

Generating Barcodes and QR Codes with 4D View Pro

By Tai Bui, Technical Services Engineer, 4D Inc.

Technical Note 24-01

Table of Contents

Table of Contents	2
Abstract.....	3
Introduction	3
Encoded Images	3
Barcodes and 1-Dimensional Encoded Images	3
QR Codes and 2-Dimensional Encoded Images	4
4D View Pro Overview	5
Barcode Functions	6
QRCode	6
EAN-13	7
EAN-8	9
Codabar	10
Code39	11
Code93	12
Code128	13
GS1_128	15
Code49	16
PDF417	17
DataMatrix	18
Offscreen Area	19
Web Area Commands	19
Encoded Image Generator.....	19
Classes.....	19
_VP_Barcode_Base	20
_VP_Barcode_Subbase.....	22
Individual Classes	22
Image Generation	23
Example Database	26
Conclusion	29

Abstract

There are various ways to represent and transport data. One convenient way to do so is through images. By encoding data in an image, they can be formatted to a consistent and portable size. A commonly known format that is used regularly is barcodes. Barcodes typically represent a string of characters or numbers that typically refer to an identification value of something, such as the product number of an item in a store. Another typically encountered format is QR codes which also contain string data but are more commonly used to contain links to a web address for promotions or relevant sites. 4D can also be expanded to take advantage of these formats in reading and generating the images. This Technical Note will provide an implementation of a feature set that allows barcode and QR code images to be generated utilizing 4D View Pro.

Introduction

This Technical Note will provide details about barcodes and QR codes. Information about the features of 4D View Pro used to implement the feature to generate the images will also be explained to provide details on the implementation. Classes are used to implement the feature and make it transferable to any other compatible 4D database. An overview of the code of the functions and attributes of the class will be provided and a walkthrough of how to use the classes will also be explained. A sample database with a demo and the source code of the implementation is also provided. The database is a 4Dv20 project mode database. As the database uses 4D View Pro, a 4D View Pro license will also be needed to run the code. Moving forward this Technical Note will refer to both formats as encoded images for brevity unless specified.

Encoded Images

Encoded images typically come in one of two formats. They are either 1-dimensional, like barcodes, or they can be 2-dimensional, like QR codes. All these encoded image formats are like each other and only differ in their presentation. Both are intended to be scanned to interpret the data and not for humans to decode without tools.

Barcodes and 1-Dimensional Encoded Images

The simpler encoded image format is 1-dimensional encoded images. Barcodes are 1-dimensional encoded images in which the data is translated into various bars in a specific pattern. Barcodes can be read from a scanner or camera which translates the bars into data by scanning across the width of the barcode in the 1 dimension getting the widths and gaps of the bars. Below is an example of a simple barcode:



Image 1: Example Barcode

The height of the image does not matter if it can be seen by the scanner. Below is an example of where a scan is performed. A scan can be performed at any vertical position of the barcode as long as the entire horizontal width of the barcode is captured. Any of the three red lines will capture the barcode and will get the same values based on the widths of the bars and gaps.



Image 2: Example of scan positions

Barcodes are regularly found on products being sold in stores to encode the product number of the items. This allows scanners for registers to pull up the product and charge the price listed in the database or for applications to pull up the product details, such as its online product page or inventory count. Barcodes have existed for a long time and have various types/formats. The examples will reference one type of barcode, but the implemented feature will support all formats supported by 4D View Pro.

QR Codes and 2-dimensional Encoded Images

For 2-dimensional encoded images, the second dimension is also considered. Most 2-dimensional images use a square or rectangular format and encode the data by filling in squares. Like barcodes, the filled squares and gaps in the square or rectangular area are used to represent the data. There are other types of 2-dimensional formats, but QR codes are one of the most recognized. QR codes are read from a scanner or camera that reads the entire 2-dimensional image. QR codes are structured so that their orientation can be determined to figure out the top and bottom to read the data properly. Below is an example of a QR code. As seen, a horizontal scan at various vertical points may not return the same widths of the blocks and gaps. Instead, the entire image must be captured as a whole.



Image 3: Example QR Code

QR codes are formatted to use the three squares in the lower left, upper left, and upper right to determine the orientation of the QR code, therefore even if the image is not straight or rotated it will return the same results.



Image 4: Example from Image 3 rotated 235 degrees.

Since smartphones with cameras and internet access are more readily available, QR codes have been typically used more for promotional purposes or to provide a relevant link. Smartphone cameras have been updated to automatically parse QR codes which can then allow the user to open the link or determine what was encoded.

These encoded images may be useful in various cases for various 4D databases. They can be used for warehouse databases as inventory and product management, user badges to log in or record activity, or even send out QR code images in emails or printouts.

4D View Pro Overview

4D View Pro uses the SpreadJS engine to run through a 4D Web Area. The feature set provides a spreadsheet feature set allowing for many functions from spreadsheet applications to be

utilized. The following is an overview of the relevant features of 4D View Pro utilized for the implementation of the image generator.

Barcode Functions

The relevant functions to this implementation are the barcode functions. The functions are all similarly formatted. The functions take in the value to be encoded as the first parameter. Depending on the image format additional parameters for the functions are available to format the resulting image. When the function is evaluated, the resulting encoded image is displayed on the specified cell of the 4D View Pro document.

As of the time of this Technical Note, SpreadJS has the encoded image formats and the associated functions:

Encoded Image Format	Function	Dimension
QRCode	BC_QRCODE	2
EAN-13	BC_EAN13	1
EAN-8	BC_EAN8	1
Codabar	BC_CODABAR	1
Code39	BC_CODE39	1
Code93	BC_CODE93	1
Code128	BC_CODE128	1
GS1_128	BC_GS1_128	1
Code49	BC_CODE	1
PDF417	BC_PDF417	2
DataMatrix	BC_DATAMATRIX	2

Each function has some unique properties for the parameters, these will be listed in the following subsections as well as providing a link to the encoded images' documentation and the functions' documentation. The nuances of the parameters are not explained in this Technical Note, and any additional details should be researched separately.

