**Vetscan QR Testing**

**Table of Contents**

[1 Overview 2](#_Toc79068821)

[1.1 Scope of this Document 2](#_Toc79068822)

[1.2 Test Equipment 2](#_Toc79068823)

[1.3 Documentation 2](#_Toc79068824)

[1.4 QR Specification 3](#_Toc79068825)

[1.4.1 Content of QR code 3](#_Toc79068826)

[1.5 FUSE VETXML Test names 4](#_Toc79068827)

[1.6 Test Labels 5](#_Toc79068828)

[1.7 Test Label Generation 5](#_Toc79068829)

[1.7.1 QR code maximum data capacity 6](#_Toc79068830)

[1.8 Test Label Printing 7](#_Toc79068831)

[2 Test #1: 9](#_Toc79068832)

[2.1 Test Setup 9](#_Toc79068833)

[3 Test Instructions 10](#_Toc79068834)

[3.1 Web App 10](#_Toc79068835)

[3.2 Galaxy S10 12](#_Toc79068836)

[3.3 Python App 13](#_Toc79068837)

[4 Test Results 14](#_Toc79068838)

[4.1 Vetscan Hub Mockup Web App 14](#_Toc79068839)

[4.2 Galaxy S10 Cell Phone 14](#_Toc79068840)

[4.3 Laptop Web App 14](#_Toc79068841)

[4.4 Laptop Python App 14](#_Toc79068842)

[4.5 Vetscan Hub Mockup Python App - Autofocus Off 14](#_Toc79068843)

[4.6 Vetscan Hub Mockup Python App - Autofocus On 14](#_Toc79068844)

[5 Conclusions 15](#_Toc79068845)

[5.1 Glare 15](#_Toc79068846)

[5.2 10mm x 10mm QR labels 15](#_Toc79068847)

[5.3 Python app verses web app 15](#_Toc79068848)

[5.4 Comparison of results 15](#_Toc79068849)

[5.5 Autofocus 16](#_Toc79068850)

[6 Revision History 17](#_Toc79068851)

# Overview

This document will contain a test plan and report involving the Vetscan’s camera. This will mainly focus on the Vetscan’s camera ability to read QR codes.

Each combination of label size, module size, and total characters will be tested. Each test will be recorded as pass or fail.

## Scope of this Document

This document will be used for engineering staff to plan and document testing.

## Test Equipment

* Vetscan Hub mockup unit
* Ruler used to measure distance from the camera to a QR code, and the size of the QR code label.
* Printed QR Labels. Encoded according to ISO 18004:2006, using Level M error correction, and the max amount of data that can be contained in a QR Label.
* Boxes. Box heights:
  + For 10mm x 10mm labels: 9.5cm
  + For 20mm x 20mm labels: 9cm
  + For 50mm x 50mm labels: 9cm
* qr\_code\_genertor.py – A Python application used to generate QR labels.

## Documentation

|  |  |
| --- | --- |
| ISO 18004:2006 | QR code specification |
| <https://www.the-qrcode-generator.com/>scan | Web site – create or read a QR label with given data |

## QR Specification

### Content of QR code

**Format:** the content of the QR code must be a valid json structure.

**Name:** Mandatory information. Valid names are current FUSE VETXML Test codes.

See Name table below.

Maximum length 8 characters.

**Lot:** Mandatory information. A string consistent with the lot id from the manufacturer.

**Exp:** Mandatory information. Expiration date expressed according to ISO 8601 as yyyy-mm-dd.

**Data:** Optional. String of data to be sent verbatim to the analyzer.

Valid QR code according to specification (ISO 18004:2006)

Version: Minimum 1 (21 × 21 modules)  
 Maximum 40 (177 × 177 modules)

Size: Minimum size 10 mm × 10 mm.  
 Maximum size 50 mm × 50 mm.

Error correction: Level M.

Printed Labels: printed on a high-contrast background

Example JSON string that would be encoded into a QR label.

