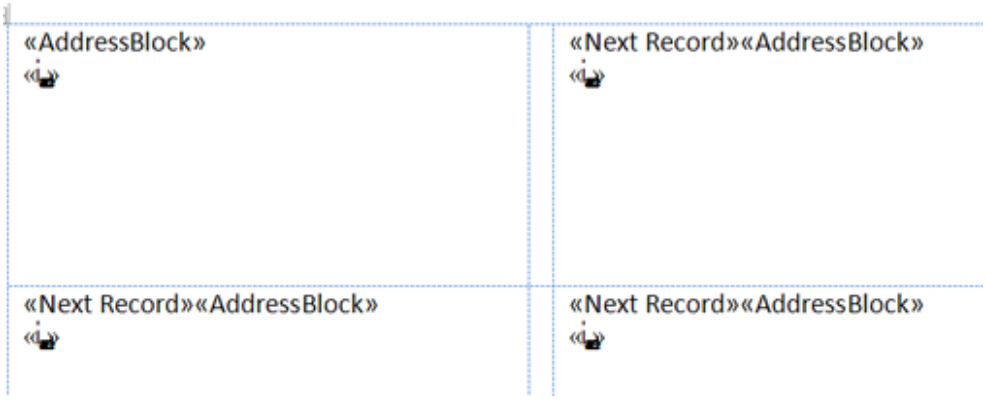
   Click on Review Results. You should see the first record show up, with hexadecimal characters in the place of barcode.

8. Now adjust the font for the address block and the barcode field. For the address block, we use Arial 16 points. For the barcode, use `MRV PDF417 N3` 14 points. Note: the font should match the yHeight value in the formula.

Warning Make sure that there is no gap between two lines in the barcode field. If you see blanks inside the barcode, adjust the paragraph setting by selecting Paragraph Spacing. Use 0 for Before and After spacing, and Single for Line spacing.

---

9. Copy the format to other labels by selecting Update labels.



10. Select Preview Results to view the sheets of labels.

Chapter 5. Encoder DLL API Reference

Creating PDF417 barcodes using DLL APIs involves three steps. The first step is to call `PDF417Encode` with data to encode and the PDF417 options. If this step succeeds, a pointer is returned from which you can query the attributes of the barcode created, or "paint" the barcode into an image file. Finally, you should call `DestroyPDF417Result` to release the memory resource used by the encoder result object.

For the meanings of PDF417 barcode attributes, such as rows, columns, aspect ratio and security level, see Chapter 2, *Overview*.

Table 5.1. List of Enumerations

Enumeration	Comment
ImageTypeEnum	Image formats supported by the encoder.

Table 5.2. List of Functions

Function	Comment
PDF417Encode	Encodes data and returns a pointers that points to encoder result object.
PDF417Encode2	Abbreviated version of PDF417Encode.
DestroyPDF417EncodeResult	Releases all resources allocated to the encoder result object.
PDF417ResultGetNumRows	Retrieves the number of rows of the PDF417 barcode created.
PDF417ResultGetNumCols	Retrieves the number of data columns of the PDF417 barcode created.
PDF417ResultGetSecurityLevel	Retrieves the security level of the PDF417 barcode created.
PDF417ResultGetAspectRatio	Retrieves the actual aspect ratio of the PDF417 barcode created.
PDF417ResultGetBarcodeString	Retrieves the barcode string that becomes a PDF417 symbol after being formated with a PDF417 font.
PDF417ResultGetBarcodeString2	Returns barcode string without requiring user to preallocate the buffer.
PDF417GetErrorMessage	Retrieves a string that describes the error.
PaintPDF417ImageRaster	Writes the PDF417 barcode in raster image format specified to a disk file.
PaintPDF417ImageVector	Writes the PDF417 barcode in vector image format specified to a disk file.
PaintPDF417ImageClipboard	Creates a Windows Enhanced Meta File object and place it into the clipboard.

ImageTypeEnum

Image formats supported by the encoder.

```
enum ImageTypeEnum {
    IMAGE_PNG== 0,
    IMAGE_SVG== 1,
    IMAGE_EMF== 2,
    IMAGE_EPS== 3,
    IMAGE_BMP== 4
};
```

Remarks

The ImageTypeEnum enum has the following values:

Table 5.3. ImageTypeEnum Enumeration

Constant	Value	Comment
IMAGE_PNG	= 0	PNG (Portable Network Graphics)
IMAGE_SVG	= 1	SVG (Scalable Vector Graphics)
IMAGE_EMF	= 2	EMF (Windows Enhanced MetaFile)
IMAGE_EPS	= 3	EPS (Encapsulated PostScript)
IMAGE_BMP	= 4	BMP (Windows Bitmap)

PNG and BMP are raster formats, which store color information of individual pixels. SVG, EMF and EPS are vector graphics formats and contains drawing commands instead. If you are using raster graphic format, one pixel should be mapped to one or integral times pixels on the printer. If you are using vector graphics format, the drawing units should map to integral times pixels on the printer. They are referenced in the PaintPDF417ImageRaster function and PaintPDF417ImageVector function.

PDF417Encode

The **PDF417Encode** function encodes data and returns a pointers that points to encoder result object.

```
int __stdcall  PDF417Encode(
    const char * dataToEncode,
    int numRows,
    int numCols,
    int securityLevel,
    int fullsizePDF,
    double aspectRatio,
    double yHeight,
    void ** ppResult
);
```

Parameters

dataToEncode

[in] Pointer to a null-terminated string containing the data to be encoded. You can use tilde codes to encode control characters such as NUL, as well as advanced features such as ECI and MacroPDF417. For more information, see the section called “Input Format”.

numRows

[in] An integer that indicates the number of rows desired in the barcode created. Valid values are between 3 and 90, or 0 for automatic selection.

numCols

[in] An integer that indicates the number of data columns desired in the barcode created. Valid values are between 1 and 30, or 0 for automatic selection.

securityLevel

[in] An integer value that corresponds to the security level desired. Valid values are between 0 and 9. If 9 is specified, the program will select a security level based on the data encoded, based on the method described in the standard. Note: The security level in the result can be higher than specified, 10:30 PM

2/20/2010 but will never be lower. When the program finds that there is room for additional security codewords in the symbol, it will increase the security level automatically. To find out the actual security level in the symbol created, use PDF417ResultGetSecurityLevel function.

fullsizePDF

[in] An integer value that indicates whether to produce full-sized PDF417 symbol, or a compact version. If the value is 0, compact PDF symbols are produced.

aspectRatio

[in] The aspect ratio of the symbol desired. If 0 is specified, the program does not use it as a criteria.

yHeight

[in] The ratio of module height vs. module width. When you are createing barcodes using the fonts, use the value that match the font. For example, the yHeight for font 'MRV PDF417 N3' is 3.1.

ppResult

A pointer to a pointer that points to the encode result created. Use this pointer to discover the actual attributes of the symbol created, or create image files.

Return Values

If the function succeeded, the return value is 0. If the function failed, it returns the error code. You can call function PDF417GetErrorMessage to obtain the description of the error.

Remarks

This function encodes the data according to the parameter specified, and returns a pointer from which you can retrieve the encoding result.