General Barcode Feature Documentation:

<https://developer.mescius.com/spreadjs/docs/versions/v14/online/WorkingwithBarcodes.html>

General Barcode Function Documentation:

<https://developer.mescius.com/spreadjs/docs/v15/formulareference/FormulaFunctions/barcode-functions>

QRCode

Feature Documentation:

<https://developer.mescius.com/spreadjs/docs/versions/v14/online/SettingQRCode.html>

General Barcode Function Documentation:

https://developer.mescius.com/spreadjs/docs/v15/formulareference/FormulaFunctions/barcode-functions/BC_QRCODE

Function:

BC_QRCODE(value, color, backgroundColor, errorCorrectionLevel, model, version, mask, connection, connectionNo, charCode, charset, quietZoneLeft, quietZoneRight, quietZoneTop, quietZoneBottom)

Parameters:

Name	Description
value	A string that represents encode on the symbol of QRCode.
color	A color that represents the barcode color. The default value is 'rgb(0,0,0)'.
backgroundColor	A color that represents the barcode backgroundcolor. The default value is 'rgb(255, 255, 255)'
errorCorrectionLevel	A string that represents the error correction level of QRCode. It has 'L M Q H' four error correction levels. The default value is 'L'.
model	A value that represents the model of QRCode. It has 1 and 2 models. The default value is 2.
version	Vesion range is 1-14 for model1 and model 2. It has 'auto 1-14 1-40' values. The default value is 'auto'.
mask	A value that represents mask pattern for QRCode. It has 'auto and 0-7' eight mask pattern.
connection	A value that represents whether the symbol is part of a structured append message. The default value is false.
connectionNo	Specifies which block the symbol is in the structured append message. It has '0-15' values. The default value is '0'.
charCode	A value that represents the collection of characters of QRCode.
charset	A value that represents which charset to use. It has 'UTF-8 and Shift-JIS'.
quietZoneLeft	A value that represents the size of left quiet zone.
quietZoneRight	A value that represents the size of right quiet zone.
quietZoneTop	A value that represents the size of top quiet zone.
quietZoneBottom	A value that represents the size of bottom quiet zone.

Feature Documentation:

<https://developer.mescius.com/spreadjs/docs/versions/v14/online/SettingEAN-13.html>

General Barcode Function Documentation:

https://developer.mescius.com/spreadjs/docs/v15/formulareference/FormulaFunctions/barcode-functions/BC_EAN13

Function:

BC_EAN13(value, color, backgroundColor, showLabel, labelPosition, addOn, addOnLabelPosition, fontFamily, fontStyle, fontWeight, fontTextDecoration, fontTextAlign, fontSize, quietZoneLeft, quietZoneRight, quietZoneTop, quietZoneBottom)

Parameters:

Name	Description
value	Specifies that the value length must be 12 or 13.
color	A color that represents the barcode color. The default value is 'rgb(0,0,0)'.
backgroundColor	A color that represents the barcode backgroundcolor. The default value is 'rgb(255, 255, 255)'
showLabel	Specifies whether to show label text when the barcode has label.
labelPosition	A value that represents the label position when the label is shown.
addOn	A string that represents the add text of QRCode. Specifies that value length must be 2 or 5.
addOnLabelPosition	The position to add the text when text is shown.
fontFamily	A string that represents the label text fontFamily. The default value is 'sans-serif'.
fontStyle	A string that represents the label text fontStyle. The default value is 'normal'.
fontWeight	A string that represents the label text fontWeight. The default value is 'normal'.
fontTextDecoration	A string that represents the label text fontTextDecoration. The default value is 'none'.
fontTextAlign	A string that represents the label text fontTextAlign. The default value is 'center'.
fontSize	A string that represents the label text fontSize. The default value is '12px'.
quietZoneLeft	A value that represents the size of left quiet zone.
quietZoneRight	A value that represents the size of right quiet zone.

quietZoneTop	A value that represents the size of top quiet zone.
quietZoneBottom	A value that represents the size of bottom quiet zone.

EAN-8

Feature Documentation:

<https://developer.mescius.com/spreadjs/docs/versions/v14/online/SettingEAN-8.html>

General Barcode Function Documentation:

https://developer.mescius.com/spreadjs/docs/v15/formulareference/FormulaFunctions/barcode-functions/BC_EAN8

Function:

BC_EAN8(value, color, backgroundColor, showLabel, labelPosition, fontFamily, fontStyle, fontWeight, fontTextDecoration, fontTextAlign, fontSize, quietZoneLeft, quietZoneRight, quietZoneTop, quietZoneBottom)

Parameters:

Name	Description
value	Specifies that the value length must be 7 or 8.
color	A color that represents the barcode color. The default value is 'rgb(0,0,0)'.
backgroundColor	A color that represents the barcode backgroundcolor. The default value is 'rgb(255, 255, 255)'
showLabel	Specifies whether to show label text when the barcode has label.
labelPosition	A value that represents the label position when the label is shown.
fontFamily	A string that represents the label text fontFamily. The default value is 'sans-serif'.
fontStyle	A string that represents the label text fontStyle. The default value is 'normal'.
fontWeight	A string that represents the label text fontWeight. The default value is 'normal'.
fontTextDecoration	A string that represents the label text fontTextDecoration. The default value is 'none'.
fontTextAlign	A string that represents the label text fontTextAlign. The default value is 'center'.
fontSize	A string that represents the label text fontSize. The default value is '12px'.

quietZoneLeft	A value that represents the size of left quiet zone.
quietZoneRight	A value that represents the size of right quiet zone.
quietZoneTop	A value that represents the size of top quiet zone.
quietZoneBottom	A value that represents the size of bottom quiet zone.

