General page information

aspEasyBarCode

Last modified : 30/07/2004 (Version 3.10)

What's aspEasyBarCode

The **aspEasyBarCode** is a product that allows internet developers to add bar codes on their applications. It's composed of two technologies to choice, one for users that uses IIS (Internet Information Server from Microsoft) or other development languages, like visual basic, and another technology, the CGI binary that allows you to use it on standard web servers, like Apache, Netscape, IIS, etc. (Always talking on windows technology).

The ASP module is a dll that you register on the server/client and you get the ability to use all the function to draw bar codes on your active server page or even on the Visual Basic application.

The CGI-BIN is an exe module that allows you to send some parameters from your html page and returns the image generated of the barcode you asked for. As easy as typing on your html page: **<img source="/cgi-bin/easybarcode.exe?text=1234&typebar=3">** and you get the image barcode from 1234, without generating files on the server.

As all of our aspEasy family programs this is also shareware without limitation on functionality but as the others it has the lack of a cross line on the generated bar. If you register it then we will send you the same program with out cross.

It supports the most standards code bars with checksums and it outputs directly to a bitmap or a standard gif format (asp module). Print it directly to your printer and do standard reports in html with out the need of other expensive reports designers.

This component is **shareware** , the only limitation is when creating the bitmap it will display a cross line on the barcode picture, if you want to remove it then you will have to register it for 50 US$. There is no limitation on functionality and time on the shareware version.

Contact us at: support@mitdata.com or visit us at: www.mitdata.com

- aspEasyBarCode -

If you want to test it how easy is to use it see this fast guide:

The main properties to known are two, Text and TypeBar , with **text** you input the code and with the **Typebar** you select the bar code type to render.
After setting the properties you should draw the the barcode, with any programming language you will use the DrawToFile function, with ASP you should use the BinaryWrite , but you can also use the DrawToFile function.

Want to get this bar code:

Example in Visual Basic:

```
set BC = createObject ( "
easyBarCode.aspBarCode ")
BC.TypeBar = 14 ' Bar type for csCodeEAN13
BC.Text = " 0123456789123 "
BC.DrawToFile " barcode.gif "
Set BC = Nothing
```

Example in ASP:

```
<%
Response.Buffer = TRUE
Response.ContentType = "image/gif"
set BC = createObject ( "
easyBarCode.aspBarCode ")
BC.TypeBar = 14 ' Bar type for csCodeEAN13
BC.Text = " 0123456789123 "
BC.BinaryWrite
%>
```

Barcodes types

Here are some barcodes specification taken from Internet and reproduce it as it.

Please visit http://www.adams1.com/pub/russadam/info.html to get more information, some parts of this section has been extracted from this site.

UCC/EAN 128

The purpose of UCC/EAN-128 is to establish a standard way of labeling a package with more information than just a product code. It provides supplemental information such as batch number and "use before" dates

There are two main components of UCC/EAN-128: the data with its Application Identifier and the bar code symbology used to code the data. The bar code symbology is code 128. The diference is the use of defined Application Identifiers with data encoded into the code 128 symbol.

UCC/EAN APPLICATION IDENTIFIER

An Application Identifier is a prefix code used to identify the meaning and the format of the data that follows it (data field).

There are AIs for i *dentification* , *traceability* , *dates* , *quantity* , *measurements* , *locations* , and many other types of information.

For example, the AI for *batch number* is **10** , and the *batch number* AI is always followed by an alphanumeric batch code not to exceed 20-characters.

The UCC/EAN Application Identifiers provide an open standard which can be used and understood by all companies in the trading chain, regardless of the company that originally issued the codes.

Identification numbers

These AIs contain data to identify:

- An **article** , the data field includes the EAN-8, EAN-13 or EAN-14 item identification number. The AI for an article is **01** plus up to a 14-digit number.
- A **logistic unit** , which must be given a unique serial number called the Serial Shipping Container Code (SSCC). It provides companies with a facility to identify logistic units (pallets,drums, rolls) for tracking and tracing purposes. The SSCC AI is **00** followed by a 18-digit number identfying the individual transport packages. This code can be used for tracking the packages and used for recept operations.
- A **returnable asset** , the data field includes the EAN-13 number of the asset plus an optional serial number.

Traceability Numbers and Dates

These AIs allow data to be encoded that provides traceability of individual products or despatch units throughout the supply chain. This category includes Batch/Lot Number, Serial Number, Production Date, Minimum Durability Date and Maximum Durability Date. For the **Year 2000 issue** , see the EAN International FAQ . The answer to the 2-digit limitation given by EAN International is that it is the application software's responsibity to figure out that "080525" means "25th May 2008" rather than "25th May 1908". The solution they give in the FAQ is a technique called "windowing" where any two digit number between 51 and 99 is considered preceeded by "19" while any number between 00 and 50 is consided preceeded by "20". News media have reported that there is a US Patent on "windowing".

Measurements and Quantities

These AIs allow quantities and trade measurements to be encoded for items which may vary in content (quantity,

length, weight, etc.) and logistical measurements for warehouse space management systems and transportation services. An example is net weight in kilograms. The AI is **3100** to **3109** with the last digit indicating the placement of the decimal point in the following data. The following data is a 6-digit number.

Transaction References and Location Numbers

These AIs allow data to be encoded that provide transaction references and location numbers facilitating the delivery, order, and invoice reconciliation process. They can also identify shipping origin, and help to sort shipments. These AIs include Customer's Purchase Order Number, Bill to (invoice to) Location Number, and Ship to (deliver to) postal code. The "Ship To" AI, for example, is **420** followed by up to 20 alphanumeric characters of data.

The following is a partial list of other AIs. The "Content" column is a description of the data to be encoded. The "AI" column is the Application Identifier number. The "Data Structure" column is the structure of the data that *follows* the AI number. Please note that some AIs are two digits long while others are three or four digits long.