PDF417Encode2

The function **PDF417Encode2** is an abbreviated version of PDF417Encode.

```
void *__stdcall    PDF417Encode2(
    const char * dataToEncode,
    int numRows,
    int numCols,
    int securityLevel,
    int fullsizePDF,
    double aspectRatio,
    double yHeight
);
```

Parameters

dataToEncode

[in] Pointer to a null-terminated string containing the data to be encoded. You can use tilde codes to encode control characters such as NUL, as well as advanced features such as ECI and MacroPDF417. For more information, see the section called “Input Format”.

numRows

[in] An integer that indicates the number of rows desired in the barcode created. Valid values are between 3 and 90, or 0 for automatic selection.

numCols

[in] An integer that indicates the number of data columns desired in the barcode created. Valid values are between 1 and 30, or 0 for automatic selection.

securityLevel

[in] An integer value that corresponds to the security level desired. Valid values are between 0 and 9. If 9 is specified, the program will select a security level based on the data encoded, based on the method described in the standard. Note: The security level in the result can be higher than specified. but will never be lower. When the program finds that there is room for additional security codewords in the symbol, it will increase the security level automatically. To find out the actual security level in the symbol created, use PDF417ResultGetSecurityLevel function.

fullsizePDF

[in] An integer value that indicates whether to produce full-sized PDF417 symbol, or a compact version. If the value is 0, compact PDF symbols are produced.

aspectRatio

[in] The aspect ratio of the symbol desired. If 0 is specified, the program does not use it as a criteria.

yHeight

[in] The ratio of module height vs. module width. When you are createing barcodes using the fonts, use the value that match the font. For example, the yHeight for font 'MRV PDF417 N3' is 3.1.

Return Values

If the function failed, the return value is 0 (NULL). If it succeeds, it returns a pointer that points to the result object. You should release the encoder result object by calling function DestroyPDF417EncodeResult.

Remarks

In this function, a pointer is returned instead of status code. A non-null return value indicates that no error occured. The pointer associated resource must be released by calling DestroyPDF417EncodeResult.

The function **PDF417Encode2** is an abbreviated version of PDF417Encode.

DestroyPDF417EncodeResult

The **DestroyPDF417EncodeResult** function releases all resources allocated to the encoder result object.

```
void __stdcall DestroyPDF417EncodeResult(
    void * pResult
);
```

Parameters

pResult

[in] Pointer to the encoder result object.

Remarks

After the object is destroyed, the specific pointer pResult is no longer valid.

PDF417ResultGetNumRows

The **PDF417ResultGetNumRows** function retrieves the number of rows of the PDF417 barcode created.

```
int __stdcall  PDF417ResultGetNumRows(
    void * pEncodeResult
);
```

Parameters

pEncodeResult

[in] The pointer that points to the encoding result object, returned from PDF417Encode or PDF417Encode2 functions.

Return Values

The number of data rows in the PDF417 barcode. Always succeed.

PDF417ResultGetNumCols

The **PDF417ResultGetNumCols** function retrieves the number of data columns of the PDF417 barcode created.

```
int __stdcall  PDF417ResultGetNumCols(
    void * pEncodeResult
);
```

Parameters

pEncodeResult

[in] The pointer that points to the encoding result object, returned from PDF417Encode or PDF417Encode2 functions.

Return Values

The number of data columns in the PDF417 barcode. Always succeed.

PDF417ResultGetSecurityLevel

The **PDF417ResultGetSecurityLevel** function retrieves the security level of the PDF417 barcode created.

```
int __stdcall  PDF417ResultGetSecurityLevel(
    void * pEncodeResult
);
```

Parameters

pEncodeResult

[in] The pointer that points to the encoding result object, returned from PDF417Encode or PDF417Encode2 functions.

Return Values

The actual security level in the PDF417 barcode. It should be a number bewteen 0 and 8.

Remarks

PDF417 specification defines nine different levels for security codewords, ranging between 0 and 9. In our software you can use 9 for automatic security level selection. The encoder always selects that the highest security level possible within the size constraint. To obtain the actual security level in the barcode, call function PDF417ResultGetSecurityLevel.

PDF417ResultGetAspectRatio

The **PDF417ResultGetAspectRatio** function retrieves the actual aspect ratio of the PDF417 barcode created.

```
double __stdcall   PDF417ResultGetAspectRatio(
    void * pEncodeResult
);
```

Parameters

pEncodeResult

[in] The pointer that points to the encoding result object, returned from PDF417Encode or PDF417Encode2 functions.

Return Values

The actual aspect ratio of the PDF417 barcode.

Remarks

The Aspect Ratio refers to the ratio of height to width of a bar code symbol. A code twice as high as it is wide has an aspect ration of 2; a code twice as wide as it is high has an aspect ratio of 0.5.

PDF417ResultGetBarcodeString

The **PDF417ResultGetBarcodeString** function retrieves the barcode string that becomes a PDF417 symbol after being formated with a PDF417 font.

```
int __stdcall  PDF417ResultGetBarcodeString(
    void * pEncodeResult,
    char * buffer,
    unsigned int * maxSize,
    const char * eol
);
```

Parameters

pEncodeResult

[in] The pointer that points to the encoding result object, returned from PDF417Encode or PDF417Encode2 functions.

buffer

[in] The pointer that points to a byte array that receives the string.

maxSize

[in,out] Pointer to a variable that specifies the size of the buffer pointed to by the buffer parameter, in bytes. When the function returns, this variable contains the size of the data copied to buffer.

eol

[in] Pointer to a NUL terminated string that will be appended to each row in the barcode string.

Return Values

If the function succeeded, it returns the length of the string (excluding terminating NUL character). If the funtion fails due to insufficient storage, it returns 0.

Remarks

When creating barcodes using font-based solution, you first call encoder function PDF417Encode or PDF417Encode2 to obtain a pointer that points to the result object. Then PDF417ResultGetBarcodeString is called to obtain a string that becomes PDF417 symbol after the font is applied. If memory is not an issue,

allocate a large buffer with 8096 bytes to hold the largest PDF417 barcode. This size is sufficient for the largest symbol with carriage return and line feed as line ending.

PDF417ResultGetBarcodeString2

The **PDF417ResultGetBarcodeString2** function returns barcode string without requiring user to preallocate the buffer.

```
const char *__stdcall  PDF417ResultGetBarcodeString2(
    void * pEncodeResult,
    const char * eol
);
```

Parameters

pEncodeResult

[in] The pointer that points to the encoding result object, returned from PDF417Encode or PDF417Encode2 functions.

eol

[in] Pointer to a NUL terminated string that will be appended to each row in the barcode string.

Return Values

Pointer to a NUL terminated string that represents the barcode (barcode string). If the function failed due to insufficient memory, it returns NULL.

Remarks

This function was added in 4.1 release to make it easier for the caller to retrieve barcode string. The encoder result manages the buffer by itself. Caller should cache the string returned, but not the pointer, as the contents may change after another call of PDF417ResultBarcodeString2 is made.

PDF417GetErrorMessage

The **PDF417GetErrorMessage** function retrieves a string that describes the error.

```
const char *__stdcall  PDF417GetErrorMessage(
    int errorno
);
```

Parameters

errorno

[in] The error number.

Return Values

a read only string that describes the error.

Remarks

The string returned is a read only string. User should not modify the string directly.