{

"analyzer": "<analyzer id>",

"uuid": "<uuid>",

"lot": "<string>",

"exp": "<yyyy-mm-dd>",

"data": "<optional data>"

}

This is 53 characters long without any data or whitespace, just the name value pairs and syntax requirements. An example SPE QR would be as follows:

{

"analyzer":"vetscan-spe",

"exp":"2022-05-07",

"uuid":"0b7ec890-3960-11eb-a081-2790e47ff2f4",

"lot":"1234",

"data":""

}

This is 114 characters when whitespace is removed.

## FUSE VETXML Test names

|  |  |  |
| --- | --- | --- |
| **Analyzer** | **Name** | **Description** |
| Chemistry | AR | Avian/Reptilian Profile Plus |
| CDP | Comprehensive Diagnostic |
| CC | Critical Care Plus |
| EPP | Equine Profile Plus |
| KPP | Kidney Profile Plus |
| LA | Large Animal Profile |
| MLP | Mammalian Liver Profile |
| Prep | Prep Profile II |
| T4 | Thyroxine(T4)/Cholesterol Test |
| EP | Electrolyte Plus |
| PCP | Preventive Care Profile Plus |
| PP | Phenobarbital Profile |
| Hematology | HEM | Hematology |
| Chemistry2 | PT/aPTT | PT/aPTT Combination Test |
| Fib | Equine Fibrinogen Test |
| EFib | Equine Fibrinogen |
| CFib | Canine Fibrinogen |
| CBT | Canine Blood Typing |
| FBT | Feline Blood Typing |
| PTaPTT | PT/aPTT Combination |
| Phb | Phenobarbital (not released) |
| RapidTests | E | Ehrlichia Rapid Test |
| P | Parvo Rapid Test |
| A | Anaplasma Rapid Test |
| G | Giardia Rapid Test |
| L | Lyme Rapid Test |
| FF | FeLV\_Fiv Rapid Test |
| c | cPL Rapid Test |
| FLE | Flex4 Rapid Test |
| H | Heartworm Rapid Test |
| Fecalanalysis | FOVA | Fecal Ova/Oocysts |
| FGIA | Fecal Giardia |
| Urinalysis | SA | Urine Sediment |
| SA10 | Urine Sediment and UA10 |
| SA14 | Urine Sediment and UA14 |
| UA10 | UA10 |
| UA14 | UA14 |

## Test Labels

A QR code (abbreviated from Quick Response code) is a type of matrix barcode (or two-dimensional barcode[1]) invented in 1994 by the Japanese automotive company Denso Wave.[2] A barcode is a machine-readable optical label that contains information about the item to which it is attached. In practice, QR codes often contain data for a locator, identifier, or tracker that points to a website or application. A QR code uses four standardized encoding modes (numeric, alphanumeric, byte/binary, and kanji) to store data efficiently; extensions may also be used.

* The test labels will be printed on paper with matte and glossy finishes.
* The test labels will be scanned under the Vetscan unit on boxes that optimize the image resolution.
* The test Label Sizes that will be used will be 10mm x 10mm, 20mm x 20mm, and 50mm x 50mm.
* The QR codes will contain example JSON data which is a combination of Numeric, Alphanumeric and Binary characters. Therefor the encoding of the JSON data will take more space then purely Numeric or Alphanumeric data.
  + **Numeric:**
    - **0 1 2 3 4 5 6 7 8 9**
  + **Alphanumeric:**
    - **0–9**
    - **A–Z (upper-case only)**
    - **space**
    - **$ % \* + - . / :**
  + **Binary:**
    - **a-z (lower-case)**
    - **{ } \n “ ,**
    - **Every other character**
* For example, when using the Version 20 QR Code with correction level M, the maximum allowable characters are:
  + **Numeric: 1600**
  + **Alphanumeric: 970**
  + **Binary: 666**
* When JSON text is added to the QR code, multiple character types will need to be encoded. The additional encode info will reduce the amount of space that may be used for the characters. When the test QR codes where created with JSON strings, the max amount of characters was reduced.

## Test Label Generation

* The qr\_code\_generator.py Python application was used to generate the images for the different label data densities.
  + https://github.com/lincolnloop/python-qrcode
* The app created image files in the QR\_labels\_Alphanumeric\_JSON directory.