Codabar

Feature Documentation:

<https://developer.mescius.com/spreadjs/docs/versions/v14/online/SettingCodabar.html>

General Barcode Function Documentation:

https://developer.mescius.com/spreadjs/docs/v15/formulareference/FormulaFunctions/barcode-functions/BC_CODABAR

Function:

BC_CODABAR(value, color, backgroundColor, showLabel, labelPosition, checkDigit, nwRatio, fontFamily, fontStyle, fontWeight, fontTextDecoration, fontTextAlign, fontSize, quietZoneLeft, quietZoneRight, quietZoneTop, quietZoneBottom)

Parameters:

Name	Description
value	A string that represents encode on the symbol of QRCode.
color	A color that represents the barcode color. The default value is 'rgb(0,0,0)'.
backgroundColor	A color that represents the barcode backgroundcolor. The default value is 'rgb(255, 255, 255)'
showLabel	Specifies whether to show label text when the barcode has label.
labelPosition	A value that represents the label position when the label is shown.
checkDigit	Specifies whether the symbol needs a check digit. The default value is 'false'.
nwRatio	A value that represents the wide and narrow bar ratio. It has values 2 3. The default value is '3'.
fontFamily	A string that represents the label text fontFamily. The default value is 'sans-serif'.
fontStyle	A string that represents the label text fontStyle. The default value is 'normal'.
fontWeight	A string that represents the label text fontWeight. The default

	value is 'normal'.
fontTextDecoration	A string that represents the label text fontTextDecoration. The default value is 'none'.
fontTextAlign	A string that represents the label text fontTextAlign. The default value is 'center'.
fontSize	A string that represents the label text fontSize. The default value is '12px'.
quietZoneLeft	A value that represents the size of left quiet zone.
quietZoneRight	A value that represents the size of right quiet zone.
quietZoneTop	A value that represents the size of top quiet zone.
quietZoneBottom	A value that represents the size of bottom quiet zone.

Code39

Feature Documentation:

<https://developer.mescius.com/spreadjs/docs/versions/v14/online/SettingCode39.html>

General Barcode Function Documentation:

https://developer.mescius.com/spreadjs/docs/v15/formulareference/FormulaFunctions/barcode-functions/BC_CODE39

Function:

BC_CODE39(value, color, backgroundColor, showLabel, labelPosition, labelWithStartAndStopCharacter, checkDigit, nwRatio, fullASCII, fontFamily, fontStyle, fontWeight, fontTextDecoration, fontTextAlign, fontSize, quietZoneLeft, quietZoneRight, quietZoneTop, quietZoneBottom)

Parameters:

Name	Description
value	A string that represents encode on the symbol of QRCode.
color	A color that represents the barcode color. The default value is 'rgb(0,0,0)'.
backgroundColor	A color that represents the barcode backgroundcolor. The default value is 'rgb(255, 255, 255)'
showLabel	Specifies whether to show label text when the barcode has label.
labelPosition	A value that represents the label position when the label is shown.

labelWithStartAndStopCharacter	Specifies whether to show the start and stop character in the label. The default value is 'false'.
checkDigit	Specifies whether the symbol needs a check digit. The default value is 'false'.
nwRatio	A value that represents the wide and narrow bar ratio. It has values 2 3. The default value is '3'.
fullASCII	Specifies whether to support full ASCII for Code39. The default value is 'false'.
fontFamily	A string that represents the label text fontFamily. The default value is 'sans-serif'.
fontStyle	A string that represents the label text fontStyle. The default value is 'normal'.
fontWeight	A string that represents the label text fontWeight. The default value is 'normal'.
fontTextDecoration	A string that represents the label text fontTextDecoration. The default value is 'none'.
fontTextAlign	A string that represents the label text fontTextAlign. The default value is 'center'.
fontSize	A string that represents the label text fontSize. The default value is '12px'.
quietZoneLeft	A value that represents the size of left quiet zone.
quietZoneRight	A value that represents the size of right quiet zone.
quietZoneTop	A value that represents the size of top quiet zone.
quietZoneBottom	A value that represents the size of bottom quiet zone.

Code93

Feature Documentation:

<https://developer.mescius.com/spreadjs/docs/versions/v14/online/SettingCode93.html>

General Barcode Function Documentation:

https://developer.mescius.com/spreadjs/docs/v15/formulareference/FormulaFunctions/barcode-functions/BC_CODE93

Function:

BC_CODE93(value, color, backgroudColor, showLabel, labelPosition, checkDigit, fullASCII, fontFamily, fontStyle, fontWeight, fontTextDecoration, fontTextAlign, fontSize, quietZoneLeft, quietZoneRight, quietZoneTop, quietZoneBottom)

Parameters:

Name	Description
value	A string that represents encode on the symbol of QRCode.
color	A color that represents the barcode color. The default value is 'rgb(0,0,0)'.
backgroundColor	A color that represents the barcode backgroundcolor. The default value is 'rgb(255, 255, 255)'
showLabel	Specifies whether to show label text when the barcode has label.
labelPosition	A value that represents the label position when the label is shown.
checkDigit	Specifies whether the symbol needs a check digit. The default value is 'false'.
fullASCII	Specifies whether to support full ASCII for Code93. The default value is 'false'.
fontFamily	A string that represents the label text fontFamily. The default value is 'sans-serif'.
fontStyle	A string that represents the label text fontStyle. The default value is 'normal'.
fontWeight	A string that represents the label text fontWeight. The default value is 'normal'.
fontTextDecoration	A string that represents the label text fontTextDecoration. The default value is 'none'.
fontTextAlign	A string that represents the label text fontTextAlign. The default value is 'center'.
fontSize	A string that represents the label text fontSize. The default value is '12px'.
quietZoneLeft	A value that represents the size of left quiet zone.
quietZoneRight	A value that represents the size of right quiet zone.
quietZoneTop	A value that represents the size of top quiet zone.
quietZoneBottom	A value that represents the size of bottom quiet zone.