UCC APPLICATION IDENTIFIERS

Data Content	AI	Plus The Following Data Structure
Serial Shipping Container Code	00	exactly 18 digits
Shipping Container Code	01	exactly 14 digits
Batch NumberS	10	up to 20 alphanumerics
Production Date (YYMMDD)	11	exactly 6 digits
Packaging Date (YYMMDD)	13	exactly 6 digits
Sell By Date (YYMMDD)	15	exactly 6 digits
Expiration Date (YYMMDD)	17	exactly 6 digits
Product Variant	20	exactly 2 digits
Serial Number	21	up to 20 alphanumerics
HIBCC Quantity, Date, Batch and Link	22	up to 29 alphanumerics
Lot Number	23*	up to 19 alphanumerics
Quantity Each	30	
Net Weight (Kilograms)	310**	exactly 6 digits
Length, Meters	311**	exactly 6 digits
Width or Diameter (Meters)	312**	exactly 6 digits
Depths (Meters)	313**	exactly 6 digits
Area (Sq. Meters)	314**	exactly 6 digits
Volume (Liters)	315**	exactly 6 digits
Volume (Cubic Meters)	316**	exactly 6 digits
Net Weight (Pounds)	320**	exactly 6 digits
Customer PO Number	400	up to 29 alphanumerics
Ship To (Deliver To) Location Code using EAN 13 or DUNS Number with leading zeros	410	exactly 13 digits
Bill To (Invoice To) Location Code using EAN 13 or DUNS Number with leading zeros	411	exactly 13 digits
Purchase from	412	exactly 13 digits
Ship To (Deliver To) Postal Code within single postal authority	420	up to 9 alphanumerics
Ship To (Deliver To) Postal Code with 3-digit ISO Country Code Prefix	421	3 digits plus up to 9 alphanumerics
Roll Products - width, length, core diameter, direction and splices	8001	exactly 14 digits
Electronic Serial number for cellular mobile phone	8002	up ro 20 alphanumerics

For date fields that only need to indicate a year and month, the day field is set to "00".
* Plus one digit for length indication.
** Plus one digit for decimal point indication.

THE UCC/EAN-128 SYMBOLOGY

The symbology specified for the representation of Application Identifier data is UCC/EAN-128, a variant of Code 128, exclusively reserved to EAN International and the Uniform Code Council (UCC). It is not intended to be used for data to be scanned at the point of sales in retail outlets.

UCC/EAN-128 offers several advantages. It is one of the most complete, alphanumeric, one-dimensional symbologies available today. The use of three different character sets (A, B and C), facilitates the encoding of the full 128 ASCII character set. Code 128 is one of the most compact linear bar code symbologies. Character set C enables numeric data to be represented in a double density mode. In this mode, two digits are represented by only one symbol character saving valuable space. The code is concatenatable. That means that multiple AIs and their fields may be combined into a single bar code. The code is also very reliable. Code 128 symbols use two independent self-checking features which improves printing and scanning reliability.

UCC/EAN-128 bar codes always contain a special non-data character known as function 1 (FNC 1), which follows the start character of the bar code. It enables scanners and processing software to auto-discriminate between UCC/EAN-128 and other bar code symbologies, and subsequently only process relevant data.

The UCC/EAN-128 bar code is made up of a leading quiet zone, a Code 128 start character A, B, or C, a FNC 1 character, Data (Application Identifier plus data field), a symbol check character, a stop character, and a trailing quiet zone.

PDF417 - Portable data file 417

PDF417 is a stacked symbology and was invented by Ynjiun Wang in 1991 at Symbol Technologies. PDF stands for Portable Data File, and the symbology consists of 17 modules each containing 4 bars and spaces (thus the number "417"). The code is in the public domain.

The structure of the code allows for between 1000 to 2000 characters per symbol with an information density of between 100 and 340 characters. Each symbol has a start and stop bar group that extends the height of the symbol.

A PDF417 symbol can be read with modified handheld laser or CCD scanners. High density printers (thermal transfer or laser) should be used to print the symbol.

Code 128.

Code 128 is a very high density alphanumeric bar code. The symbol can be as long as necessary to store the encoded data. It is designed to encode all 128 ASCII characters, and will use the least amount of space for data of 6 characters or more of any 1-D symbology.

Each data character encoded in a Code 128 symbol is made up of 11 black or white modules. The stop character, however, is made up of 13 modules. Three bars and three spaces are formed out of these 11 modules. Bar and spaces can vary between 1 and 4 modules wide.

The symbol includes a quiet zone (10 x-dimensions), a start character, the encoded data, a check character, the stop character, and a trailing quiet zone (10 x-dimensions).

There are 106 different 3 bar/3space combinations. Each of the 106 combinations can be assigned one of three different character set meanings. These meanings are set by using one of three different start characters. Start Code A allows encoding all the standard alphanumeric keyboard characters plus control characters and special characters. Start Code B includes all standard alphanumeric keyboard characters plus lower case alpha and special characters. Start Code C includes a set of 100 digit pairs from 00 to 99 and can be used to double the density of encoding numeric-only data.

Within a symbol, one can shift between code sets by using the special character CODE and SHIFT. The CODE character shifts the code for all subsequent characters to the specified code set. The SHIFT character just changes the next character and only changes between Code Set A and Code Set B or the reverse.

The FNC codes define instructions for the bar code reader. FNC 1 is reserved for future use. FNC 2 tells the reader to store the data read and transmit it with the next symbol. FNC 3 is reserved for code reader initializing and other code reader functions. FNC 4 is reserved for future applications.

Each character has a value ranging from 0 to 105. This value is used to calculate the check character for each symbol.

The check character is a Modulus 103 Checksum that is calculated by summing the start code value plus the product of each character position (most significant character position equals 1) and the character value of the character at that position. This sum is divided by 103. The remainder of the answer is the value of the Check Character. Every encoded character is included except the Stop and Check Character.