### QR code maximum data capacity

Below is a table with each QR code’s maximum data capacity for alpanumberic text. The table also shows that using JSON text is in the QR code will reduce the maximum data capacity.

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Modules** | **Max AlphaNum** | **Max JSON Example** |
| 1 | 21 x 21 | 20 | Too small |
| 2 | 25 x 25 | 38 | Too small |
| 3 | 29 x 29 | 61 | Too small |
| 4 | 33 x 33 | 90 | Too small |
| 5 | 37 x 37 | 122 | 88 |
| 6 | 41 x 41 | 154 | 120 |
| 7 | 45 x 45 | 178 | 144 |
| 8 | 49 x 49 | 221 | 187 |
| 9 | 53 x 53 | 262 | 228 |
| 10 | 57 x 57 | 311 | 274 |
| 11 | 61 x 61 | 366 | 329 |
| 12 | 65 x 65 | 419 | 382 |
| 13 | 69 x 69 | 483 | 446 |
| 14 | 73 x 73 | 528 | 491 |
| 15 | 77 x 77 | 600 | 564 |
| 16 | 81 x 81 | 656 | 619 |
| 17 | 85 x 85 | 734 | 697 |
| 18 | 89 x 89 | 816 | 779 |
| 19 | 93 x 93 | 909 | 872 |
| 20 | 97 x 97 | 970 | 933 |
| 21 | 101 x 101 | 1035 | 999 |
| 22 | 105 x 105 | 1134 | 1097 |
| 23 | 109 x 109 | 1248 | 1211 |
| 24 | 113 x 113 | 1326 | 1289 |
| 25 | 117 x 117 | 1451 | 1415 |
| 26 | 121 x 121 | 1542 | 1505 |
| 27 | 125 X 125 | 1637 | 1600 |
| 28 | 129 X 129 | 1732 | 1695 |
| 29 | 133 X 133 | 1839 | 1803 |
| 30 | 137 x 137 | 1994 | 1957 |
| 31 | 141 x 141 | 2113 | 2076 |
| 32 | 145 x 145 | 2238 | 2201 |
| 33 | 149 x 149 | 2369 | 2332 |
| 34 | 153 x 153 | 2506 | 2469 |
| 35 | 157 x 157 | 2632 | 2595 |
| 36 | 161 x 161 | 2780 | 2744 |
| 37 | 165 x 165 | 2894 | 2857 |
| 38 | 169 x 169 | 3054 | 3017 |
| 39 | 173 x 173 | 3220 | 3183 |
| 40 | 177 x 177 | 3391 | 3355 |

## Test Label Printing

Using the program ifranview (<https://www.irfanview.com/>), it is possible to scale and print all the QR codes at once.

1. Install irfanview
2. Open irfanview
3. Select File->Thumbnails
4. Using file view pane on left hand side of screen, navigate to the directory where the QR code images are located
5. Now click on the files to select the ones you wish to print
6. Select File->Print selected files as single images (batch print)
7. Select “Print Size:” “Custom”. Set the Width and Height.
   1. For the 10mm x 10mm QR codes: 1.1 cm
   2. For the 20mm x 20mm QR codes: 2.1 cm
   3. For the 50mm x 50mm QR codes: 5.3 cm
8. Now check the box for Headnote and put $D $F in the input box so that the file name is printed with the image
9. In the “Position on paper” section, select “Center horz”
10. In the “Position on paper” section, enter a “Top margin” of 5.00
11. You can go into the Printer setup and set the printer options on the printer driver (2-sided, flip, etc.)
12. See below.



# Test #1:

The purpose of this test is to determine what is the maximum amount of data that can be read with the camera at given distances, label sizes, and label densities.

## Test Setup

* The camera for each test configuration will be tested using printed QR labels and QR code reader applications that can read the QR label’s code.
* The applications will display the camera image and display the QR code.
* The QR labels will be placed at distances to optimize the camera’s performance.
* When testing the Vetscan Hub Mockup, a box will be placed on the base to raise the QR code closer to the camera.
* The display of the Vetscan Hub Mockup will need to be tilted until it aligns to the label.
* If the QR label is decoded, the data will be displayed by the QR code reader application.
* Each test case shall be recorded in the tables below.
* If the QR label is decoded, then the test case shall be marked as “Pass”, else it shall be marked as “Fail”.