Code128

Feature Documentation:

<https://developer.mescius.com/spreadjs/docs/versions/v14/online/SettingCode128.html>

General Barcode Function Documentation:

Function:

BC_CODE128(value, color, backgroudColor, showLabel, labelPosition, codeSet, fontFamily, fontStyle, fontWeight, fontTextDecoration, fontTextAlign, fontSize, quietZoneLeft, quietZoneRight, quietZoneTop, quietZoneBottom)

Parameters:

Name	Description
value	A string that represents encode on the symbol of QRCode.
color	A color that represents the barcode color. The default value is 'rgb(0,0,0)'.
backgroundColor	A color that represents the barcode backgroundcolor. The default value is 'rgb(255, 255, 255)'
showLabel	Specifies whether to show label text when the barcode has label.
labelPosition	A value that represents the label position when the label is shown.
codeSet	A value that represents which code is set to use for QRCode. It has 'auto A B C' values. The default value is 'auto'.
fontFamily	A string that represents the label text fontFamily. The default value is 'sans-serif'.
fontStyle	A string that represents the label text fontStyle. The default value is 'normal'.
fontWeight	A string that represents the label text fontWeight. The default value is 'normal'.
fontTextDecoration	A string that represents the label text fontTextDecoration. The default value is 'none'.
fontTextAlign	A string that represents the label text fontTextAlign. The default value is 'center'.
fontSize	A string that represents the label text fontSize. The default value is '12px'.
quietZoneLeft	A value that represents the size of left quiet zone.
quietZoneRight	A value that represents the size of right quiet zone.
quietZoneTop	A value that represents the size of top quiet zone.
quietZoneBottom	A value that represents the size of bottom quiet zone.

GS1_128

Feature Documentation:

https://developer.mescius.com/spreadjs/docs/versions/v14/online/SettingGS1_128.html

General Barcode Function Documentation:

https://developer.mescius.com/spreadjs/docs/v15/formulareference/FormulaFunctions/barcode-functions/BC_GS1_128

Function:

BC_GS1_128(value, color, backgroundColor, showLabel, labelPosition, fontFamily, fontStyle, fontWeight, fontTextDecoration, fontTextAlign, fontSize, quietZoneLeft, quietZoneRight, quietZoneTop, quietZoneBottom)

Parameters:

Name	Description
value	A string that represents encode on the symbol of QRCode.
color	A color that represents the barcode color. The default value is 'rgb(0,0,0)'.
backgroundColor	A color that represents the barcode backgroundcolor. The default value is 'rgb(255, 255, 255)'
showLabel	Specifies whether to show label text when the barcode has label.
labelPosition	A value that represents the label position when the label is shown.
fontFamily	A string that represents the label text fontFamily. The default value is 'sans-serif'.
fontStyle	A string that represents the label text fontStyle. The default value is 'normal'.
fontWeight	A string that represents the label text fontWeight. The default value is 'normal'.
fontTextDecoration	A string that represents the label text fontTextDecoration. The default value is 'none'.
fontTextAlign	A string that represents the label text fontTextAlign. The default value is 'center'.
fontSize	A string that represents the label text fontSize. The default value is '12px'.
quietZoneLeft	A value that represents the size of left quiet zone.
quietZoneRight	A value that represents the size of right quiet zone.
quietZoneTop	A value that represents the size of top quiet zone.

quietZoneBottom	A value that represents the size of bottom quiet zone.
-----------------	--

Code49

Feature Documentation:

<https://developer.mescius.com/spreadjs/docs/versions/v14/online/SettingCode49.html>

General Barcode Function Documentation:

https://developer.mescius.com/spreadjs/docs/v15/formulareference/FormulaFunctions/barcode-functions/BC_CODE49

Function:

BC_CODE49(value, color, backgroudColor, showLabel, labelPosition, grouping, groupNo, fontFamily, fontStyle, fontWeight, fontTextDecoration, fontTextAlign, fontSize, quietZoneLeft, quietZoneRight, quietZoneTop, quietZoneBottom)

Parameters:

Name	Description
value	A string that represents encode on the symbol of QRCode.
color	A color that represents the barcode color. The default value is 'rgb(0,0,0)'.
backgroundColor	A color that represents the barcode backgroundcolor. The default value is 'rgb(255, 255, 255)'
showLabel	Specifies whether to show label text when the barcode has label.
labelPosition	A value that represents the label position when the label is shown.
grouping	Specifies whether the symbol mode is Group Alphanumeric Mode. The default value is 'false'.
groupNo	A value that represents the index of symbol in the group. The default value is '0'
fontFamily	A string that represents the label text fontFamily. The default value is 'sans-serif'.
fontStyle	A string that represents the label text fontStyle. The default value is 'normal'.
fontWeight	A string that represents the label text fontWeight. The default value is 'normal'.
fontTextDecoration	A string that represents the label text fontTextDecoration. The default value is 'none'.
fontTextAlign	A string that represents the label text fontTextAlign. The

	default value is 'center'.
fontSize	A string that represents the label text fontSize. The default value is '12px'.
quietZoneLeft	A value that represents the size of left quiet zone.
quietZoneRight	A value that represents the size of right quiet zone.
quietZoneTop	A value that represents the size of top quiet zone.
quietZoneBottom	A value that represents the size of bottom quiet zone.

