VTAP

Configuration Guide

Core Firmware from v2.2.7.0

VTAP50 and VTAP100

Revised February 2025 v3.84

DOT ORIGIN

If you need help to set up or use your VTAP reader, beyond what is contained in this Configuration Guide, then please contact our support team.

Email: **vtap-support@dotorigin.com**

Download the latest documentation and firmware from **https://vtapnfc.com**

Telephone UK and Europe: +44 (0) 1428 685861

Telephone North America and Latin America: +1 (562) 262-9642

If you have any feedback on setting up or using your VTAP reader or this documentation, then please contact our support team. The product is constantly being reviewed and improved and we value feedback about your experience.

Dot Origin Ltd
Unit 7, Coopers Place Business Park, Combe Lane, Wormley
Godalming GU8 5SZ United Kingdom
+44 (0) 1428 685861

Contents

1 Using this guide

This guide is for first-time users of the VTAP50 and VTAP100, and anyone using a VTAP reader with a simple USB connection to a PC. It will help you configure your VTAP reader for a simple applications. Information about maintenance features includes how to update the firmware on your VTAP reader, when a new release is available.

The VTAP50 is a compact version of the VTAP100 which shares most features, but does not have on-board LEDs as standard. Configuration is exactly the same for VTAP50 and VTAP100, but the firmware files are specific to the type of VTAP reader and are not interchangeable.

Advanced users, such as mobile app developers and system integrators, who want to control multiple units remotely and integrate VTAP reader outputs with other systems, will need the additional information contained in the VTAP Application Notes, VTAP Serial Integration Guide or VTAP Commands Reference Guide. All of this is available as online, searchable help in the **VTAP Knowledge Base**.

The **Default Config.txt file** is included as an Appendix, so that you can always revert to factory configuration, should you make mistakes or delete guidance comments you later need.

You must be running core firmware v2.2.7.0 or later to access all of these commands or settings. Visit **https://vtapnfc.com** if you need to download the latest firmware version. Instructions to help you **Update firmware** are included in section of this guide.

If you need help beyond what is contained in this guide please contact **vtap-support@dotorigin.com**.

2 How a VTAP reader works

Simply tap your smartphone against the VTAP reader. Your mobile NFC pass will be read and data sent to the connected PC. Since the VTAP reader includes automatic pass selection, you do not need to open your phone and select the pass.

Figure 2-1 VTAP100-USB in use

When a VTAP reader is connected to a computer it appears as a generic mass storage device (like a memory stick). It can be set to send keyboard inputs to the PC, or behave as a virtual serial COM port device. In the case of a VTAP PRO reader it could send that data to a cloud platform.

In order to configure your VTAP reader, you simply edit or create text files. These will be automatically read and control the operation of the VTAP reader. This approach allows you to easily specify pass reading parameters and define the format to send data (per interface). You can optionally define LED or buzzer actions required when a pass is detected.

By default the VTAP reader is fully upgradable in the field, using a file-based method for distributing firmware updates in a secure manner. However, the VTAP reader hardware can be optionally locked, before deploying the unit, so that operation is no longer easily changed.

Other files are generated automatically by the VTAP reader, to provide you with status information. This data can be read as and when needed, from an attached PC, either in text files or over a serial connection.

(The Wiegand model, VTAP100-PAC-W, is also configured over USB from a PC. After configuration it is connected to an access controller, and will send pass data over the Wiegand interface to the controller, like any other reader.)

2.1 Configure a VTAP reader

This is an overview of the steps you need to take in configuring a VTAP reader.