PaintPDF417ImageRaster

The **PaintPDF417ImageRaster** function writes the PDF417 barcode in raster image format specified to a disk file.

```
int __stdcall  PaintPDF417ImageRaster(
    void * pEncodeResult,
    const char * lpszFilename,
    int pixelsPerModule,
    int forecolor,
    int backcolor,
    int imageType
);
```

Parameters

pEncodeResult

[in] Pointer that points to the encoder result object.

pszFilename

[in] Pointer to the file name for the image file to be created.

pixelsPerModule

[in] Number pixels per module with.

forecolor

[in] Value of the foreground color, in RGB colorspace.

backcolor

[in] Value of the background color, in RGB colorspace.

imageType

[in] An integer that indicates the image file format. The current version supports two formats: PNG(0) and BMP(4).

Return Values

If the function failed, it returns the error code. You can call function PDF417GetErrorMessage to obtain the description of the error.

Remarks

Note: the order of the component bytes are R, G and B, which is opposite of **COLORREF** type on Windows.

PaintPDF417ImageVector

The **PaintPDF417ImageVector** function writes the PDF417 barcode in vector image format specified to a disk file.

```
int __stdcall  PaintPDF417ImageVector(
    void * pEncodeResult,
    const char * lpszFilename,
    int module_width_hm,
    int target_dpi,
    int forecolor,
    int backcolor,
    int imageType
);
```

Parameters

pEncodeResult

[in] Pointer that points to the encoder result object.

module_width_hm

[in] Module width (X dimension), in the unit of high metric. 1 unit high metric = 1/1000 cm.

target_dpi

[in] The resolution of the target printer.

forecolor

[in] Value of the foreground color, in RGB colorspace.

backcolor

[in] Value of the background color, in RGB colorspace.

imageType

[in] An integer that indicates the image file format. The current version supports three formats: SVG(1), EMF(2) and EPS(3).

Return Values

If the function failed, it returns the error code. You can call function PDF417GetErrorMessage to obtain the description of the error.

Remarks

Note: the order of the component bytes are R, G and B, which is opposite of **COLORREF** type on Windows.

PaintPDF417ImageClipboard

The **PaintPDF417ImageClipboard** function Creates a Windows Enhanced Meta File object and place it into the clipboard.

```
int __stdcall  PaintPDF417ImageClipboard(
    void * pEncodeResult,
    int module_width_hm,
    int target_dpi,
    int forecolor,
    int backcolor
);
```

Parameters

pEncodeResult

[in] Pointer that points to the encoder result object.

module_width_hm

[in] Module width (X dimension), in the unit of high metric. 1 unit high metric = 1/1000 cm.

target_dpi

[in] The resolution of the target printer.

forecolor

[in] Value of the foreground color, in RGB colorspace.

backcolor

[in] Value of the background color, in RGB colorspace.

Return Values

If the function failed, it returns the error code. You can call function PDF417GetErrorMessage to obtain the description of the error.

Chapter 6. PDF417 ActiveX Reference

Many programming environments support the use of ActiveX objects. We have tested the Morovia PDF417 ActiveX with Visual Basic, Visual C++, Internet Explorer, Microsoft Word, Excel, Access and IIS programs. The object can be used as a control embedded in a VB form or a dialog, or be used at background for image creation and printing purposes. The PDF417 ActiveX object can also be inserted into Microsoft Office documents and many other ActiveX-aware programs.

Specification

Table 6.1. PDF417 Control Specification

Prog ID	Morovia.PDF417Control
ClassID	{89C50E31-CBFE-4AF3-B36F-DFD8FE60ACD0}
Licensed	no
File name	PDF417Ctrl.dll
Interface	IPDF417Control
Interface ID	{F6FC2671-7888-4DDB-8DAF-A6544C205546}

Properties

Table 6.2. List of PDF417 Control Properties

Name	Description
AspectRatio	Returns or sets a value for the overall height to width ratio of PDF417 symbols generated.
BackColor	Specifies the background color for the control.
Cols	Returns or sets a value for the number of codeword columns in PDF417 symbols generated.
CompactPDF	Returns or sets a value that determines whether to generate the compact version of all PDF417 symbols generated.
ForeColor	Specifies the foreground color for the control.
ModuleWidth	Specifies the width of a module (the smallest unit in a PDF417 barcode).
Picture	Returns a snapshot of the drawing in Windows Enhanced Metafile Format (EMF).
Rows	Returns or sets a value for the maximum number of codeword rows in all PDF417 barcodes generated.
SecurityLevel	Returns or sets a value for security level used in all PDF417 barcodes generated.
TargetDPI	Specifies the value of DPI when exporting the images to vector graphics.
Text	Specifies the data to encode.

Name	Description
YHeight	Returns or sets a value that reflects the height of the module vs. the width in all PDF417 barcodes generated.

Table 6.3. List of PDF417 Control Methods

Name	Description
CopyToClipboard	Place the drawing to clipboard in EMF format.
GetActualAtrributes	Retrieves the actual values of atrtibutes in the PDF417 barcodes generated.
ExportImageVector	Export image to a file in vector graphics format specified (EMF, EPS or SVG).
ExportImageRaster	Export image to a file in raster graphics fromat specified (PNG or BMP).

AspectRatio Property

Description

Returns or sets a value for the overall height to width ratio of PDF417 symbols generated.

Syntax

```
object.ActualVersion = [Number]
```

Remarks

The `AspectRatio` determines the overall shape of the PDF417 symbol and is defined as 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.

Generaly speaking, a lower `AspectRatio` produces more space-efficient barcodes.

Setting the value to 0 allows the program to select a size that produces the least number of codewords, under the constraints of `Rows` and `Cols` specified.

The default value is 0.0 (automatic selection).

BackColor, ForeColor Properties

Description

BackColor - returns or sets the background color of the control.

ForeColor - returns or sets the foreground color of the control.

Syntax

```
object.BackColor[= Color]
object.ForeColor[= Color]
```

Remarks

For open systems we strongly recommend to set the background color to solid white (`0xFFFFFF`) and foreground color to black (`0x000000`). Note: barcode requires decent contrast between the foreground color

and the background color in order to be readable. Always test the readability thoroughly when you select a color pair different from black and white.

Cols Property

Syntax

```
object.Cols = Number
```

Description

Returns or sets a value for the number of codeword columns in PDF417 symbols generated.

Remarks

When the value is 0, the program selects one that produces the least number of codewords.

The default value is 0.

CompactPDF Property

Syntax

```
object.CompactPDF = Boolean
```

Description

Returns or sets a value that determines whether to generate the compact version of PDF417 symbols.

Remarks

PDF417 standard specifies a compact version, with the stop pattern reduced to a single terminating bar. Compact version symbols should only be used in a clean and controlled environment.

The default value for this property is `FALSE`.

ModuleWidth Property

Description

Returns or sets a value that determines the width of a single module in the PDF17 symbols generated.

Syntax

```
object.ModuleWidth[= String]
```

Remarks

The "real estate" unit of a PDF417 symbol, the *module*, is always rectangular. This property sets the width of the rectangle. It affects the overall symbol size.

The default value for **ModuleSize** is `20` mils. The property can be any numbers between `1` and `100`.

This property is a string, and accepts the following units: mil, himetric, mm, cm, pt and inch. For example, “0.01cm” and “1mm” are all valid. If no measure unit is specified, unit of mil (which is 1/1000th inch) is assumed.

Picture Property

Description

Readonly property. Returns a snapshot of the drawing in Windows Enhanced Metafile Format (EMF).

Syntax

```
object.Picture
```

Remarks

The *Picture* property provides a convenient method to retrieve the drawing without first saving it to disk. The picture object contains an enhanced metafile handle which can be passed to *clipboard* or played on a device.