# Test Instructions

* For each test configuration, use a box that elevates the QR code off the base of the Vetscan Hub Mockup and optimizes the image in the camera. The app being used will display the QR code and can be used to determine the optimal distance from the camera.
* For each test configuration, use the QR codes printed per the “Test Label Generation” section.

## Web App

Web page: <https://www.the-qrcode-generator.com/scan>

Vetscan:

Monitor titled to max angle.

Box height: 10mm x 10mm: 9.5cm, 20mm x 20mm: 9cm, 50mm x 50mm: 9cm

ThinkPad P50 Laptop:

Distance from Web camera: ~2.25-inches

Perform the following test setup once at the beginning of testing:

1. Turn on the Vetscan Hub Mockup or Laptop that is under test.
2. Open the web site: <https://4qrcode.com/scan-qr-code.php>.
3. On the web page, click on the button labelled “Open camera”.
4. You should see an image from the unit’s camera displayed on the web page.
5. See screen capture of web page below.



For each test case, select the proper box and QR label.

1. Place the test box under the Vetscan Hub Mockup’s or Laptop’s camera.
2. Place the QR label on the top of the box if performing test on Vetscan Hub Mockup.
3. Using the web page’s camera image, center the QR tag’s image in the center of the image.
4. If the QR label can be decoded, the web page will display the QR label’s data on web page.
5. Once the image is centered, wait at most 10 seconds for the web site to decode the QR label.
6. If the QR label was decoded within 10 seconds, then the test case passed, else the test case failed.
7. Record the pass/fail status of each test case.

## Galaxy S10

Android app: QR & Barcode Reader - TeaCapps

Distance from camera: ~2-inches

Perform the following test setup once at the beginning of testing:

1. Open the QR & Barcode Reader – TeaCapps on the Galaxy S10.
2. See below. Left: QR code being scanned. Right: QR code decoded.

 

For each test case, select the proper box and QR label:

1. Move the Galaxy S10 a few centimeters above the QR code.
2. Using the camera view on the phone, center the QR tag’s image in the center of the image.
3. If the QR label can be decoded, the value will be displayed on the phone screen.
4. If the QR was decoded within 10 seconds, then the test case passed, else the test case failed.
5. Record the pass/status of each test case.

## Python App

Python app: qr\_scanner.py

Perform the following test setup once at the beginning of testing:

1. Turn on the Vetscan Hub Mockup or the Laptop being tested.
2. In a console window, enter the command:

$ python3 qr\_scanner.py

1. A window will display the view of the camera.
2. When the QR code is decoded successfully , a box is drawn around the QR code in the window, and the JSON is decoded.



For each test case, select a box that optimizes the QR label in the window.

1. Place the test box under the Vetscan Hub Mockup’s or Laptop’s camera.
2. Place the QR label on the top of the box if performing test on Vetscan Hub Mockup.
3. Using the displayed camera image, center the QR tag’s image in the center of the image.
4. If the QR label can be decoded, the JSON text will be displayed.
5. Once the image is centered, wait at most 10 seconds for the web site to decode the QR label.
6. If the QR label was decoded within 10 seconds, then the test case passed, else the test case failed.
7. Record the pass/fail status of each test case.

## 

# Test Results

## Vetscan Hub Mockup Web App

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dimension | **Version** | **Modules** | **Max AlphaNum** | **Max JSON Example** |
| 10mm x 10mm | Failed |  |  |  |
| 20mm x 20mm | 12 | 65 x 65 | 419 | 382 |
| 50mm x 50mm | 14 | 73 x 73 | 528 | 491 |

## Galaxy S10 Cell Phone

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dimension | **Version** | **Modules** | **Max AlphaNum** | **Max JSON Example** |
| 10mm x 10mm | 16 | 81 x 81 | 656 | 619 |
| 20mm x 20mm | 29 | 133 X 133 | 1839 | 1803 |
| 50mm x 50mm | 22 | 105 x 105 | 1134 | 1097 |