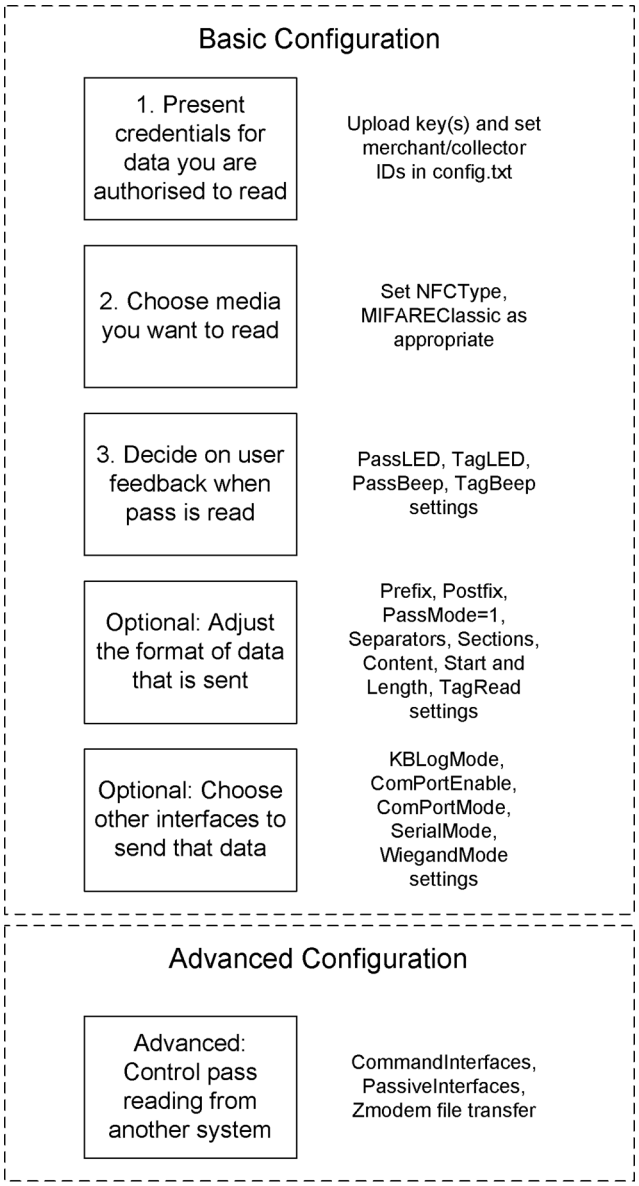


Figure 2-2 VTAP reader configuration steps

The VTAP reader is very flexible, in supporting different data formats for different media and different interfaces. So, we would recommend you follow an incremental approach to setting up and testing your settings.

1. First, check that the VTAP reader delivers **Default operation**.
2. Then set up for reading mobile passes only, with a keyboard emulation output, which is described in **Start reading your own passes**.

3. Only then add in capability to **Read cards or tags** or **Connect to other systems** using other interfaces.

If you are an integrator or developer, and need to go further in controlling VTAP50 and VTAP100 behaviour from other systems, you will find that information in the VTAP Application Notes, VTAP Serial Integration Guide and VTAP Commands Reference Guide.

2.2 Default operation

Before anyone changes the configuration from its default, you can confirm that the unit is working.

These steps demonstrate that the hardware can detect and interact with an OriginPass demo mobile NFC pass, which is ready to work with the default configuration of your VTAP reader.

1. Obtain an OriginPass from Dot Origin by visiting **https://originpass.com/VTAP/** and add it to Google or Apple Wallet. (You will require a username and password - contact **vtap-support@dotorigin.com** to get these.)
2. Connect the VTAP reader to your PC, using a USB cable.
3. Open a text editor, such as Windows Notepad.
4. When you tap the OriginPass on the VTAP reader:
   - Pass contents will be displayed in the open text editor, through keyboard/barcode emulation.
   - The feedback LEDs on the VTAP reader PCB will flash green. [VTAP100 or VTAP50 v2 only]
   - Your smartphone may signal with a buzz or beep.

Note: Some Android phones will only interact if their screen is on, although it does not need to be unlocked. You may need to enable NFC in the settings for the smartphone.

Note: If the pass detected does not match the key and ID on the VTAP, or is moved away too quickly to be read, the pass contents displayed may be an 8 digit random hex string, such as '08E22AC1', different on each presentation. OriginPass contents will be a consistent string, such as '3~ffymeK9f_mziYtA6~53999301628695~Valued'. Any separator, such as '~' or '|', will depend on your keyboard language settings. (See VTAP Commands Reference Guide for option to ignore random UIDs if needed.)