The included VB6, VC++ and C# examples use this property to retrieve an EMF handle and play it on the target printer to print the barcode.

Rows Property

Description

Readonly property. Returns the number of rows requested.

Syntax

```
object.Rows
```

Remarks

Set the number of rows required for PDF417 barcodes generated. 0 means automatic.

SecurityLevel Property

Description

Specifies the security leve in the PDF417 barcode generated.

Syntax

```
object.SecurityLevel = Number
```

Remarks

In PDF417 symbols security level is selectable and ranges between 0 and 8. Specifying 9 for automatic selection based on the amount of data encoded. The value is considered "minimum" by our software, as it will increase the security level if the size constraint allows. To retrieve the actual value of the security level, use `GetActualAttributes` method.

TargetDPI Property

Specifies the resolution of the target printer, applicable during exporting vector graphics.

Syntax

```
object.TargetDPI = Number
```

Remarks

Whe producing small sizes of barcodes, especailly on lowe resolution printers, lengths must be align to the boundary of pixels in order to retain high quality. The ActiveX control adjusts the drawing units automaticaly based on this property when exporting vector graphics images.

Text Property

Specifies the Data to be encoded into the PDF417 symbol.

Description

Returns or sets a string for the message to be encoded.

Syntax

```
object.Text[= String]
```

Remarks

YHeight Property

Specifies the ratio of height of a module vs. its width.

Description

Returns or sets the y-height in the PDF417 barcode generated.

Syntax

```
object.YHeight[= Double]
```

Remarks

For better decodability, this value should be greater than 3.0.

CopyToClipboard Method

Copies the PDF417 image into the system clipboard, in EMF format.

Syntax

```
obj.CopyToClipboard
```

Description

Use this method to place a copy of barcode image to the clipboard, so that you can either retrieve it immediately in your application, or other applications such as Microsoft Word.

ExportImageRaster Method

Description

Exports the current drawing to a disk file, in raster image format (PNG or BMP).

Syntax

```
obj.ExportImageRaster(filename, pixelsPerModule, imageType)
```

Parameters

filename

[in] filename. A string that corresponds to the file to be written.

pixelsPerModule

[in] Number of pixels per module.

imageType

[in] Image type. The current version supports two raster image formats:

- 0 - PNG
- 4 - BMP

ExportImageVector Method

Description

Exports the current drawing to a disk file, in raster image format (EMF, SVG and EPS).

Syntax

```
obj.ExportImageVector(filename, imageType)
```

Parameters

filename

[in] filename. A string that corresponds to the file to be written.

imageType

[in] Image type. The current version supports two three image formats:

- 1 - SVG
- 2 - EMF
- 3 - EPS

GetActualAttributes Method

Description

Retrieves the actual values of atrtibutes in the PDF417 barcodes generated.

Syntax

```
obj.GetActualAttributes(numRows, numCols, securityLevel, aspectRatio)
```

Parameters

numRows

[out] The actual number of rows in the barcode generated.

numCols

[out] The actual number of cols in the barcode generated.

securityLevel

[out] The actual security level in the barcode generated.

aspectRatio

[out] The actual aspect ratio of the barcode generated.

Remarks

You can specify exact values to Rows, Cols, SecurityLevel and AspectRatio, or leave one or more of them to be determined by the program. Use this method to retrieve the actual value in the barcode generated.

Chapter 7. Adding PDF417 Symbols to Crystal Reports

Adding PDF417 barcodes to Crystal Reports is straightforward. The example included in the software was authored in Crystal Reports 9.

Warning Due to the length constraint imposed by Crystal Reports, the approach outlined in this chapter requires Crystal Reports version 9 and above, unless you are encoding fairly small amount of data.

User Function DLL

The UFL in version 5 is changed in order to work with 64-bit Crystal runtime. The new UFL is a COM-type dll and installed in the software's installation directory. Both 32-bit and 64-bit DLLs are included in the software.

PDF417 Fontware and Writer SDK 4 includes a type-25 UFL for Crystal Reports. Unfortunately type-25 UFL is not supported in 64-bit Crystal Reports. A new DLL is included in version 5 release as a result. However the function names are changed. If you have reports authored in version 4, you must change the function names. The previous version UFL is copied to the installation directory. However it is not visible to Crystal Reports. To allow old reports continue to work, copy the DLL to Windows `system32` directory (on 64-bit Windows, the directory is `C:\windows\SysWOW64`, or to Crystal runtime directory (the one that has `crpe32.dll`.

This UFL exports two functions:

Table 7.1. List of Crystal Reports UFL Functions (`crufl PDF417V5.dll`)

Function Name	Comment
Number PDF417EncodeSet (DataToEncode, numRows, numCols, securityLevel, aspectRatio, yHeight)	Encodes the data into a full sized PDF417 barcodes with attributes specified. Returns the number of segments required for the data encoded.
Number PDF417CompactEncodeSet (DataToEncode, numRows, numCols, securityLevel, aspectRatio, yHeight)	Encodes the data into a compact PDF417 barcodes with attributes specified. Returns the number of segments required for the data encoded.
String PDF417EncodeGet(chunkNo)	Returns a segment of the result. Each segment consists of no more than 254 characters (the maximum number allowed by Crystal Reports).

The `PDF417EncodeGet` contains a single parameter which is the chunk index. The two encoder functions, `PDF417EncodeSet` and `PDF417CompactEncodeSet` have size parameters each. The first parameter is data encoded. The following six parameters are number of rows, number of columns, security level, aspect ratio and y-height [1]. The function returns a number, which indicates the number of subsequent calls to `PDF417EncodeGet`. Each call of `PDF417EncodeGet` retrieves a chunk of encoder results.

[1]Note: the y-height specified must match the font used, otherwise the aspect ratio will not be achieved correctly. To learn how these factors affect the final symbol size, see Chapter 2, *Overview*.

The following script demonstrates the basic flow to get it work in Crystal Reports:

```
// Crystal Reports impose a constraint on UFL string length: max 255 characters.
// To get around this limit, use the code below
// This workaround requires Crystal Report Version >= 9

StringVar DataToEncode := "Morovia PDF417 Fontware and Writer SDK " +
        "is a powerful software solution to create PDF417 symbols in Crystal Reports.";

NumberVar numRows := 0;
NumberVar numCols := 3;
NumberVar securityLevel := 9;
NumberVar aspectRatio := 0.0;
// Note: 3.1 is for MRV PDF417N3 font.
// other font requires change of this parameter.
NumberVar yHeight := 3.1;

StringVar BarcodeString := "";

// First find out how many chunks required to get the whole barcode string
NumberVar chunks := PDF417EncodeSet(DataToEncode,
          numRows, numCols,
          securityLevel, aspectRatio,
          yHeight);

// Loop until all the data is retrieved
NumberVar i:=0;
For i :=0 To chunks Step 1 Do
(
    BarcodeString := BarcodeString + PDF417EncodeGet(i);
);

// Returns the complete string
BarcodeString;
```

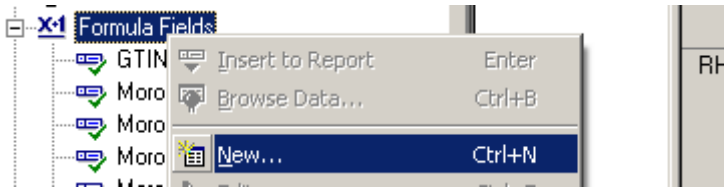
Because of the length restraint that UFL imposes on the string, it is not possible to complete in one UFL call. Instead, you prepare all parameters and call `PDF417EncodeSet`. This function calls encoder at the back end, and returns the number of chunks required. After that the program enters a loop that calls `PDF417EncodeGet` repeatedly until all result are retrieved and stored in `BarcodeString`.