## Laptop Web App

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dimension | **Version** | **Modules** | **Max AlphaNum** | **Max JSON Example** |
| 10mm x 10mm | Failed |  |  |  |
| 20mm x 20mm | 6 | 41 x 41 | 154 | 120 |
| 50mm x 50mm | 15 | 77 x 77 | 600 | 564 |

## Laptop Python App

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dimension | **Version** | **Modules** | **Max AlphaNum** | **Max JSON Example** |
| 10mm x 10mm | Failed |  |  |  |
| 20mm x 20mm | 9 | 53 x 53 | 262 | 228 |
| 50mm x 50mm | 24 | 113 x 113 | 1326 | 1289 |

## Vetscan Hub Mockup Python App - Autofocus Off

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dimension | **Version** | **Modules** | **Max AlphaNum** | **Max JSON Example** |
| 10mm x 10mm | 6 | 41 x 41 | 154 | 120 |
| 20mm x 20mm | 17 | 85 x 85 | 734 | 697 |
| 50mm x 50mm | 18 | 89 x 89 | 816 | 779 |

## Vetscan Hub Mockup Python App - Autofocus On

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dimension | **Version** | **Modules** | **Max AlphaNum** | **Max JSON Example** |
| 10mm x 10mm | 9 | 53 x 53 | 262 | 228 |
| 20mm x 20mm | 21 | 101 x 101 | 1035 | 999 |
| 50mm x 50mm | 19 | 93 x 93 | 909 | 872 |

# Conclusions

## Glare

The glare from the Vetscan’s camera light is causing glare in the captured image. The QR reader is not able to read the label when the glare appears within the image of the QR label.

Masking tape was placed over the camera’s light in order to reduce glare in the images. This improved the resolution but required that the overhead lights in the office be turned on to get enough light on the QR codes.

## 10mm x 10mm QR labels

The current version of the Vetscan’s camera does not have the high resolution needed to read 10mm x 10mm QR labels.

## Python app verses web app

On the Vetscan, the Python app was much better at decoding the labels then the web app was.

## Comparison of results

Below is a comparison of the test results for each test configuration.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Configuration** | **Max Data with JSON text** | | |
| **10mm x 10mm** | **20mm x 20mm** | **50mm x 50mm** |
| Vetscan Hub Mockup Web App | Failed | 419 | 528 |
| Galaxy S10 Cell Phone | 656 | 1839 | 1134 |
| Laptop Web App | Failed | 154 | 600 |
| Laptop Python App | Failed | 262 | 1326 |
| Vetscan Hub Mockup Python App  Autofocus Off | 154 | 734 | 816 |
| Vetscan Hub Mockup Python App  Autofocus On | 262 | 1035 | 909 |

## Autofocus

The autofocus feature is slow, requiring that the QR code be held at the same distance for a few seconds before the camera is in focus.

The Autofocus did help read the higher density QR codes, but the QR code must be held very still for 2 or 3 seconds so that the slow autofocus can move the lens.

# Revision History

Record the results of testing labels at 3 inches (76.2 mm).

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| A beta | 15 JUL 2021 | Bruce Graham | Initial work. The use of the 10 mm x 10 mm QR labels is to be resolved with initial testing. |
| 1 | 22 JUL 2021 | Brian Newberry | 1.4.1 Updated Content of label code. |
| 2 | 26 JUL 2021 | Bruce Graham | 4.5 Added test results Vetscan: 20mm  4.6 Added test results Laptop: 20mm -web app  4.7 Added test results Laptop: 20mm – Python app  4.8 Added test results Galaxy S10 cell phone: 20mm  4.9 Added test results – Vetscan: 20mm with Python app |
| 3 | 28 JUL 2021 | Bruce Graham | Condensed tables for each device.  5 Conclusions - Put the results into a table to make it easier to compare the text configurations. |
| 4 | 5 Aug 2021 | Bruce Graham | 1.7.1 Added table - QR code maximum data capacity  Simplified tables to show only the max QR code desity that each test configuration could measure.  5.4 Added chart - Comparison of results |