Example: BarCode 1 Message : Start B B a r C o d e 1 Value 104 34 65 82 35 79 68 69 0 17 Position: - 1 2 3 4 5 6 7 8 9 Calculate Total: 104 + (34x1) + (65x2) + (82x3) + (35x4) + (79x5) + (68x6) + (69x7) + (0x8) + (17x9) = 2093 2093/103 = 20 remainder 33 33 = A Final message: (Start B)BarCode 1(A)(STOP)

The height of the bars must be at least .15 times the symbol's length or .25 inches, whichever is larger. The overall length of the symbol is given by the equation:

L = (11C + 35)X (alphanumeric) **L = (5.5C + 35)X** (numeric only using Code C)

where

L = length of symbol (not counting quiet zone) C = number of data characters, code characters and shift characters (do not include start, stop or checksum. They are automatically added in.) X = X-dimension

Interleaved 2 of 5

Interleaved 2 of 5 is a a numbers-only bar code. The symbol can be as long as necessary to store the encoded data. The code is a high density code that can hold up to 18 digits per inch when printed using a 7.5 mil X dimension. A check digit is optional.

The "Interleaved" part of the name comes from the fact that a digit is encoded in the bars and the next digit is encoded in the spaces. The encoded digits are "Interleaved" together. There are five bars, two of which are wide and five spaces, two of which are wide.

The symbol includes a quiet zone, the start character (narrow bar-narrow space- narrow bar-narrow space), the encoded data, the stop character (Wide bar-narrow space-narrow bar), and a trailing quiet zone.

Number	Pattern
0	NNWWN
1	WNNNW
2	NWNNW
3	WWNNN
4	NNWNW
5	WNWNN
6	NWWNN
7	NNNWW
8	WNNWN
9	NWNWN

The X-dimension is the width of the smallest element in a bar code symbol. The minimum X-dimension for an "open system" (a bar code label that will be read by scanners from outside your company) is 7.5 mils (a mil is 1/1000 inch) or 0.19 mm. The "wide" element is a multiple of the "narrow" element and this multiple must remain the same throughout the symbol. This multiple can range between 2.0 and 3.0 **if** the narrow element is greater than 20 mils. If the narrow element is less than 20 mils, the ratio must exceed 2.2. Quiet zones must be at least 10X or at least .25 inches

The height of the bars must be at least .15 times the symbol's length or .25 inches, whichever is larger. The overall length of the symbol is given by the equation:

L = (C (2N + 3) + 6 + N)X

where

L = length of symbol (not counting quiet zone) C = number of data characters X = X-dimension N = wide-to-narrow multiple

Caution About Partial Scans
A partial scan of an I 2 of 5 symbol may decode and cause incorrect data to be read. To prevent partial scans on long symbols, one should include bearer bars. These are bars that run along the top and bottom edges of the symbol in the scanning direction. If a partial scan of the symbol occurs, the scanning beam will hit the bearer bar and will not decode. The bearer bar must touch the top and bottom of all the bars and must be at least 3X wide.

Another solution for the short scanning problem is to fix all I 2 of 5 symbols to a set number of digits. Zeros can be used to pad the data to the set number of digits. The application program would then be set to only accept scans of the correct number of digits.

Finally, a check digit may be used. The I 2/5 symbology has an optional check character which uses a weighted Modulo 10 scheme. The check character is the last character in the symbol and should checked by the decoder and then transmitted with the data. Since I 2/5 must always have an even number of digits, the leftmost character may need to be a zero when the check character is added. The standard check digit is calculated by assigning alternating 3,1,3,1.. weighs to respective data digits. These weights are then multiplied by their respective data digits and the products are summed. The check digit is the digit needed to be added to the sum to make it an even multiple of 10. An example would be if the sum of the products was 37, the check digit would be 3.

Here is another explanation of how to calculate the check digit. Starting with the left most character, multiply every other character (odd positions) by 3 and sum the results with the values of the even positions. (The check character is always an even position so it is never multiplied by 3.) The sum should be a multiple of 10. To compute the check digit for printing a bar code, perform the summing operation above without the check character, do a mod 10 on the result, and

subtract the value from 10 to yield the check digit. (If you start with an even number of digits, make sure you add a leading zero before you do the calculation.)

UPC

(article taken from HowStuffWorks.com)

"UPC" stands for **Universal Product Code** . UPC bar codes were originally created to help grocery stores speed up the checkout process and keep better track of inventory, but the system quickly spread to all other retail products because it was so successful.

UPCs originate with a company called the Uniform Code Council (UCC). A manufacturer applies to the UCC for permission to enter the UPC system. The manufacturer pays an annual fee for the privilege. In return, the UCC issues the manufacturer a six-digit **manufacturer identification number** and provides guidelines on how to use it. You can see the manufacturer identification number in any standard 12-digit UPC code., like this one that comes off the back of the book "The Teenager's Guide to the Real World," published by BYG Publishing :

You can see that the UPC symbol printed on a package has two parts:

- The machine-readable bar code
- The human-readable 12-digit UPC number

BYG Publishing's manufacturer identification number is the first six digits of the UPC number -- 639382. The next five digits -- 00039 -- are the **item number** . A person employed by the manufacturer, called the **UPC coordinator** , is responsible for assigning item numbers to products, making sure the same code is not used on more than one product, retiring codes as products are removed from the product line, etc. In general, every item the manufacturer sells, as well as every size package and every repackaging of the item, needs a different item code. So a 12-ounce can of Coke needs a different item number than a 16-ounce bottle of Coke, as does a 6-pack of 12-ounce cans, a 12-pack, a 24-can case, and so on. It is the job of the UPC coordinator to keep all of these numbers straight!