Working with Crystal Reports

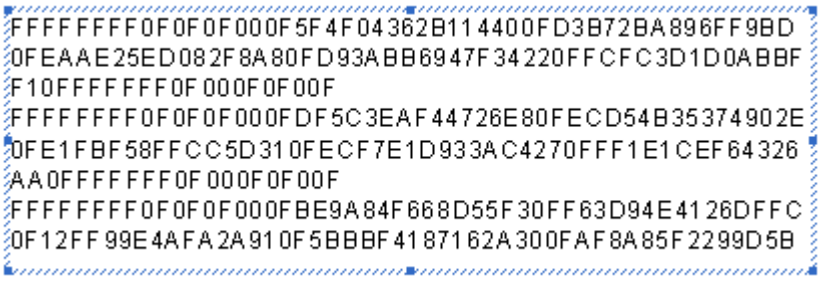
The screen shots in the following tutorial were taken in Crystal Reports 9. Interfaces in other versions may appear differently.

Assume that you have a database table with several fields. You already put them into the report, and want to add a column to hold PDF417 barcode for ID field.

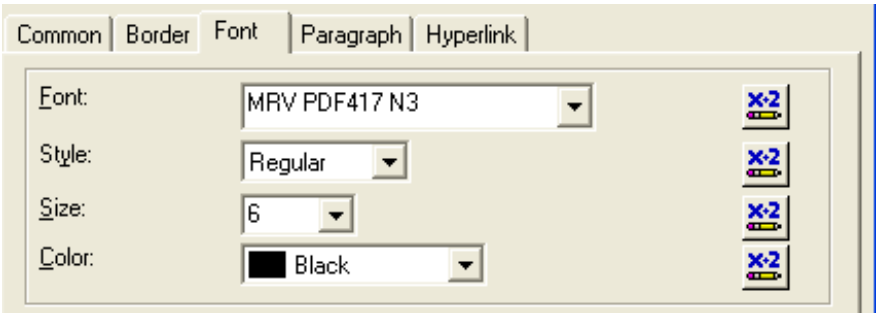
1. First switch to design mode. Choose Report → Formula Workshop.
2. Right click on Formula Fields and choose New.
3. Give your formula field a name, in this example we will name it `Morovia_PDF417`. In versions 9 and above, you will be prompted to use the editor or the expert, choose Use Editor.



4. In the Formula Editor, choose Functions → Additional Functions → Visual Basic UFLs (u2lcom.dll). You should see `PDF417EncodeSet` appears under. If not, the DLL is not installed properly.
5. In the edit box below, copy and paste entire script in the section called "User Function DLL". Make necessary changes. For example, you are likely to change the value of `DataToEncode` variable to one of your database field. Version, Error Correction Level can be changed in the code accordingly.
6. Choose Save → Close to close Formula Workshop.
7. From the Field Explorer, drag the Morovia_PDF417 Formula Field to the report.
8. Choose File → Print Preview. You should see a series of meaningless lower case letters and digits in the box. This is normal as the barcode string is completely difference from the data encoded in case of PDF417 barcode.



9. Switch back to design mode. Right click on the field and choose Format Field. Click on the Font tab, and choose the `MRV PDF417 N3` font. Set the font size to 6 points or to the size appropriate for your environment.



10. Drag the cursor to make the formula field big enough to contain the entire barcode. You may need to adjust both horizontally and vertically. Be sure to leave some spaces for quiet zone requirement.
11. Run the Print Preview again. The barcode should come up. Print a page and scan the barcode to make sure that the font size is appropriate and the bounding rectangle is big enough to hold the whole barcode.

Note If you add barcodes to report using the trial version and subsequently upgrade to the full version, you need to refresh the report once. Otherwise the barcode will continue to be scrambled.

Distributing UFL, Fonts with your report application

Once you finish the report design, you can distribute your report application with Crystal run time files, barcode fonts and the UFL library.

Note Developer License is required to distribute font files and encoder UFL DLL outside your organization.

Two run time files need to be included in your distribution package:

1. **Morovia PDF417 Font Files**. The font files are located `c:\windows\fonts`, and should be copied to the target computer's `fonts` folder.
2. **PDF417 Encoder UFL**. This DLL is installed at `C:\Program Files (x86)\Morovia PDF417 Fonts & Encoder 5` by default. In the production computer, you can place this DLL anywhere. However, this is a COM DLL and requires registration. If you are deploying to a 64-bit Crystal Reports runtime, you need the 64-bit version of the DLL.

Chapter 8. Adding PDF417 Symbols to SQL Server Reporting Service

Adding PDF417 symbols to SQL Reporting Service is straightforward with PDF417 Fonts & Encoder 5. This chapter explains how you can achieive the objective. Our report service plugin supports SQL Reporting Service 2008 and above. The example in this chapter uses Visual Studio 2010/SQL Reporting Service 2012 for demonstration. If you want to follow the tutorial steps by step, the AdventureWorks2012 sample database is required to be installed. You can find the sample database by searching Internet.

Custom Assembly

SSRS can't use the encoder DLL directly. A ready-to-use custom assembly (`ReportServicePlugin_PDF417.dll`) built under Visual Studio 2010 (.net 2.0) is included in the software.

Note In case that you need to build the assembly by yourself, The source code is also included in the distribution. The project files are packed in `ReportServicePlugin_src.zip`, which is located under the program (x86) folder. The build script is written in NAnt.

This assembly exposes two functions: PDF417Encode and PDF417CompactEncode. The prototype for this function is as below:

```
string PDF417Encode(string strDataToEncode,
    int numRows, int numCols,
    int securityLevel,
    double aspectRatio,
    double yHeight);

string PDF417CompactEncode(string strDataToEncode,
    int numRows, int numCols,
    int securityLevel,
    double aspectRatio,
    double yHeight);
```

The two functions have the same signature. The only difference is that the first function produces barcode strings for full size PDF417 barcodes, and the other compact PDF417 barcodes. or the meanings of the parameter, refer to the section called “PDF417Encode2”. Note that the `yHeight` parameter should match the font used in order to calculate the *aspect ratio* correctly, unless you are not using this parameter. To learn how parameters affect the symbol size, see the section called “Sizing Parameters”.

Installing Custom Assembly

Copy the DLL into the following directories:

- Report Designer Folder: `[Program Files]Microsoft Visual Studio 10.0\Common7\IDE \PrivateAssemblies`.
- Deploy folder `[Program Files]Microsoft SQL Server\MSSQL\Reporting Services \ReportServer\bin`.

