

MakeHDBarcode: The HD Barcode Generator

This document explains the operation of MakeHDBarcode. This program is a server which generates HD Barcodes suitable for printing or for displaying on a screen.

The zip file contains the following:

MakeHDBarcode.pdf	This document
MakeHDBarcode	64-bit Linux executable which implements the server
SerialTest.exe	Windows executable which demonstrates interfacing
SerialTestSource	Folder containing the source code for SerialTest
vcredist_2015_x86.exe	Visual Studio 2015 Redistributable

The zip file does NOT contain the file MakeHDBarcode.ini. This file will be emailed separately. It will contain an authorization key which is unique to your company name.

To run a basic test, you need two computers. One computer must be running a 64-bit Linux operating system. On that computer, create a folder and place MakeHDBarcode (Linux executable) and MakeHDBarcode.ini (configuration file) in that folder. Either change the firewall to allow connections to port 7777 (the default), or change MakeHDBarcode.ini to specify another port and make sure that port is accessible. Do not change the order of lines or other information in MakeHDBarcode.ini or the server will not operate.

The other computer must be running a version of Microsoft Windows. On that computer, make sure you install the Visual Studio 2015 runtime if needed. Then create a folder and place SerialText.exe (Windows executable) in that folder. Open a command prompt and run SerialDemo.exe with two arguments: the IP address of the Linux computer, and the port to use.

If successful, this will create 201 barcodes. This is described in more detail on the following page.

Anti-Reverse Engineering

In order to protect our intellectual property, the MakeHDBarcode executable includes multiple techniques to make reverse engineering extremely difficult. The program makes extensive use of self-modifying code. It must be able to obtain permission to change itself or it will not function. It requires a 64-bit Intel processor.

If any change is made to the program no matter how insignificant, it will cease to operate.

Many of the techniques used are similar to those used by malware, so it is possible that a program designed to detect malware might be fooled into thinking there is a problem with this program when in fact there is none. Fortunately, this type of issue is more likely on Windows computers which is just one reason the server is compiled for Linux and not for Windows.

Serial Demo

This program demonstrates interfacing with the MakeHDBBarcode server.

You must run the program from a directory which has write permissions, and you must specify two arguments:

- The IP address of the server
- The port number (specified in MakeHDBBarcode.ini)

The program creates 201 different HD Barcodes:

- 100 serialized public HD barcodes (readable by anyone with the reader app)
- 100 serialized company private HD Barcodes (can't be read without authorization)
- 1 temporary universal authorization HD Barcode

Each barcode created is a monochrome BMP file.

The 100 public HD Barcodes are encoded with "ABCDE00001" through "ABCDE00100". They are stored in a folder named Public.

The 100 private HD Barcode are encoded with "XYZZY00001" through "XYZZY00100". They are stored in a folder named Private.

The single universal authorization HD Barcode is stored in a folder named Authorization.

Please refer to the source code for more information.

This program is written in Microsoft Visual Studio 2015, but is easily adaptable to other operating systems such as Linux or OS X. Before running SerialDemo.exe, you may need to install the Visual C 2015 Runtime package which is included.

*Note: This software was intended to be a very clear illustration of how to interface with the MakeHDBBarcode server. It is **NOT** intended to be a model of good programming practices. All unexpected errors are fatal and the program exits without cleaning up properly.*

MakeHDBarcode API (Application Program Interface)

TCP Packet Protocol Layer

Because TCP is a “stream” protocol and not a “packet” protocol, a simple protocol layer is needed around packets to allow them to be received in pieces but then reassembled into a single packet. There are two special bytes used for this:

0xAA	is the EndPacketByte
0xAB	is the EscPacketByte

Every packet must end with a 0xAA byte. If the packet data itself includes 0xAA and/or 0xAB bytes, then each of these must be replaced with the EscPacketByte (0xAB) followed by the inverse of the desired character. In other words:

0xAA	→	0xAB followed by 0x55
0xAB	→	0xAB followed by 0x54

The source code for the included SerialDemo program can be consulted as a reference.

MakeHDBarcode Command Codes

The first byte of the packet is the command code. The following command codes are implemented:

- 0x11: Set Geometry
- 0x21: Set Text
- 0x22: Append Text
- 0x31: Set Binary
- 0x32: Append Binary
- 0x41: Set Secret Phrase
- 0x42: Set Identification Tag
- 0x51: Generate Public HD Barcode
- 0x52: Generate Company Private HD Barcode
- 0x53: Generate Permanent Authorization HD Barcode
- 0x54: Generate Temporary Authorization HD Barcode
- 0x61: Retrieve First Line of BMP
- 0x62: Retrieve Next Line of BMP

The response always consists of two or more bytes.

The first byte of the response duplicates the command code.

The second byte of the response is the error code. A 0x00 means no error.

Bytes beyond the second byte only occur where there is no error. For the 0x5x series, there are three bytes after the first byte which contain the actual number of rows, columns, and error correction used. For the 0x6x series, the bytes after the first two contain binary data which is used to build a bitmap of the HD Barcode.