Note: If local security settings prevent or limit the use of removable storage devices, or the connection of additional keyboards, an administrator may need to alter those permissions.

3 Start reading your own passes

To read any mobile NFC pass, you will need to provide your pass reading parameters in the `config.txt` file. This means a collector ID or merchant ID and ECC keys. These allow you to read and decrypt pass data that is held by your users, on their smartphones. (There is a VTAP Application Note which explains more about ECC key pairs and how to generate your own keys.)

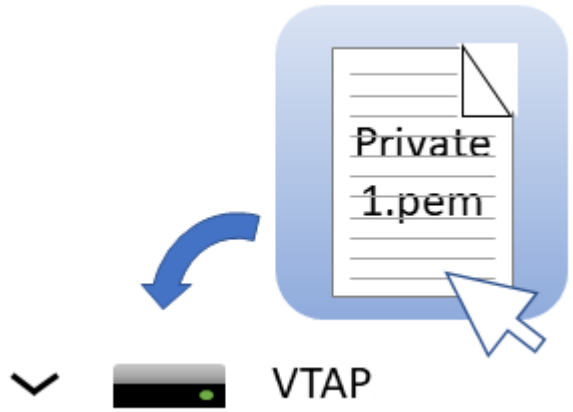
This first time, you will need to connect the VTAP reader to your PC, using a USB cable. (If needed, you can make changes remotely in future over a virtual COM port or serial port, see VTAP Serial Integration Guide.)

Step 1: Upload private key file(s) to your VTAP reader

1. Ensure these are ECC private key(s). Each is stored in a file with the name `private#.pem`, following the `.pem` format, where `#` is replaced with a number from `1` to `6`, matching the key slot you will save it in. (The demo passes are accessed using the key in KeySlot 6, so don't overwrite this one unless you are finished with Dot Origin demo passes.)

   Note: A VTAP reader cannot use more than 6 private key files.

2. Load your keys by copying these files onto your VTAP reader, which shows up in the file system of your PC as a mass storage device.



Note: When you reboot the VTAP reader your key will have been stored in hardware, and will no longer be listed as a file on the device. You can confirm key file(s) have been loaded when you **Check status in Boot.txt**. If the key file does not disappear and there is an error in `Boot.txt`, check your `.pem` file as it is likely it did not adhere to the standard - perhaps it was not an ECC key?

Step 2: Declare Merchant ID(s)/Collector ID(s) in the `config.txt` file

1. Open the file `config.txt` in a text editor (such as Windows Notepad). It already contains parameters for accessing the demo passes, prefixed `VAS1` and `ST1`, both relying on KeySlot 6. You can overwrite these, or keep them in addition to your own pass reading parameters.
2. Add your pass reading parameters in the `config.txt` file to access up to 6 Apple VAS and up to 6 Google Smart Tap IDs, and identify the keys to be used in each case.

   Note: Although the VTAP reader supports multiple IDs, Apple expect most users will only use one. Multiple collector IDs are not supported by Android, which means you cannot request more than one Collector ID from Google. Only one should be live at any one time. Multiple IDs is an advanced feature that should be used with care. The VAS# and ST# numbers define the order in which IDs will be requested from Apple or Android phones respectively. The lowest numbered ID will be requested first, then continuing in ascending numeric order. (There is a VTAP Application Note on Multiple Passes which explains more.)

   Put each parameter on a new line. Order of parameters does not matter to the VTAP reader, but could help other people who need to edit the file. Start any comment lines in the `config.txt` file, that the VTAP reader should ignore, with a semicolon. Each parameter should only appear once - if it accidentally appears more than once then only the last instance will take effect.