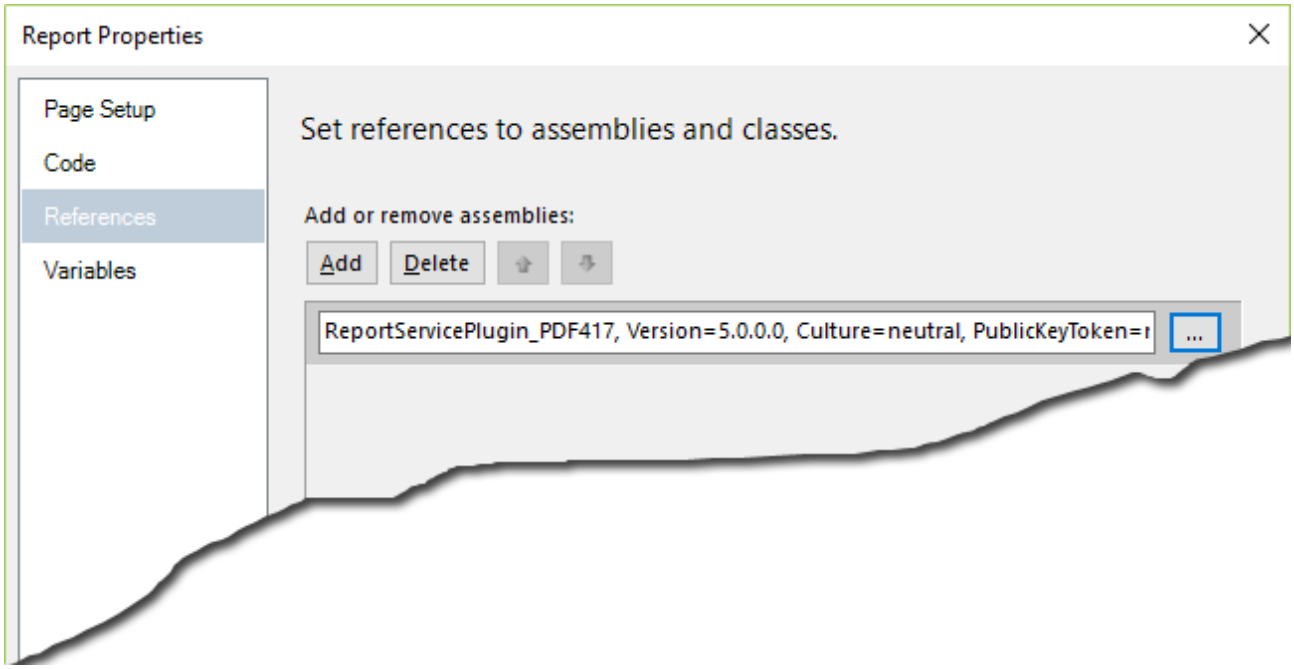
If you have multiple SQL Server instances, you will have multiple `MSSQL.n` directories. Locate the one that the Reporting Service is installed under.

If you are running Visual Studio 2008 and above, you also need to modify `RSPreviewPolicy.config` in order to preview the report. See the section "Deployment" for details on the step.

Adding PDF417 Barcodes to Report

1. Open Visual Studio or SQL Server Business Intelligence Development Studio and create a new Report Server Project.
2. In Query builder, enter SQL statement `SELECT Name, ProductNumber, ListPrice FROM Production.Product`.
3. Layout the report as desired. Make changes to the format of each field. Add a new column to the right that will hold the barcodes. The result looks like below:

4. Select Report → Report Properties to bring up Report Properties dialog.
5. click on References tab. Navigate to the location of the customer assembly and add it.

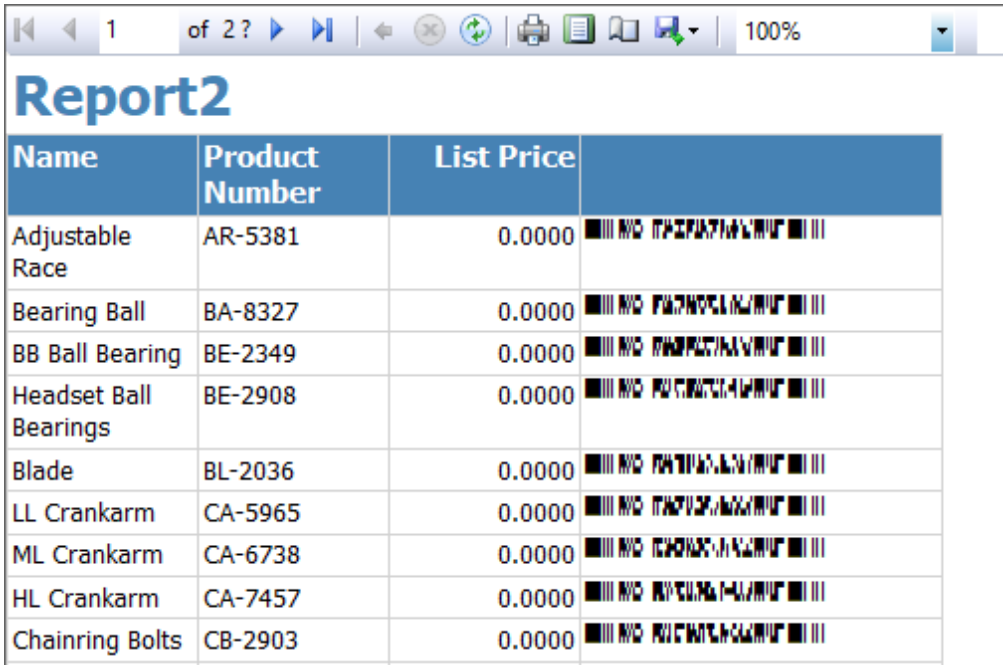


6. Click on the new column created, and select `expressions`, change its value to the formula below:

   ```
   =Morovia.ReportService.PDF417V5.PDF417Encode(Fields!ProductNumber.Value, 0, 3, 0, 0, 3.0)
   ```

   The formula calls the `PDF417Encode` function with automatic error correction level 3 columns and target aspect ratio 3.0.
7. Now preview the report. You should see lines of hexadecimal characters at the place of the barcode.

8. Format the text box with `MRV PDF417 N3`, 6 points. Preview the report again, you should see the barcodes.



If you are printing to a laser printer, you should get a good quality barcode label printed. Now if you are printing to a low resolution thermal printer, such as a Zebra with 203 dpi in resolution, you should examine if the font size produces the optimal results. Follow the steps below:

1. According to Table 2.1, "List of PDF417 fonts", `MRV PDF417 N3` produces barcodes with X dimension at 13.35 mils at 12 points. Now translate it into inches and multiply the result by the printer resolution: 13.35x0.001*203=2.71.
2. 2.71 is not a optimized value so we either round to 3, or 2. 3 will render X dimension as 3*1000/203=14.77 mils. The optimal font size therefore is 14.77*12/13.35=13.28 points.
3. Change the font size of the barcode text box to 13.28 points. Now the barcode produced will have high quality when printed on target printers.

Deployment

The Reporting Service require any custom assembly defined in the security policy otherwise a run-time error will be thrown and all you get is #Error without any explanation. Follow the steps below to change security policy. Two security policy files are required to change:

- `RSPreviewPolicy.config`. This policy file is used for `DebugLocal` preview in Visual Studio. This file is located in the Report Designer folder which is `[Program Files]Microsoft Visual Studio 8\Common7\IDE\PrivateAssemblies` for Visual Studio 2010. Locate the according directory if you are using a different version of Visual Studio.
- `rssrvpolicy.config`, the policy file used for running Report Server. The file is located under `[Program Files]Microsoft SQL Server\MSSQL\Reporting Services\ReportServer\bin` directory.