The last digit of the UPC code is called a **check digit** . This digit lets the scanner determine if it scanned the number

correctly or not. Here is how the check digit is calculated for the other 11 digits, using the code 63938200039 from "The Teenager's Guide to the Real World" example shown above:

1. Add together the value of all of the digits in odd positions (digits 1, 3, 5, 7, 9 and 11).
   6 + 9 + 8 + 0 + 0 + 9 = 32

2. Multiply that number by 3.
   32 * 3 = 96

3. Add together the value of all of the digits in even positions (digits 2, 4, 6, 8 and 10).
   3 + 3 + 2 + 0 + 3 = 11

4. Add this sum to the value in step 2.
   96 + 11 = 107

5. Take the number in Step 4. To create the check digit, determine the number that, when added to the number in step 4, is a multiple of 10.
   107 + 3 = 110

   The check digit is therefore **3** .

Each time the scanner scans an item, it performs this calculation. If the check digit it calculates is different from the check digit it reads, the scanner knows that something went wrong and the item needs to be rescanned.

Widely used in the USA as a retail code.
UPC-A is a 12 digit, numeric symbology used in retail applications. UPC-A symbols consist of 11 data digits and one check digit. The first digit is a number system digit that usually represents the type of product being identified. The following 5 digits are a manufacturers code and the next 5 digits are used to identify a specific product.
UPC-E is a smaller, six digit, UPC symbology for number system 0. It is often used for small retail items. UPC-E is also called "zero suppressed" because UPC-E compresses a normal 12 digit UPC-A code into a six digit code by "suppressing" the number system digit, trailing zeros in the manufacturers code and leading zeros in the product identification part of the bar code. A seventh check digit is encoded into a parity pattern for the six main digits. UPC-E can thus be uncompressed into a standard UPC-A 12 digit number.
For UPC-E bar codes, you may enter up to 6 digits. Again, B-Coder will calculate the check digit and truncate or pad the number to a total length of 6 digits as with UPC-A.

EAN UCC SYSTEM

Code 39.

Code 39 is an alphanumeric bar code. The symbol can be as long as necessary to store the encoded data. It is designed to encode 26 uppercase letters, 10 digits and 7 special characters. It can be extended to code all 128 ASCII characters by using a two character coding scheme.

Each data character encoded in a Code 39 symbol is made up of 5 bars and 4 spaces for a total of 9 elements. Each bar or space is either "wide" or "narrow" and 3 out of the 9 elements are always wide. That's what gave the code its other name - Code 3 of 9.

The symbol includes a quiet zone (10 x-dimensions or 0.10 inches which every is greater), the start character "*", the encoded data, the stop character "*", and a trailing quiet zone (10 x-dimensions or 0.10 inches which ever is greater). The asterisk is only used as a start and stop code.

The X-dimension is the width of the smallest element in a bar code symbol. The minimum X-dimension for an "open system" (a bar code label that will be read by scanners from outside your company) is 7.5 mils (a mil is 1/1000 inch) or 0.19 mm. The "wide" element is a multiple of the "narrow" element and this multiple must remain the same throughout the symbol. This multiple can range between 2.0 and 3.0 **if** the narrow element is greater than 20 mils. If the narrow element is less than 20 mils, the multiple can only range between 2.0 and 2.2.

The height of the bars must be at least .15 times the symbol's length or .25 inches, whichever is larger. The overall length of the symbol is given by the equation:

$$L = (C + 2)(3N + 6)X + (C + 1)I$$

where
L = length of symbol (not counting quiet zone, dimension will be in mils) C = number of data characters X = X-dimension (width of the smallest element in mils.) N = wide-to-narrow multiple (use 3.0 if your code has a 3 to 1 ratio, etc.) I = intercharacter gap width

Note that the maximum value (based on the Code 39 specification) for I is 5.3X for X less than 10 mils. If X is 10 mils or greater, the value of I is 3X or 53 mils, whichever is greater. However, for good quality printers, I often equals X. I, strictly speaking, equals X-t where t is the print tolerance in mils. If you do not know the actual value for t, you can calculate the length using the maximum value for I and calculate the minimum value setting I=X.

Code 39 does not normally include a check character, however there is an established check character for applications that need it. The value of each data character is summed up and divided by 43. The remainder is the value of the character to use as the check character. The table of characters does not include the values for each character, but you can find the value by counting from the first character (a value of 0) and counting up to the last character (a value of 42).

- aspEasyBarCode -

Prices & Conditions

All products when registered have the following features:

- Free email support for all of our products
- Minor, Major upgrades and bug fixes
- Notification by email when newer versions are released

Prices and conditions

To order one product, click on the price.
We deliver orders in 24/48 hours.

Product	**Code**	**Price per/ Server**	**Site License**	Minor releases	M ajor releases	Special Prices	Bug fixes
aspEasyBarCode	34855	$50USD		**Yes**	**Yes**	**Yes**	**Yes**

Email us for special conditions and quantities prices.
Schools and Universities (Educational Services) have 35% off from regular price

The site license enables you to install it on all servers / machines from your company with out any restrictions or develop your own selling software with our component included

History

aspEasyBarCode

Last modified : 30/07/2004

History from the ASP module

3.10 (30/July/2004)
" Code128, removed some bugs on the bar code generating (The compression routine was falling)
" Added the SaveStream Method
" Added Ratio property for increasing the size of the barcode by a factor value
" Errors messages are not being truncated by a the default size width.
" Examples; Corrected the VB example and added a C# sample

3.0 (02/Jan/2004)
" Removed **Create** and **Destroy** functions, now it's called automatically.
" **Colors properties** has been renamed, please see help file to update the code.
" Fonts and size can now be set.
" Angles to 90, 180 and 270.
" Added more bar codes, including the PDF417.
" csCodeUPC_Supp2 and csCodeUPC_Supp5 has been removed to use the supplementary code on the barcode.
" Debug information now reports correctly for each function
" GIF can now be background transparent.
" Improved graphic rendering.