PDF417

Feature Documentation:

<https://developer.mescius.com/spreadjs/docs/versions/v14/online/SettingPDF417.html>

General Barcode Function Documentation:

https://developer.mescius.com/spreadjs/docs/v15/formulareference/FormulaFunctions/barcode-functions/BC_PDF417

Function:

BC_PDF417(value, color, backgroundColor, errorCorrectionLevel, rows, columns, compact, quietZoneLeft, quietZoneRight, quietZoneTop, quietZoneBottom)

Parameters:

Name	Description
value	A string that represents encode on the symbol of QRCode.
color	A color that represents the barcode color. The default value is 'rgb(0,0,0)'.
backgroundColor	A color that represents the barcode backgroundcolor. The default value is 'rgb(255, 255, 255)'
errorCorrectionLevel	A string that represents the error correction level of PDF417. It has 'auto 0-8' values. The default value is 'auto'.
rows	A value that specifies the number of rows in the symbol. It has 'auto 3-90' values. The default value is 'auto'.
columns	A value that specifies the number of columns in the symbol. It has 'auto 1-30' values. The default value is 'auto'.
compact	Specifies whether it is a compact PDF417. The default value is 'false'.
quietZoneLeft	A value that represents the size of left quiet zone.
quietZoneRight	A value that represents the size of right quiet zone.

quietZoneTop	A value that represents the size of top quiet zone.
quietZoneBottom	A value that represents the size of bottom quiet zone.

DataMatrix

Feature Documentation:

<https://developer.mescius.com/spreadjs/docs/versions/v14/online/SettingDataMatrix.html>

General Barcode Function Documentation:

https://developer.mescius.com/spreadjs/docs/v15/formulareference/FormulaFunctions/barcode-functions/BC_DATAMATRIX

Function:

BC_DataMatrix(value, color, backgroudColor, eccMode, ecc200SymbolSize, ecc200EndcodingMode, ecc00_140Symbole, structureAppend, structureNumber, fileIdentifier, quietZoneRight, quietZoneTop, quietZoneBottom)

Parameters:

Name	Description
value	A string that represents encode on the symbol of QRCode.
color	A color that represents the barcode color. The default value is 'rgb(0,0,0)'.
backgroundColor	A color that represents the barcode backgroundcolor. The default value is 'rgb(255, 255, 255)'
eccMode	A value that represents which ecc mode to use. It has the following values : 'ECC000, ECC050, ECC080, ECC100, ECC140, ECC200'.
ecc200SymbolSize	A value that specifies the size of the ECC200 symbol only. The default value is 'squareAuto'.
ecc200EndcodingMode	A value that specifies which encoding mode to use for the symbol. The default value is 'auto'.
ecc00_140Symbole	A value that specifies the size of the ECC000-140 symbol only. The default value is 'auto'.
structureAppend	Specifies whether the symbol is part of a structured append message ECC200 only.The default value is 'false'.
structureNumber	A value that represents which block the symbol is in the structured append message. It has the value '0-15', only for ECC200. The default value is '0'.
fileIdentifier	A value that specifies the file identification. It has values '1-254', only for ECC200. The default value is '0'.

quietZoneRight	A value that represents the size of right quiet zone.
quietZoneTop	A value that represents the size of top quiet zone.
quietZoneBottom	A value that represents the size of bottom quiet zone.

Offscreen Area

Another feature of 4D View Pro utilized for generating the encoded images is the offscreen area. With the introduction of offscreen web areas in 4D, 4D View Pro can also be run in an offscreen area. The offscreen area is run by calling the **VP Run offscreen area** method. The method takes in an object of parameters that can run code based on form events applicable to a View Pro area. The main event used is the [On VP Ready](#) event.

This feature allows the use of 4D View Pro features without needing to run a form, including access to the barcode functions. The implementation uses an offscreen 4D View Pro area to run a new 4D View Pro document. 4D View Pro commands are then used to insert the specified barcode functions to generate the image. The encoded image can then be extracted from the 4D View Pro document to return the image as a 4D picture.

Web Area Commands

Since 4D View Pro is a SpreadJS feature run in a 4D Web Area, 4D Web Area commands can be used on a 4D View Pro area. Commands like **WA Evaluate JavaScript** and **WA EXECUTE JAVASCRIPT FUNCTION** can be used to run javascript on the 4D View Pro area, which also can be run on an offscreen 4D View Pro area. The main use of this feature was to format the 4D View Pro area and extract the encoded image from the 4D View Pro document.

As a spreadsheet application solution, 4D View Pro contains many features. Of these features, the ones used for the implementation of an encoded image generator are the barcode functions, the offscreen area, and the web area javascript commands. These features are used to generate the encoded images and extract them as a 4D picture.

Encoded Image Generator

The encoded image generator is implemented as classes. The feature is broken up into many classes to maintain simplicity and modularity. The overall process of the use is explained in the following steps. First, an instance of the class for the specific barcode function of 4D View Pro is initialized. Next, the instance's properties which mirror the properties of the barcode function can then be modified if needed. When properly formatted, a function is called to generate and return the image.