Perform the following steps to add security policy required to run custom assembly:

1. Open `RSPreviewPolicy.config`, and add the following content at the end (just before two ending `CodeGroup` tags):[1]

```
<CodeGroup class="FirstMatchCodeGroup"
  version="1"
  PermissionSetName="FullTrust"
  Name="ReportServicePlugin_PDF417.dll" Description="ReportServicePlugin_PDF417.dll">
  <IMembershipCondition class="UrlMembershipCondition" version="1"
Url="C:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\IDE" \
"PrivateAssemblies\ReportServicePlugin_PDF417.dll"/>
</CodeGroup>
```

2. Save the file. In Visual Studio, change the active configuration to `DebugLocal` and run the report. You should see the barcodes on the report. Examine the contents in the Output window.

   If you see message A first chance exception of type 'System.Security.SecurityException' occurred in mscorlib.dll, the security is not configured properly.

3. After you have successfully run the report under `DebugLocal` configuration, publish the report to the Report Server. Open `rssrvpolicy.config` and add similar lines.

```
<CodeGroup class="FirstMatchCodeGroup"
  version="1"
  PermissionSetName="FullTrust"
  Name="ReportServicePlugin_PDF417.dll" Description="ReportServicePlugin_PDF417.dll">
  <IMembershipCondition class="UrlMembershipCondition" version="1"
Url="C:\Program Files\Microsoft SQL Server\MSSQL.2\Reporting Services\" \
"ReportServer\bin\ReportServicePlugin_PDF417.dll" />
  </CodeGroup>
```

   Note that you may change the file path if it is located in a different location.

4. Restart SQL Server Reporting Service and browse the report. You should see the barcodes on the report this time.

Chapter 9. Technical Support

Morovia offers a wide variety of support services. To help you save time and money when you encounter a problem, we suggest you try to resolve the problem by following the options below in the order shown.

- Consult the documentation. The quickest answer to many questions can be found in the Morovia product documentation.
- Review the tutorial and sample applications. The tutorial steps you through the development process for a typical barcode application. The sample applications provide working code examples in several programming languages. All sample applications are extensively commented.
- Access Morovia Online. Morovia Online provides a knowledge base which documents the frequently asked questions and a web forum.

  The web address for knowledge base is http://support.morovia.com. You can ask question at support forum at http://forum.morovia.com.
- Contact Morovia Technical Support Service. The Technical Support service is provided for free up to 180 days after the purchase. Email Morovia support engineers at `support@morovia.com`.

Note If you purchased your software from our reseller, check to see if they provide support services before contacting Morovia.

Support services and policies are subject to change without notice.

Appendix A. Input Format for ECI and Macro PDF417

Extended Character Interpretation (ECI) and Macro PDF417 are optional features defined in PDF417 specification. ECI allows content of mixed encoding types (for example different languages) encoded in the symbol. Macro PDF417 defines a standard method to split large amount data into multiple symbols that can be decoded and concatenated without relying on the order of scan. Both features require support of the scanner.

The two features must be entered using the special syntax outlined below.

Extended Channel Interpretation (ECI)

ECI was introduced to allow output data stream to have interpretations different from the default character set (*ISO8859-1*). An ECI can be any number between `0` and `999999`. The tilde code sequence `~7nnnnnn` is used to enter the ECI value. The tilde code sequence can appear at any places of the input, provided that exact `6` digits follows `~7`. For example to start an interpretation , enter **`~7000010`**.

Macro PDF417

Using Macro PDF417, large amount data is split into several file segments and encoded into individual symbols. To create Macro PDF417 symbols, you need to enter the control block information using `~2` tilde code sequence. A sample input looks like this:

```
123456789012345678901~2[3][LA-CONFIDENTIAL][6][fn:part2|ts:199044|ad:Justin Power|fs:110990]
```

Syntax

The tilde code sequence for Macro PDF417 control block information is as follows:

```
~2[SI][FID][TS][fn:string|...]
```

The `~2` must appear at the end of the message. The data after the control block is ignored. The first three fields are required. The last field is optional and can contain several additional sub-fields.

Segment Index (SI)

In Macro PDF417, each symbol represents a segment of the whole file. To rebuild the whole file, the segment must be constructed in proper order. The value of segment index is 0 based. For example, for a file divided into k segments, the segment index can be any number between 0 and k-1.

The value allowed for SI is between 0 and 99.

File ID (FID)

All symbols belong to the same group have the same file ID. The File ID can be any string, such as `archive2`. Although the standard does not set a limit on the length of the File ID, keep in mind that the control block reduces the overall symbol capacity.

Total Segments (TS)

The “total segments” field is required for every symbol in the group. It should remain constant among all symbols.

Optional Fields

Macro PDF417 defines several optional fields to encode additional file information such as file name, timestamp, file size and checksum. All these fields must be at the end of the control block. If more than two optional fields are present, they should be separated with vertical bars `|`. Within a field, a colon `:` divides the name part and value part.

The acceptable field names are listed below:

Table A.1. Optional Fields in Macro PDF417

Field Designator	Field Name (abbreviated)	Field Name (normal)	Comment	Data Type
0	fn	filename	File Name	string
1	sc	segmentcount	Segment Count	number
2	ts	timestamp	Time Stamp	number
3	sd	sender	Sender	string
4	ad	addressee	Addressee	string
5	fs	filesize	File Size	number
6	cs	checksum	Checksum	number

For example, **`[fn:archive1.zip|ts:20051231|sd:user@example.com|cs:9901234]`** encodes 4 optional fields: file name (`archive1.zip`), time stamp (`20051231`), sender (`user@example.com`) and checksum (`9901234`).

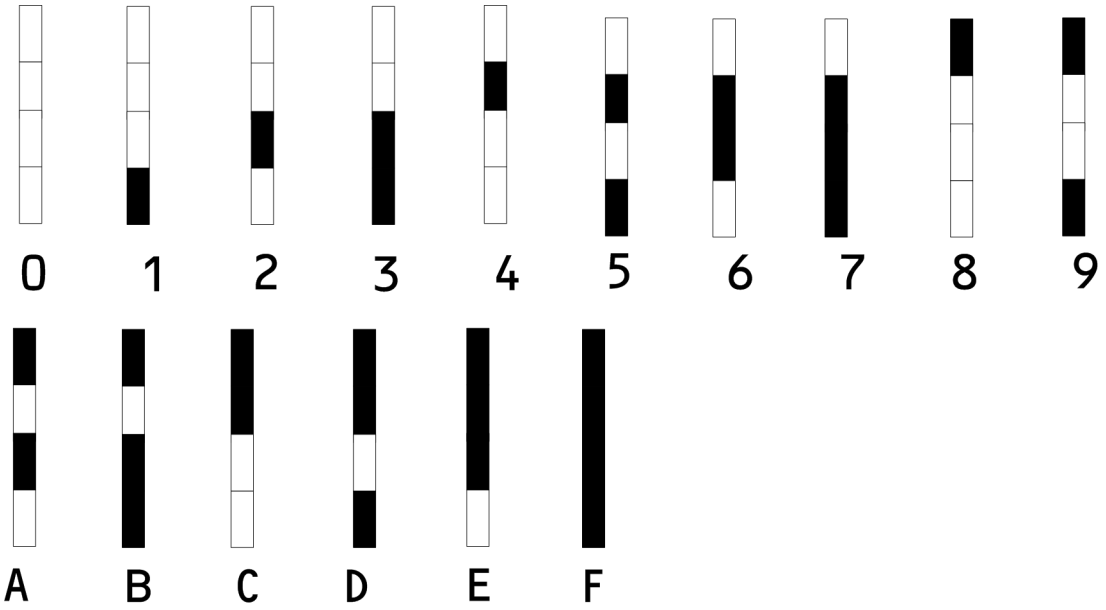
Appendix B. Font Character Set

Warning The glyph structures described here apply on PDF417 Fontware 4.0. Beginning with 4.0, the glyph structures are changed to overcome restrictions imposed by some printing software.