   Example: Settings in `config.txt` to interact with both Apple VAS and Google Smart Tap mobile passes

```
!VTAPconfig

VAS1MerchantID=<your merchant ID>
VAS1KeySlot=1
; This says use the key added as file 'private1.pem' to read and
; decrypt any pass connected to your merchant ID on an Apple iPhone

ST1CollectorID=<your collector ID>
ST1KeySlot=2
ST1KeyVersion=1
; This says use the key added as file 'private2.pem' at key version 1
; to read and decrypt any pass connected to your collector ID
; on an Android phone
```

3. Save the amended config.txt file and these changes will take effect immediately. (A small number of changes to the `config.txt` file require a reboot to take effect, for instance to the status of the virtual COM port, but these are highlighted in later sections).

Note: If a `VAS#KeySlot` parameter is omitted, or set to 0, then all available keys will be automatically tried to choose the right key. If the data received by the VTAP reader cannot be decrypted, the Apple iphone will register a pass read, but the data will not be output.

Note: If an `ST#KeySlot` parameter is omitted, or set to 0, then authentication will be omitted and decryption will not be performed. In this case, Google Smart Tap data will be received and sent on by the VTAP reader, only if the pass does not require authentication by the terminal.

3.1 Edit `config.txt`

`config.txt` is the main file controlling operation of the VTAP reader. To change any operation:

1. Connect the VTAP reader to your PC, using a USB cable.
2. Browse to the VTAP reader mass storage drive.
3. Open the file `config.txt` file in a text editor (such as Windows Notepad).

> **Note:** If local security settings prevent or limit the use of removable storage devices, or the connection of additional keyboards, an administrator may need to alter those permissions.

The groups of parameters in your `config.txt` file are summarised in the following figure:

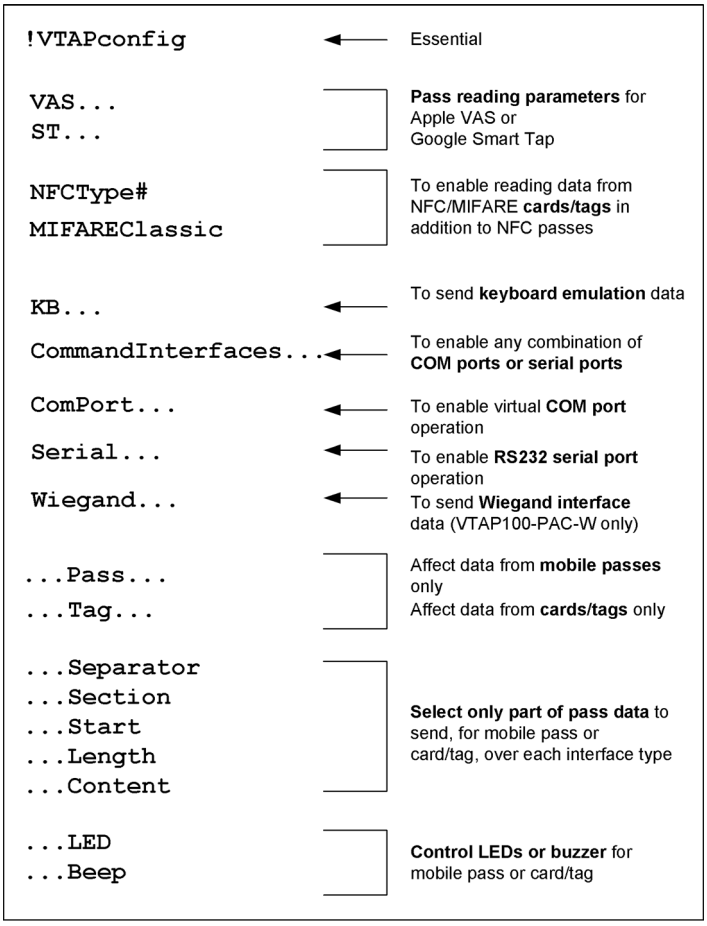


Figure 3-1 Overview of `config.txt`

Changes to `config.txt` usually take effect as soon as the file is saved. (The only exception is when you enable or disable the virtual COM port. In this case you need to reboot the VTAP reader before it takes effect, see **Send pass data over virtual COM port**.)