2.0 (28/8/2002)
" Added the BinaryWrite method
" Removed some bugs of the component.
" New license protection
" New help file with examples
" New installer

1.4 (7/9/2001)
" More options added, like colors, filename format and removed a small bug in the drawing that could do an out of memory on the IIS.
" Also the unregistered version doesn't display the unregistred label, now it displays a cross line on the picture and enables you to define the height of the code bar.

1.3 (13/2/2001)
" Added the possibility to use it on Visual Basic or whatever language that can use dll, is not dependent to the asp and the IIS. See the example in VB included on the package.

1.2 (12/12/2000)
" By numerous requests now the file output is by default in GIF format and the DrawToFile method detects by filename extension wich format to use, this reduces a lot the space used by the file graphic.

1.1 (8/12/2000)
" Added the set size property, now you can specify the size of the output bar code. See the documentation.

1.0 (25/10/2000)
" First release of the component.

History from the CGI module

1.0 (23/9/2001)
" First version released of the CGI module, incorporated major functionalities from the ASP module

" The CGI module has been tested on the IIS 5.0 environment and on the Apache 1.3.20-win32 without any problem.

Angle

Name	**Angle**
Syntax in VB	***BC* .Angle = integer**
Parameter type	**integer**
Read / Write	✓ / ✓
Default	**0**
From version:	**3.0**

Allows to rotate the barcode to a valid angle:

Value

Description	
Horizontal	
Vertical	
Horizontal Mirror	
Vertical Mirror	

Example in Visual Basic script:

```
set BC = createObject ( " easyBarCode.aspBarCode ")
BC.Display = 1 ' Display code
BC.TypeBar = 14 ' Bar type for csCodeEAN13
BC.Filetype = 1 ' GIF
BC.Angle = 90
BC.Text = " 9771210107001 "
BC.DrawToFile " barcode.gif "
if BC.Error <> 0 then WScript.Echo
BC.ErrorMessage
Set BC = Nothing
```

Results:

Color _Back

Name	**Color_Back**
Syntax in VB	***BC*** **.Color_Back = string**
Parameter type	**string**
Read / Write	/
Default	**#FFFFFF (White)**
From version:	**3.0**

You set a color attribute for the background image of the barcode. The value can be any hexadecimal number, specified according to the sRGB color space, or one of sixteen color names. Hexadecimal numbers must be prefixed by a "#" character. You can also use color names.

Note : On previous versions this property was BackColor, it has been renamed to Color_Back. If you were using the old property you should change to the new one to be compatible to new versions.

Color Name	sRGB Value
Black	#000000
Silver	#C0C0C0
Gray	#808080
White	#FFFFFF
Maroon	#800000
Red	#FF0000
Purple	#800080
Fuchsia	#FF00FF
Green	#008000
Lime	#00FF00
Olive	#0808000
Yellow	#FFFF00
Navy	#000080
Blue	#0000FF
Teal	#008080
Aqua	#00FFFFF

Example in ASP

```
<%
set BC = server.createobject (" easyBarCode.aspBarCode
")
BC.Color_Back = " silver "
```

```
BC.DrawToFile " barcode.gif "
set BC = nothing
%>
```

Result

See also

Color_Code Color_Type Color_Lines

Color_Code

Name	**Color_Code**
Syntax in VB	***BC*** **.Color_Code = string**
Parameter type	**string**
Read / Write	/
Default	**#000000 (Black)**
From version:	**3.0**

You set a color attribute for the code text image of the barcode. The value can be any hexadecimal number, specified according to the sRGB color space, or one of sixteen color names. Hexadecimal numbers must be prefixed by a "#" character. You can also use color names.

Note : On previous versions this property was TextColor, it has been renamed to Color_Code. If you were using the old property you should change to the new one to be compatible to new versions.

Color Name	sRGB Value
Black	#000000
Silver	#C0C0C0
Gray	#808080
White	#FFFFFF
Maroon	#800000
Red	#FF0000
Purple	#800080
Fuchsia	#FF00FF
Green	#008000
Lime	#00FF00
Olive	#0808000
Yellow	#FFFF00
Navy	#000080
Blue	#0000FF
Teal	#008080
Aqua	#00FFFF

Example in ASP

```
<%
set BC = server.createobject (" easyBarCode.aspBarCode
```

```
")
BC.Color_Code = " #FF0000 "
BC.DrawToFile " barcode.gif "
set BC = nothing
%>
```

Result

See also

Color_Back Color_Type Color_Lines

Color_Lines

Name	**Color_Lines**
Syntax in VB	***BC*** **.Color_Lines = string**
Parameter type	**string**
Read / Write	/
Default	**#000000 (Black)**
From version:	**3.0**

You set a color attribute for the lines of the barcode. The value can be any hexadecimal number, specified according to the sRGB color space, or one of sixteen color names. Hexadecimal numbers must be prefixed by a "#" character. You can also use color names.

Note : On previous versions this property was BarColor, it has been renamed to Color_Lines. If you were using the old property you should change to the new one to be compatible to new versions.

Color Name	sRGB Value
Black	#000000
Silver	#C0C0C0
Gray	#808080
White	#FFFFFF
Maroon	#800000
Red	#FF0000
Purple	#800080
Fuchsia	#FF00FF
Green	#008000
Lime	#00FF00
Olive	#0808000
Yellow	#FFFF00
Navy	#000080
Blue	#0000FF
Teal	#008080
Aqua	#00FFFFF

Example in ASP

```
<%
set BC = server.createobject (" easyBarCode.aspBarCode
")
BC.Color_Code = " #FF0000 "
```

```
BC.Color_Lines = " #008000 "
BC.DrawToFile " barcode.gif "
set BC = nothing
%>
```

Result

See also

Color_Back Color_Type Color_Code

Color_Type

Name	**Color_Type**
Syntax in VB	***BC*** **.Color_Type = string**
Parameter type	**string**
Read / Write	/
Default	**#000000 (Black)**
From version:	**3.0**

You set a color attribute for the type text of the barcode. The value can be any hexadecimal number, specified according to the sRGB color space, or one of sixteen color names. Hexadecimal numbers must be prefixed by a "#" character. You can also use color names.