Classes

The feature is broken up into separate classes. There are thirteen classes, a base class for all of the barcode classes, a sub-base class that extends the base class for 1-dimensional

classes, and then eleven classes for each of the individual barcode functions available in 4D View Pro with the 2-dimensional classes extending the base class, and the 1-dimensional classes extending the sub-base class. Below is the hierarchical list of classes with the inheritance based on the leveling of the list:

- **_VP_Barcode_Base** –Base Class for all Encoded Images
 - **VP_DataMatrix**
 - **VP_PDF417**
 - **VP_QRCode**
 - **_VP_Barcode_Subbase** – Base Class for 1D Images, Extends **_VP_Barcode_Base**
 - **VP_Barcode_EAN13**
 - **VP_Barcode_EAN8**
 - **VP_Barcode_GS1_128**
 - **VP_Barcode128**
 - **VP_Barcode39**
 - **VP_Barcode49**
 - **VP_Barcode93**

_VP_Barcode_Base

As shown in the barcode functions' overview in the 4D View Pro Overview section, many properties are shared across the functions such as the value, colors, and quiet zones. Another shared feature across all the image formats is that the generation of the image will be the same. As such, a generic base class was created allowing for all subclasses to inherit these properties and functions. The base class is named '**_VP_Barcode_Base**'. By adding an underscore to the start of an object notation item, the item is considered "hidden" and will not be actively shown in the method editor.

The class contains the following properties which represent the similarly named properties used in all the barcode functions:

Name	Description
.Value	A string that represents encode on the symbol...
.Color	A color that represents the barcode color. The default value is 'rgb(0,0,0)'.
.BackgroundColor	A color that represents the barcode backgroundcolor. The default value is 'rgb(255, 255, 255)'
.QuietZone_Left	A value that represents the size of left quiet zone.
.QuietZone_Right	A value that represents the size of right quiet zone.

<code>.QuietZone_Top</code>	A value that represents the size of top quiet zone.
<code>.QuietZone_Bottom</code>	A value that represents the size of bottom quiet zone.

Additional properties are added for the format of the resulting 4D picture.

Name	Description
<code>.imgFormat</code>	Format of the image, must be HTML compatible. Defaults to “png”
<code>.imgHeight</code>	Height of the 4D picture in px. Defaults to 380
<code>.imgWidth</code>	Width of the 4D picture in px. Defaults to 380

The class contains four functions:

Name	Description
<code>._formulaParams1()</code>	Returns a string of the starting parameters for the VP barcode formula to enter into the 4D View Pro document.
<code>._formulaParamsQuietZone()</code>	Returns a string of the ending quiet zone parameters for the VP barcode formula to enter into the 4D View Pro document.
<code>.onEvent</code>	A function for the offscreen 4D View Pro area to generate the encoded image and extract it.
<code>.generateImage()</code>	Returns a picture of the encoded image based on the parameters entered.

This class is not intended to be used directly, which is why it is hidden. The features of the class are to initialize the shared properties, assist with building the 4D View Pro formula with the shared properties, and mainly run the code to generate the image.

The images are generated from the '`generateImage()`' function. The function calls the **VP Run offscreen area** method passing itself as the formatting object. The offscreen area then uses the class's '`onEvent`' function to run the 4D View Pro area. When the area is ready, the [On VP Ready](#) event is triggered. The event inserts the formula for the barcode into cell 0,0 of the 4D View Pro document. It then sets the height and width of the cell to the dimensions specified in the attributes. Next, javascript is used to create a new HTML document with just cell 0,0 and then the document is converted to an image in base64 format and returned to 4D. 4D can then decode the base64 data into a picture. If the function succeeds it will create three new attributes for the instance of the class as well as directly returning the image:

Name	Description
<code>.result</code>	A Boolean that is a True if the VP Offscreen area is completed
<code>.image</code>	A Picture containing the resulting image
<code>.blob</code>	A Blob of the resulting image

_VP_Barcode_Subbase

The ‘**_VP_Barcode_Subbase**’ class extends the ‘**_VP_Barcode_Base**’ class and inherits its attributes and functions. The 1-dimensional classes all share some additional properties that the 2-dimensional classes do not have which the ‘**_VP_Barcode_Subbase**’ class includes to inherit. These properties are for the labels that can be included in the images:

Name	Description
.ShowLabel	Specifies whether to show label text when the barcode has label.
.LabelPosition	A value that represents the label position when the label is shown.
.FontFamily	A string that represents the label text fontFamily. The default value is 'sans-serif'.
.FontItalic	A Boolean that represents the label text for fontStyle. The default value is False.
.FontBold	A Boolean that represents the label text for fontWeight. The default value is False.
.FontTextDecoration	A string that represents the label text fontTextDecoration. The default value is 'none'.
.FontTextAlign	A string that represents the label text fontTextAlign. The default value is 'center'.
.FontSize	A string that represents the label text fontSize. The default value is '12px'.

The class contains five functions. Two are hidden functions used to build the 4D View Pro barcode formula for the label parameters, and three are used to return a collection of valid values for specific properties. For example, the ‘*.listFontTextAligns()*’ function returns a collection with the four values: “center”, “left”, “right”, and “group”. These are the only values that apply to the ‘fontTextAlign’ parameter of the 4D View Pro barcode functions. The other two ‘*.list...*’ functions are used similarly.

This class is also not intended to be used directly and is used to reduce the repetition of code in the individual classes for 1-dimensional images.

Individual Classes

The remaining classes for each of the encoded image formats all use a similar structure. Each barcode function of 4D View Pro may include some specific properties. These properties are added as an attribute of the class. For each attribute that can only contain a specific set of values, a ‘*.list...*’ function that returns a collection of allowed values for the attribute. The functions are named in the pattern of ‘*list{attributeName}s*’.

So for the '**FontTextAligns**' property from the '**_VP_Barcode_Subbase**' class, the '**.list...()**' function was **.listFontTextAligns()**. Attributes using Booleans or numeric values will not have a '**.list...()**' function.

Each class also has a '**.VPFormula()**' function. This function returns the full 4D View Pro formula that can be applied to a 4D View Pro document's cell to generate the image of the current class. All the barcode functions follow a consistent ordering of the parameters, allowing for the base classes to be used to generate parts of the formula as mentioned in the description of the hidden formula functions.