0x11: Set Geometry

This command code is followed by 6 byte values.

Value #1 is the number of rows. This must be an even number between 2 and 254. 0 is also allowed, which means the server can automatically select a value. 1 is also allowed, but only if the number of columns is one of the following values: 4, 8, 12, or 16.

Value #2 is the number of columns. This must be an even number between 2 and 254. 0 is also allowed, which means the server can automatically select a value. 1 is also allowed, but only if the number of rows is one of the following values: 4, 8, 12, or 16.

Value #3 is the amount of error correction. This must be a number between 2 and 17. This is the number of errors that can be corrected in each group of 48 bytes. 0 is also allowed, which means the server can automatically select a value.

Value #4 is the grid size, which is the number of pixels between adjacent parallel lines or dots. This must be a number between 3 and 32. 0 is not allowed – the grid size must be specified.

Value #5 is the style. Use 1 for lines, or use 2 for dots. No other values including 0 are allowed.

Value #6 is the line thickness or dot size. It must be a number between 1 and 8. 0 is not allowed – the thickness must be specified. The thickness should be no more than half of the grid size, but the server does not enforce this.

0x21: Set Text

This command is used to define the text portion to encode in the HD Barcode. Any previously set text is overwritten with this information.

This command may be sent as a single byte without any text information following the command code. In this case, the text data is removed from the next HD Barcode created.

0x22: Append Text

This command is identical to the 0x21 command, except it adds to any existing text data.

Multiple 0x22 commands may be used to generate exceptionally long text strings, but it is usually more efficient to embed a ZIP file containing an HTML representation of the text.

0x31: Set Binary

This command sets the binary portion of the HD Barcode. Only 3 formats are supported:

JPEG	(first two bytes are 0xFF and 0xD8)
PNG	(first two bytes are 0x89 and 0x50)
ZIP	(first two bytes are 0x50 and 0x4B)

If the first two bytes are not one of these three combinations, a 0x31: File Type Not Valid error will result.

This command may be sent as a single byte without any binary information. In this case, the binary data is completely removed from the next HD Barcode created.

0x32: Append Binary

This command adds to the binary portion of the HD Barcode. A previous 0x31 command must have been issued. Otherwise, a 0x31: File Type Not Valid error will result.

0x41: Set Secret Phrase

The secret phrase is used to implement encryption. Company private HD Barcodes are encrypted using the secret phrase. Authorization HD Barcodes embed the secret phrase, and are used to communicate the encryption key.

The secret phrase is just that – secret. Without knowing the secret phrase, it is not possible to read the HD Barcode. Even we here at HD Barcode, LLC cannot read the code!

Each company may have multiple secret phrases, and control who is authorized to read HD Barcodes created with each phrase. To help keep track of which readers are authorized, there is also something called an identification tag. The identification tag is paired with a secret phrase. The identification tag appears on the screens of authorized readers, but it has no value when it comes to decrypting codes – only the secret phrase matters. The secret phrase does not appear on the screen of authorized readers.

In theory, you could reuse the same identification tag for more than one secret phrase, but then this makes it next to impossible to know who is authorized and who is not.

0x42: Set Identification Tag

The identification tag is used in authorization HD Barcodes to identify the codes it will authorize. It is NOT part of the encryption process.

0x51: Generate Public HD Barcode

This command generates a public HD Barcode using previously sent information. Public means that everyone is capable of reading the code.

The command consists of a single byte. No additional bytes are allowed. If there is no error, the response consists of 5 bytes:

0x51	Duplicates the command code
0x00	Indicates no error
xx	Indicates the actual number of rows used
yy	Indicates the actual number of columns used
zz	Indicates the actual error correction used

0x52: Generate Company Private HD Barcode

This command is exactly like the 0x51 command, except that it generates a company private HD Barcode using previously sent information. A company private HD Barcode can only be read by authorized readers. This authorization comes from a different HD Barcode called an authorization HD Barcode.

0x53: Generate Permanent Authorization HD Barcode

This command is exactly like the 0x51 command, except that it generates an authorization HD Barcode. Such a code contains a secret phrase, an identification tag, and an expiration date. The expiration date will be the same as the one in the server – you cannot alter it.

Permanent means once a reader learns this authorization, it keeps it as long as the app is installed.

If the command code is the only byte in the packet, then this creates a “Universal” authorization HD barcode, which means anyone can read it and become authorized.

If there is a text string after the command code, and if that string is in the form #####-#####-#####, then this creates a “Unique” authorization HD Barcode. It is only be readable by an app whose reader serial number matches the number provided.

If there is a text string after the command code which is not a reader serial number, then this creates a “Cascaded” Authorization HD Barcode. A cascaded code uses two secret phrases. In order to read the code generated, the reader must have been previously authorized for the secret phrase sent in this 0x53 command. If they are capable of reading the code, then they will become authorized for the new secret phrase and identification tags which were sent in earlier 0x41 and 0x42 commands.