In some circumstances, it is desirable to substitute the font with the custom drawing routines. This section explains the mapping of each character in the font.

The basic element in a PDF417 barcode is a rectangle cell called *module*, in either black or white. If "1" represents a black module and "0" represents a white module, we can use an array of strings to represent a PDF417 barcode. The result is further compressed by combining adjacent cells into a character. In a PDF417 font (version 4.0), four vertical modules are merged into one character, as illustrated below:

Figure B.1. PDF417 Font (V4) Character Set



The PDF417 font uses 16 characters (0-9, A-F) to represent 16 possible scenarios. To draw a PDF417 barcode from the encoding results, you have two options:

- Convert the encoding results to an array of string consisting of 1 and 0 only, and draw cells from the left to the right, and from the top to the bottom.
- Create a drawing routine to draw these 16 glyphs. Scan the encoding results and call the routine one by one.

When designing drawing routines, you may also consider joining adjacent dark cells into polygons to reduce the number of drawing commands.

> **Note** Be ware that the cell is a rectangle instead of a square. The height of a cell is yHeight times of the width. If you are writing bitmaps from the barcode string, make sure to round the height into integral times of pixels and the height of all cells should be consistents in the whole symbol.

For the list of yHeight and X dimensions of the PDF417 fonts, see Table 2.1, “List of PDF417 fonts”.

Appendix C. Fontware License Agreement

By using or installing font software (referred as "Fontware" and "SOFTWARE" in this agreement, including fonts, components, source code, install program etc.) created by Morovia Corporation (referred as "Morovia" below), you (or you on behalf of your employer) are agreeing to be bound by the terms and conditions of this License Agreement. This License Agreement constitutes the complete agreement between you and Morovia. If you do not agree to the terms and condition of the agreement, discontinue use of the Fontware immediately.

License Grant

Number of Installation or Users: In consideration for the license fee paid, Morovia grants to you only, the License, the non-exclusive, non-transferable right to use the font in accordance with the license you purchase. If you are using this product for your employer, this agreement also includes your employer. You may only use the font on computers (CPUs) for which the Fontware is licensed.

The Single User License allows an individual to use the license Fontware on 1 CPU in your organization connected to any number of printers or other image producing devices. If you install or use the fonts on more than one CPU in your organization, or multiple individuals access the barcode printing functionality (e.g. printing from a printer server), multiple single user licenses must be purchased.

The Corporate License allows unlimited use of the licensed fonts in the organization that purchases it. After the Corporate License is purchased, the fonts may be installed and used on multiple CPUs in that organization without any additional fees to be paid to Morovia.

The Developer License allows 1 developer to install the fonts within his organization (Corporate License) as well as rent, lease or distribute the licensed fonts bundled with an application up to 10,000 users (formerly referred as Distribution License). The developer may not resell, rent, lease or distribute the fonts alone, they must be bundled with an application or with the application installation files. The developer may not resell, rent, lease or distribute the fonts in any way that would directly compete with Morovia. If use exceeds 10,000 concurrent users or installations, additional Developer License is required.

Copyright

All title and intellectual property rights in and to the Software Product (including but not limited to any images, photographs, animations, video, audio, music, text, and "applets" incorporated into the Software Product), the accompanying printed materials, and any copies of the Software Product are owned by Morovia. All title and intellectual property rights in and to the content that is not contained in the Software Product, but may be accessed through use of the Software Product, is the property of the respective content owners and may be protected by applicable copyright or other intellectual property laws and treaties. This Agreement grants you no rights to use such content. If this Software Product contains documentation that is provided only in electronic form, you may print one copy of such electronic documentation. You may not copy the printed materials accompanying the Software Product.

Distribution Limits

You must own a Developer License to distribute fonts outside your organization. You are allowed to distribute the software inside or outside your organization for up to 10,000 copies. When you distribute the software, you adhere to the following terms: (a) You may not resell, rent, lease or distribute the Software alone. The Software must be distributed as a component of an application and bundled with an application or with the application's installation files. The Software may only be used as part of, and in connection with, the bundled application. (b) You may not resell, rent, lease or distribute Software in any way that would compete with Morovia. (c) You must include the following MOROVIA copyright notice in your Developed Software documentation and/or in the "About Box" of your Developed Software, and wherever the copyright rights notice is located in the Developed Software ("Portions Copyright (c) Morovia Corporation 2004. All Rights

Reserved."). (d) You agree to indemnify, hold harmless, and defend MOROVIA, its suppliers and resellers, from and against any claims or lawsuits, including attorney's fees that may arise from the use or distribution of your Developed Software. (e) you may use the SOFTWARE only to create Developed Software that is significantly different than the SOFTWARE. (f) You must use font embedding technology when you use the SOFTWARE in PDF and WORD documents; (g) You must specify the exact domain name when creating embedded fonts for web pages. (h) All GUI programs coming with the font package are non-distributable.

Termination

This Agreement is effective until terminated. This Agreement will terminate automatically without notice from Morovia if you fail to comply with any provision contained here. Upon termination, you must destroy the written materials, the Morovia product, and all copies of them, in part and in whole, including modified copies, if any.

Warranty & Risks

The fonts provided by Morovia are licensed to you as is and without warranties as to performance of merchantability, fitness for a particular purpose or any other warranties whether expressed or implied. You, your organization and all users of the font, assume all risks when using it. Morovia shall not be liable for any consequential, incidental, or special damages arising out of the use of or inability to use the font or the provision of or failure to provide support services, even if we have been advised of the possibility of such damages. In any case, the entire liability under any provision of this agreement shall be limited to the greater of the amount actually paid by you for the font or US $5.00.

Glossary

DLL	Acronym for Dynamic Link Library, a library of executable functions or data that can be used by a Windows application. A DLL can be used by several applications at the same time. Some DLLs are provided with the Windows operating system and available for any Windows application. Other DLLs are written for a particular application and are loaded with the application.
EMF	Acronym for Enhanced MetaFile. A newer 32-bit version of Windows MetaFile. EMF contains frame information and contain more drawing commands then its predecessor, WMF.
Macro PDF417	A method to link multiple PDF417 symbols together in order to encode large amount of data.
PDF417	PDF417 is a multi-row, variable-length symbology with high data capacity and error-correction capability. PDF417 has some unique features which makes it the widely used 2D symbology. A PDF417 symbol can be read by linear scanners, laser scanners or two-dimensional scanners. PDF417 is capable of encoding more than 1100 bytes, 1800 text characters or 2710 digits. Large data files can be encoded into a series of linked PDF417 symbols using a standard methodology referred to as Macro PDF417.
PNG	Acronym for Portable Network Graphics. PNG is a bitmap image format that employs lossless data compression.
RTF	Acronym for Rich Text Format. A document file developed by Microsoft since 1987 for cross-platform document interchange. Most word processors are able to read and write RTF documents.

Index

A

M

P

S

T