The formatting is as follows:

{BC_function} ({Value}, {Colors}, {Function Specific Properties}, {Quiet Zones})

With 1-dimensional functions adding in label properties and label fonts:

{BC_function} ({Value}, {Colors}, {Label Properties}, {Function Specific Properties}, {Label Font}, {Quiet Zones})

The functions, {BC_function}, are applied with the individual '**.VPFormula()**' functions of each class.

The {Value} and {Color} parameters are applied using the '**_VP_Barcode_Base**' class's '**.formulaParams1()**' function.

The {Label Properties} for 1D images are applied using the '**_VP_Barcode_Subbase**' class's '**.formulaParamsLabel()**' function.

The {Function Specific Properties} are applied within the individual class as part of the '**.VPFormula()**' function.

The {Label Font} parameters for 1D images are applied using the '**_VP_Barcode_Subbase**' class's '**.formulaParamsFont()**' function.

The {Quiet Zones} parameters are applied using the '**_VP_Barcode_Base**' class's '**.formulaParamsQuietZone()**' function.

All classes are not needed to run other classes, only the class for the image type and their dependent base classes. For example, if only QR Codes will be used, only the base class (**_VP_Barcode_Base**) and QR Code (**VP_QRCode**) classes are needed. For a 1D class like a barcode 128 image, it will also need the subbase class. So, if a database only needs to create a barcode 128 image it will need the base class (**_VP_Barcode_Base**), the sub base class (**_VP_Barcode_Subbase**), and the class for barcode 128 (**VP_Barcode128**).

Image Generation

Using the class is made to be as simple as possible. An instance of the class for the specific image format is created.

Example of generating a QR Code with all default attributes:

```
// Declare variables
var $QRCode_Instance : cs.VP_QRCode
var $varImage : Picture
```

```
// Initialize instance
$QRCode_Instance := cs.VP_QRCode.new("Hello")

// Generate Image
$varImage := $QRCode_Instance.generateImage()
```



Image 5: Resulting image in `$varImage` for simple example above.

As shown, with default properties used, an image can be generated using two lines of code (not including comments and variable typing). By typing the variable as an instance of a class, the method editor will suggest functions and attributes for the class, including inherited ones:

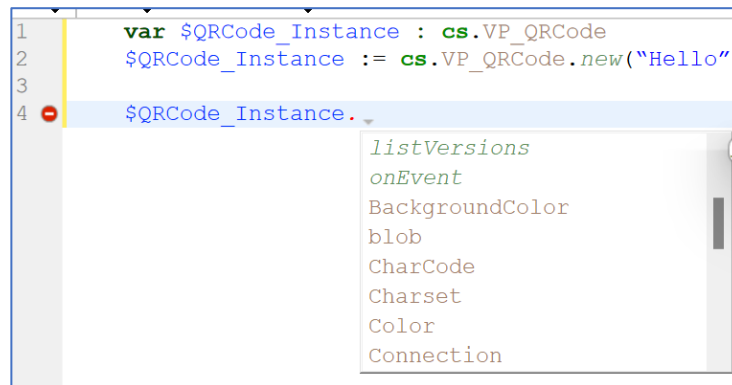


Image 6: Example of autocomplete suggestions based on class declaration.

Example of generating a QR Code and updating some attributes:

```
// Initialize instance
var $QRCode_Instance : cs.VP_QRCode
$QRCode_Instance := cs.VP_QRCode.new("Hello")

// Optionally change any attributes
$QRCode_Instance.Value := "Hello World"
$QRCode_Instance.Color := "#FFFFFF"
$QRCode_Instance.BackgroundColor := "#FF80FF"

// Generate Image
var $varImage : Picture
$varImage := $QRCode_Instance.generateImage()
```




Image 7: Resulting image in `$varImage` for example with some modified attributes.

As shown, applying changes to attributes is as simple as making an assignment with a valid value.

Example of generating a Barcode 128 Image, all of the barcode classes have a similar usage with the only differences being the specific attributes:

```
// Initialize instance
var $Barcode_Instance : cs.VP_Barcode128
$Barcode_Instance := cs.VP_Barcode128.new("Hello World")

// Optionally change any attributes
$Barcode_Instance.ShowLabel := True

// Generate Image
var $varImage : Picture
$varImage := $Barcode_Instance.generateImage()
```

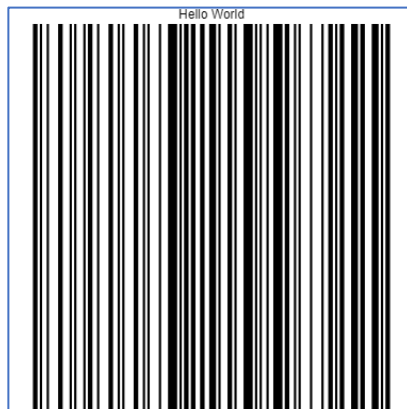


Image 8: Resulting image in `$varImage` for example of Barcode 128

The encoded image generator is implemented as a set of classes. The classes are implemented so that the features are balanced between being modular, portable, and simplistic. By splitting the 4D View Pro barcode functions into separate classes the feature set is modular, allowing for specific functions to be used while removing others. Being implemented as a class in a project, allows the feature to be portable by allowing it to be copied to any other 4D database. The feature set is simplistic as shown in the examples of the

uses, where an encoded image using default attributes can be created in as little as one line of code:

```
$varImage:= cs.VP_QRCode.new("Hello World").generateImage()
```

Any new barcode functions added to 4D View Pro, using the same pattern can be added using a similar coding structure. The code can also be modified to better fit the needs of the application.

Example Database

An example database with the implementation of the encoded image generator is provided. The example database provides an example of a user interface to generate the encoded images, which can be scanned to test on the display.