Color Name	sRGB Value
Black	#000000
Silver	#C0C0C0
Gray	#808080
White	#FFFFFF
Maroon	#800000
Red	#FF0000
Purple	#800080
Fuchsia	#FF00FF
Green	#008000
Lime	#00FF00
Olive	#0808000
Yellow	#FFFF00
Navy	#000080
Blue	#0000FF
Teal	#008080
Aqua	#00FFFF

Example in ASP

```
<%
set BC = server.createobject (" easyBarCode.aspBarCode
")
BC.Display = 3 ' Display code and type
BC.TypeBar = 14
BC.Color_Type = " Maroon "
BC.DrawToFile " barcode.gif "
```

```
set BC = nothing
%>
```

Result

See also

Color_Back Color_Lines Color_Code

Debug

Name	**Debug**
Syntax in VB	***BC*** **.Debug = True / False**
Parameter type	**Boolean**
Read / Write	/

Sets the component in debug mode, when you made any operation then it displays the debug on the screen. Default is set to False, so there is no debug information when running up.

Example

```
<%
  BC.Debug = True

%>
```

See also

License

Display

Name	**Display**
Syntax in VB	***BC*** **.Debug = integer**
Parameter type	**Integer**
Read / Write	/

When drawing a barcode you can select if you want to display additional information inside the barcode, you can print the bar type selected and the text to rendered as a BarCode.
By default it will print the text code to be rendered on the barcode.

Constant	Definition	Image Sample
csOutputNone =0	None	
csOutputCode = 1 (default)	Only code info	
csOutputType = 2	Only type code	
csOutputBoth = 3	Both, code and type	

Example in ASP

```
<%
set BC = server.createobject (" easyBarCode.aspBarCode
")
BC.Display = 3 ' Display code and type
BC.TypeBar = 14
BC.Color_Type = " Maroon "
BC.DrawToFile " barcode.gif "
set BC = nothing
%>
```

Result

Name	**Error**
Syntax in VB	***integer* = *BC* .Error**
Parameter type	**Integer**
Read / Write	/
From version	**3.0**

Return if there was an error when drawing the barcode, 0 means no error, other values are error codes.
Attention the graphic will be created, but with coded text with the error and without the barcode if there is an error.

Example in Visual Basic script:

```
set BC = createObject ( " easyBarCode.aspBarCode ")
BC.TypeBar = 13 ' Bar type for csCodeEAN8
BC.Text = " 1234 " ' Only 4 numbers, this will raise with an error because it need 8 digits
BC.DrawToFile " barcode.gif "
if BC.Error <> 0 then WScript.Echo BC.ErrorMessage
Set BC = Nothing
```

Result

ERROR:Invalid length for barcode

See also
ErrorMessage

Name	**ErrorMessage**
Syntax in VB	***string = BC*** **.ErrorMessage**
Parameter type	**string**
Read / Write	/
From version	**3.0**

Return the error explanation of the last performed action:

Error Code	Description
0	None.
100	(must be %d)
101	Unspecified error
102	Unavailable char for this type of barcode.
103	Internal
104	Internal
105	Invalid length of barcode.
106	ITF14 - first char must be 0
107	. char must be space.
108	2of5 Interleaved/ITF - count of digits must be even.
109	Internal error on text code

See also

Error

Filename

Name	**FileName**
Syntax in VB	***BC*** **.FileName = string**
Parameter type	**string**
Read / Write	/
Default	**~aspBC**

This property sets the default name to use when using it in conjunction with the Draw method. When you use the Draw method it automatically gets all properties, renders the barcode and saves it to disk. The filename that uses is a combination of the filename setting with a sequential number that checks on that directory that must not exist.

After drawing the barcode do not check this property as it does not contain the saved name, it was returned by the Draw method.

The default FileName that uses every time that saves it to disk is "~aspBC" plus a number.

See also

Draw

FileType

Name	**FileType**
Syntax in VB	***BC*** **.FieType = integer**
Parameter type	**integer**
Read / Write	/
Default	**1 (GIF)**

You can set the graphic output to be a Bitmap or a GIF. For setting a Bitmap output set it to 0 and for GIF format to 1. By default is set to GIF format as it has a better compression technique.

Code	Description
0	BMP (Windows Bitmap)
1	GIF
2	JPEG

Note: On the CGI module this can not be set, it always returns GIF graphic.

See also
Transparent

Font_Code

Name	**Font_Code**
Syntax in VB	***BC*** **.Font_Code = string**
Parameter type	**String**
Read / Write	/
Default	**Arial**
From version	**3.0**

Sets the default font to use when drawing the type of the bar code

Example in Visual Basic script:

```
set BC = createObject ( "
easyBarCode.aspBarCode ")
BC.TypeBar = 14
BC.Font_Code = " Courier New "
BC.DrawToFile " barcode.gif "
Set BC = Nothing
```

Result

See also

Font_Type Size_Type

Font_Type

Name	**Font_Type**
Syntax in VB	***BC*** **.Font_Type = string**
Parameter type	**String**
Read / Write	/
Default	**Arial**
From version	**3.0**

Sets the default font to use when drawing the type of the bar code

Example in Visual Basic script:

```
set BC = createObject ( "
easyBarCode.aspBarCode ")
BC.TypeBar = 14
BC.Display = 3
BC.Font_Type = " Courier New "
BC.DrawToFile " barcode.gif "
Set BC = Nothing
```

Result

See also

Font_Code Size_Type

Height

Name	**Height**
Syntax in VB	***BC*** **.Height = integer**
Parameter type	**Integer**
Read / Write	/

Sets the Height size of the bar code.

Note: On Shareware version you can not set an height lower than 40 pixels to prevent the shareware message to dissapear.