0x54: Generate Temporary Authorization HD Barcode

This command is exactly like the 0x53 command, except that the code generated is temporary and not permanent. This means that it is forgotten every time the user exits the reader app. In order to re-authorize reading of private codes, this code must be scanned again.

0x61: Retrieve Start of BMP

This retrieves the BMP file header. The header is always 62 bytes long and uses the following format:

File identifier	2 bytes	'B' 'M'	
Size of BMP file	4 bytes	s3 s2 s1 s0	
Reserved	4 bytes	00 00 00 00	
Offset to pixel data	4 bytes	3E 00 00 00	
Header size	4 bytes	28 00 00 00	
Image width in pixels	4 bytes	w3 w2 w1 w0	
Image height in pixels	4 bytes	h3 h2 h1 h0	
Number of planes	2 bytes	01 00	
Bits per pixel	2 bytes	01 00	
Compression	4 bytes	00 00 00 00	
Image size	4 bytes	i3 i2 i1 i0	
XPelsPerMeter	4 bytes	46 5C 00 00	(600 DPI)
YPelsPerMeter	4 bytes	46 5C 00 00	(600 DPI)
Color map entries used	4 bytes	02 00 00 00	
Important colors	4 bytes	00 00 00 00	
'0' pixel color	4 bytes	FF FF FF 00	(white)
'1' pixel color	4 bytes	00 00 00 00	(black)

If there is no error, the response is actually 64 bytes long, because the first byte is the command code (0x61) and the next byte is 0x00 indicating no error.

0x62: Retrieve Next Line of BMP

This retrieves the next line of the BMP file. The number of bytes will be equal to the image width in pixels divided by 8 rounded up to the next multiple of 4. A 1 bit indicates black, while a 0 bit indicates white.

Since every HD Barcode has a top and bottom border, the first few lines and the last few lines will be all 0's to indicate a completely white line. Since there is also a left and right border, the first few bits and the last few bits of every line will also be 0's to indicate a white edge.

If there are no more lines to send, the error code 0x62: No More Lines is sent.

Text Strings

The following commands accept text data after the command code:

- 0x21: Set Text
- 0x22: Append Text
- 0x41: Set Secret Phrase
- 0x42: Set Identification Tag
- 0x53: Generate Authorization HD Barcode

For these commands, the text which appears after the command code may not contain embedded null (0x00) characters. If UTF-8 is being used, then it must use one of the following three formats:

U+0000 to U+007F	0xxxxxxx
U+0080 to U+07FF	110xxxxx 10xxxxxx
U+0800 to U+FFFF	1110xxxx 10xxxxxx 10xxxxxx

MakeHDBarcode Error Codes

The error code will always be in the 2nd byte of the response. The first byte will be the command code from the request.

0x00: No Error

No errors were detected. Depending on the command, additional bytes may appear after the error code.

0x01: Command Code Not Valid

The command code (first byte) is either invalid, or not yet implemented.

0x02: Too Little Packet Data

The packet was too short for the command code. Some required data was missing.

0x03: Too Much Packet Data

The packet was too long for the command code. Extra data was not understood.

0x04: Invalid Text String

The text string contained embedded null characters (0x00), or characters outside of the supported UTF-8 range (0xF0 to 0xFF).

0x05: Out Of Memory

The computer was unable to obtain memory to process the packet.

0x06: Authorization Key Card Has Expired

The server's authorization key has expired.

0x11: Rows Not Valid

The value for rows must be an even number between 2 and 254. 0 is also allowed, which means the server can automatically select a value. 1 is also allowed, but only if the number of columns is one of the following values: 4, 8, 12, or 16.

0x12: Columns Not Valid

The value for columns must be an even number between 2 and 254. 0 is also allowed, which means the server can automatically select a value. 1 is also allowed, but only if the number of rows is one of the following values: 4, 8, 12, or 16.

0x13: Error Correction Not Valid

The value for error correction must be a number between 2 and 17. 0 is also allowed, which means the server can automatically select a value.

0x14: Grid Size Not Valid

The value for grid size must be a number between 3 and 32. 0 is not allowed.

0x15: Style Not Valid

The value for style must be 1 for lines, or 2 for dots.

0x16: Thickness Not Valid

The value for thickness must be a number between 1 and 8. Thickness is not allowed to be more than half the grid size.

0x31: File Type Not Valid

The first two bytes of the binary file determine its type, which must be JPEG, PNG, or ZIP.

0x51: Geometry Not Set

A valid 0x11 command is needed before an HD Barcode can be generated.

0x52: Data To Encode Not Set

Either text data, or binary data, or both must be set prior to creating an HD Barcode.

0x53: Secret Phrase Not Set

Company private HD Barcodes require a secret phrase to be set.

0x54: Too Much Data For Geometry

There is too much text and/or binary data to fit in the requested geometry.

0x61: Bitmap Not Set

The BMP file cannot be retrieve until an HD Barcode is generated.

0x62: No More Lines

All of the lines in the BMP have been transmitted.