When the example database is run, an introductory window will be displayed with a similar summarized explanation of how to interact with the example. It also warns that the nuances of the properties of each of the barcode functions are not explained. For example, the difference between an EAN-13 Barcode and a Barcode 128 Barcode or what exactly is the file identifier property of a DataMatrix image is.

To start the user interface, click on the “Run Encoded Image Generator” button. This will open the barcode generator and start with a QR Code. The user interface is similarly laid out for all image formats. At the top is a drop-down to select the image format.

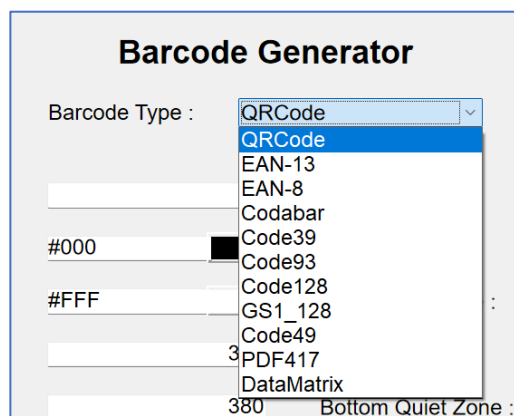


Image 9: Image Format Drop-down in Example Database

The first section of attributes below are the attributes that all of the image formats use.

Encoded Value :	<input type="text"/>		
Image Color :	#000	<input type="checkbox"/>	Left Quiet Zone : <input type="text"/>
Background Color :	#FFF		Right Quiet Zone : <input type="text"/>
Image Height :	<input type="text"/>	380	Top Quiet Zone : <input type="text"/>
Image Width :	<input type="text"/>	380	Bottom Quiet Zone : <input type="text"/>

Image 10: Attributes all Image Formats share

For the colors, clicking on the area at the end will open a color picker:

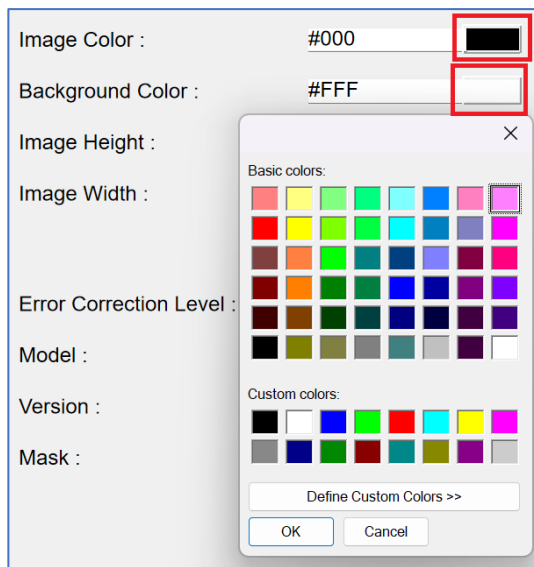


Image 11: Example of Color Picker

The bottom section of attributes are the image format-specific attributes which will change based on the selected barcode type:

Barcode Generator			
Barcode Type :		QRCode	
Encoded Value :			
Image Color :	#000	Left Quiet Zone :	
Background Color :	#FFF	Right Quiet Zone :	
Image Height :	380	Top Quiet Zone :	
Image Width :	380	Bottom Quiet Zone :	
Error Correction Level :	L	Connection :	<input type="checkbox"/>
Model :	2	Connection No :	0
Version :	auto	CharCode	
Mask :	auto	Charset :	UTF-8

Image 12: Example of QR Code attributes shown

Barcode Generator

Barcode Type : Code128

Encoded Value :

Image Color : #000

Background Color : #FFF

Image Height : 380

Image Width : 380

Show Label : ☐

Label Position : top

Code Set : auto

Left Quiet Zone :

Right Quiet Zone :

Top Quiet Zone :

Bottom Quiet Zone :

Font Family :

Font Italic : ☐

Font Bold : ☐

Font TextDecoration : none

Font TextAlign : center

Font Size : 12

Image 13: Example of Barcode 128 attributes shown

Some attributes are disabled unless the relevant attribute is enabled. For example, with the 1-dimensional barcode formats, the label-related attributes such as the position and font are disabled unless 'Show Label' is enabled.

When properly formatted, the 'Generate Image' button on the right side can be used to generate the image in the initially blank area. The 'TRACE' option can be enabled to debug and follow through the code to generate the image.

Generate Image
☐ TRACE

Image 14: Example of the right side of the image generator with 'Generate Image' and 'TRACE'

Some details should be noted when using the example. Some image formats have requirements for the values that are not enforced by the code. For example, EAN-12 uses a 12-digit value. If there are any issues with the parameters for the formula, 4D View Pro will evaluate into '#VALUE!' instead of the image:



Image 15: Example of improperly formatted formula.

Due to limitations of the 4D text type, the image height and width should be reasonably sized, if the size is too big, the base64 encoded image from the extraction will not be fully contained in the 4D text variable and may return a blank image.

Conclusion

This Technical Note provides an implementation of a feature to generate encoded images using 4D View Pro. This is a fully native feature that does not require external calls or additional resources to run. The feature utilizes the 4D View Pro feature set that has barcode functions to generate encoded images, the 4D View Pro offscreen area feature, and the ability to run JavaScript on a 4D View Pro area to implement the feature. Classes are used to implement the feature in a modular, portable, and simplistic manner. The feature can be added to other databases in a pick-and-choose manner by selecting the classes needed for the specific image formats. The use of the features is simple enough to be called in a single line by utilizing object notation. These features can be applied in various scenarios, such as for applications that deal with products and inventory, for documents like invoices or tickets, and more.