See also

SetSize Width

Lic_ Debug

Name	**Lic_Debug**
Syntax in VB	***BC*** **.Lic_Debug = True / False**
Parameter type	**Boolean**
Read / Write	/

This properties is only for registered users and outputs debug information on how it loads and checks the license. This is useful to check if there is some problem with the IP address of the machine and the licensed file or if the license is corrupted.

If it displays any problem, please send this debug information to support@mitdata.net to repair and send a new license file.

Example in ASP

```
<%
dim BC
' Debug the license routine on the BC version
' This is done when you get problems on the license machine
set BC = server.createobject (" easyBarCode.aspBarCode
")
BC.DEBUG = True
BC.LIC_DEBUG = True
' Loads the license file, change it if you use a different path
BC.License (" easyBC.lic ")
response.write " <br>Version Information:<br> " &
BC.Version
set BC = nothing
%>
```

See also

License

NVersion

Name	**NVersion**
Syntax in VB	***integer = BC*** **.NVersion**
Parameter type	**Integer**
Read / Write	/

Returns the version number of the library / component, you can check it to provide compatibility between different versions of the component.

Returning values
100 - this is version 1.0
110 - for version 1.10
200 - for version 2.00
300 - for version 3.00

Example

```
<%
if BC.NVersion < 300 then
   response.write " This version is not supported by our source code get a newer one
"
end if
%>
```

See also
Version

PDF417_SecurityLevel

Name	**PDF417_SecurityLevel**
Syntax in VB	***BC*** **. PDF417_SecurityLevel = integer**
Parameter type	**Integer**
Read / Write	/
Default	**0**
From version	**3.0**

When using the PDF417 type bar you can set an additional property to add error correction, you have 9 levels of corrections, on each level you correct a number of damaged characters by reducing the capacity of the barcode.

Level	Correct error	ASCII Capacity
0	0	1850
1	1	1846
2	3	1838
3	7	1822
4	15	1790
5	31	1726
6	63	1598
7	127	1342
8	255	830

Ratio

Name	**Ratio**
Syntax in VB	***BC*** **.Ratio = integer**
Parameter type	**Integer**
Read / Write	/
Default	**1**
From version	**3.10**

Increase the size of the barcode by a factor value, it will multiply the minimum readable width value by the ratio. So if a barcode has a minimum of 50 pixels wide and we set a ratio of 2, it will display a BC of 100px wide.

On old scanners setting the ratio 1, (it's default value), it will not get the sufficient resolution to read the barcode. If you have this problem then set the ratio to 2 and it will use a better resolution for the BC.

Example in Visual Basic script:

```
set BC = createObject ( "
easyBarCode.aspBarCode ")
BC.Ratio = 2
BC.DrawToFile " barcode.gif "
Set BC = Nothing
```

See also

Width

Size_Code

Name	**Size_Code**
Syntax in VB	***BC*** **.Size_Code = integer**
Parameter type	**Integer**
Read / Write	/
Default	**8**
From version	**3.0**

Sets the default font size to use when drawing the type of the bar code.

Note: Setting a value to 0 or -1 it will use an automated resize of the text in combination of the Width parameter to make the code bigger if the bar code width is bigger.

Example in Visual Basic script:

```
set BC = createObject ( "
easyBarCode.aspBarCode ")
BC.Size_Code = 12
BC.DrawToFile " barcode.gif "
Set BC = Nothing
```

Result

See also

Font_Code Size_Type Width

Size_Type

Name	**Size_Type**
Syntax in VB	***BC*** **.Size_Type = integer**
Parameter type	**Integer**
Read / Write	/
Default	**8**
From version	**3.0**

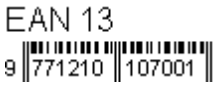
Sets the default font size to use when drawing the type of the bar code.

Note: Setting a value to 0 or -1 it will use an automated resize of the text in combination of the Width parameter to make the type bigger if the bar code width is bigger.

Example in ASP

```
<%
set BC = server.createobject (" easyBarCode.aspBarCode
")
BC.Display = 3 ' Display code and type
BC.TypeBar = 14
BC.Size_Type = 12
BC.DrawToFile " barcode.gif "
set BC = nothing
%>
```

Result



See also

Font_Type Size_Code Width

Text

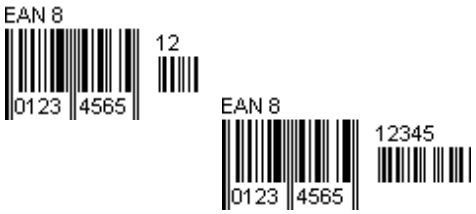
Name	**Text**
Syntax in VB	***BC*** **.Text = string**
Parameter type	**String**
Read / Write	/
Default	**0123456789**

Sets the code text of the barcode that you want to render.
Note : You can use supplementary code, this is a code used to encode information supplementary to that in the main symbol. Barcode types Ean, UPC, ISSN, ISMN, ISBN uses 2 or 5 digits as add-on. To use the supplementary code you should add an space on the text.

Example in ASP

```
<%
set BC = server.createobject (" easyBarCode.aspBarCode
")
BC.Display = 3 ' Display code and type
BC.TypeBar = 13  ' Ean8
BC.Height = 60
BC.Text = " 01234567 12 "
BC.DrawToFile " barcode_Sup2.gif "
BC.Text = " 01234567 12345 "
BC.DrawToFile " barcode_Sup5.gif "
set BC = nothing
%>
```

Result



Name	**Transparent**
Syntax in VB	***BC*** **.transparent = Boolean**
Parameter type	**Boolean**
Read / Write	/
Default	**False**
From version	**3.0**

When using GIF graphics for rendering the bar code you can mark the background as transparent type.

See also

FileType

TypeBar

Name	**TypeBar**
Syntax in VB	***BC*** **.TypeBar = Integer**
Parameter type	**Integer**
Read / Write	/
Default	**0**

There are 36 different barcode types that can be generated. Set the one that you want to use it.

Set bar code you want to use it:

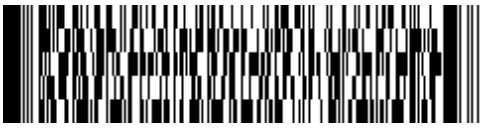
Constants

```
const csCode_2_5_interleaved = 0 ( default )
const csCode_2_5_industrial = 1
const csCode_2_5_matrix = 2
const csCode39 = 3
const csCode39Extended = 4
const csCode128A = 5
const csCode128B = 6
const csCode128C = 7
const csCode93 = 8
const csCode93Extended = 9
const csCodeMSI = 10
const csCodePostNet = 11
const csCodeCodabar = 12
const csCodeEAN8 = 13
const csCodeEAN13 = 14
const csCodeUPC_A = 15
const csCodeUPC_E0 = 16
const csCodeUPC_E1 = 17
const csCodeUPC_Supp2 = 18
const csCodeUPC_Supp5 = 19
const csCodeEAN128A = 20
const csCodeEAN128B = 21
const csCodeEAN128C = 22
' News barcodes from version 3.0
const csCode_2_5_datalogic = 23
const csCode_2_5_IATA = 24
const csCode_2_5_Invert = 25
const csCode_2_5_Coop = 26
const csCodeABCCodabar = 27
const csCodeITF = 28
const csCodeISBN = 29
const csCodeISSN = 30
const csCodeISMN = 31
const csCodeOPC = 32
const csCode11 = 33
const csCodePZN = 34
const csCodePDF417= 35
```

Example in ASP

```
<%
set BC = server.createobject (" easyBarCode.aspBarCode ")
BC.TypeBar = 35  ' PDF417
BC.Height = 60
BC.Text = " This is a test of the PDF417 barcode "+ chr(13) +" from
MITData,S.C.P. "
BC.DrawToFile " barcode.gif "
set BC = nothing
%>
```

Result



Version

Name	**Version**
Syntax in VB	***srting = BC*** **.Version**
Parameter type	**String**
Read / Write	

Returns the verbose version of the component.

See also

NVersion

Width

Name	**Width**
Syntax in VB	***BC*** **.Width = Integer**
Parameter type	**Integer**
Read / Write	/
Default	**0 (Auto)**

Sets the width size of the barcode. To make the barcode wider you should use bigger values.

Note: If you specify 0 then the barcode will be resized automatically to the correct size.

Example in ASP

```
<%
set BC = server.createobject (" easyBarCode.aspBarCode
")
BC.Display = 3 ' Display code and type
BC.TypeBar = 13  ' Ean8
BC.Height = 60
BC.Width = 200
BC.Text = " 01234567 12 "
BC.DrawToFile " barcode.gif "
set BC = nothing
%>
```

Result

See also

Height

BinaryWrite From version 2.0

Sometimes we don't want to save the barcode to a file, because it's a temporary one or you really don't want to create thousand of code bars on your hard disks. The CGI module was created to transfer it dynamically with out using a physical GIF graphic, but some users reported that some ISP or some security issues doesn't recommend to use binary EXE files. This is Why we have create a different method that also save the graphic directly to memory and you can retrieve it on a standard html page.

Syntax
BC .BinaryWrite

Example of bc-virtual.asp

```
<%
set BC = server.createobject("easyBarCode.aspBarCode")
BC.Display = 2
BC.TypeBar = 5
BC.Text = Request("Text")
BC.BinaryWrite
%>
```

To use this example you just have to use the img tag pointing to this script and filling the parameter text:

<img src="bc-virual.asp?text=012345">

Draw

When you have set all the properties you call the draw method, this generates a temporary bitmap file on the windows temporal directory of the code bar. After you call this method you have to store the returned string to a variable for using after on the html.
You can specify the file format to use by the FileType property.

Syntax
String = *BC* .Draw

Example

```
<%

set BC = server.createobject ("easyBarCode.aspBarCode")
BC.Display = 2
BC.TypeBar = 5
BC.Text = "01234"
BitmapFile = BC.Draw
set BC = nothing

%>
...
< img src ="<%=BitmapFile %>">
```

DrawToFile

The previous method generates automatically a temporary bitmap file, with this method you specify the filename, remember that you have to have write permissions on the IIS admin to generate the file.

New from version 1.2:
When specifying the filename it detects the filename extension and produces the output in bitmap or gif depending the extension given.

Write permissions:
First check on the IIS the directory security option, go to modify authentication control and anonymous access, then check the user for anonymous, usually is IUSER_ *MachineName* , but could be whatever, make sure the password is the same.
Then go to the directory where you are setting the bitmap files and check the security page, and add write permission to the user.

Syntax
BC .DrawToFile FileName as String

License

Only for registered customers.
When you need to specify another directory different from the c:\[windows]\system32 then you can use this method to locate the license file.

Syntax
BC .License LicenseFile as String

SaveStream

This function has been set for non-script programmers that wants to work directly with the pointer contents of the Barcode in memory.

You will also find it useful in .NET environments

Syntax
(PChar) = *BC* .SaveStream

SetSize

You can select the output size of the bar code image.

NOTE: Setting the Width parameter will be adjusted to a new size that can be read by a scan machine, you **can not** specify a real width. If you want a small size of the code bar then you must play with the html image property:

```
<img src="<%=BitmapFile %>" width="100" height="100">
```

UNREGISTERED USERS: You can not specify a lower height of 40 pixels. (This is for displaying the registering information)

Syntax

BC .SetSize Width as Integer , Height as Integer

```
<%

    ' Create the object, previously installed on the server
    set BC = server.createobject (" easyBarCode.aspBarCode ")
    BC.Display = 2
    BC.TypeBar = 5
    BC.Text = " 937524594 "
    FileName1 = BC.Draw
    ' response.Write "Generate Temp = " & FileName1 & "<br>"




%>

< p >Draw functions return: < br >
< img src =" <%=FileName1 %> ">
< /p >
```