Note: If you make changes to your `config.txt` file from which you cannot recover, copy the `config.txt` content from the Appendix **Default Config.txt file** and paste it into the `config.txt` file on your VTAP reader. This will return the VTAP reader to factory default settings.

3.2 Send pass data as keyboard emulation over USB

Keyboard emulation over the USB interface means that the VTAP reader sends mobile NFC pass data exactly as if it were entered on a keyboard, or like some barcode scanners for paper or plastic passes.

Note: The VTAP keyboard emulation mimics a US English keyboard layout. Data sent is a down key press, followed by an up key press for each character. The key press ID reflects the position in the keyboard grid, which your operating system converts to characters based on your language setting. If you are not able to set a language per device (as with Microsoft Windows) you might choose to use a keyboard map on the VTAP reader to correct certain key presses. (Find out more about the setting `kbmap` in the VTAP Commands Reference Guide). Alternatively, use a virtual COM port interface instead, which works with ASCII data, therefore avoiding this issue.

Keyboard emulation is enabled with `KBLogMode=1` in the default configuration for VTAP100-USB, for all mobile passes or NFC tags/cards that the VTAP reader can read, using `KBSource=A1`. So if you open a text editor on the PC connected to the VTAP reader, any successful pass read will result in pass contents appearing in that open text editor.

If you need to switch off sending pass data as keyboard emulation set `KBLogMode=0`.

There are adjustments you can make to the way that data is sent as a keyboard emulation, using other parameters prefixed `KB` in your `config.txt` file. These include adding extra prefix or postfix characters with `KBPrefix` or `KBPostfix` to the data that is sent, or choosing to **Extract only part of the pass data**.

Note: If local security settings prevent or limit the use of removable storage devices, or the connection of additional keyboards, an administrator may need to alter those permissions.

3.3 Extract only part of the pass data

By default the whole pass data payload will be sent, using `KBPassMode=0`, but you can choose to send only part of the pass data, by setting values in `config.txt`.

Note: The examples in this section use the `KBPass` prefix on all of the parameters, to control the behaviour of mobile pass NFC data being sent as a keyboard emulation. The same parameters work in the same way with prefixes such as `ComPortPass` or `WiegandPass` to control the mobile pass data sent over those other interfaces.

To extract only a part of the data use `KBPassMode=1`. The following examples extract data in different ways:

Example: Changes to `config.txt` to extract a fixed length of data from a particular start point

```
!VTAPconfig

KBPassMode=1
KBPassSeparator=|
KBPassSection=2
KBPassStart=5
KBPassLength=10
```

Here you specify which section separators to find, and count sections and characters from 0, to extract the right data. In this case:

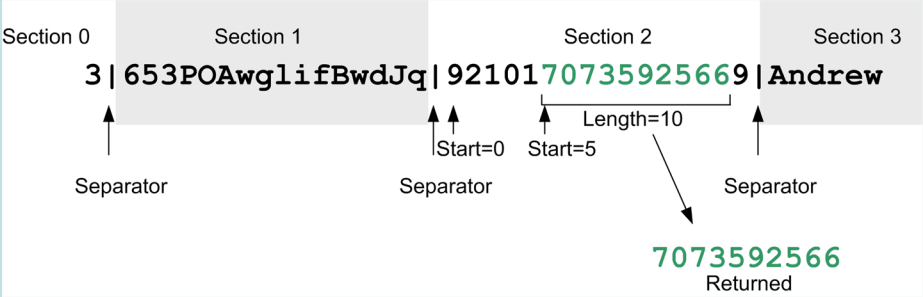


Figure 3-2 Separator |, Section 2, Start 5, Length 10

Example: Changes to `config.txt` using ContentMode 1 and two levels of data separators to extract data

```
!VTAPconfig

KBPassMode=1
KBPassSeparator=|
KBPassSection=2
KBPassContentMode=1
KBPassContentSeparator=%
KBPassContentSection=1
```

In this case there are separators to identify sections **and** content separators within the target section:

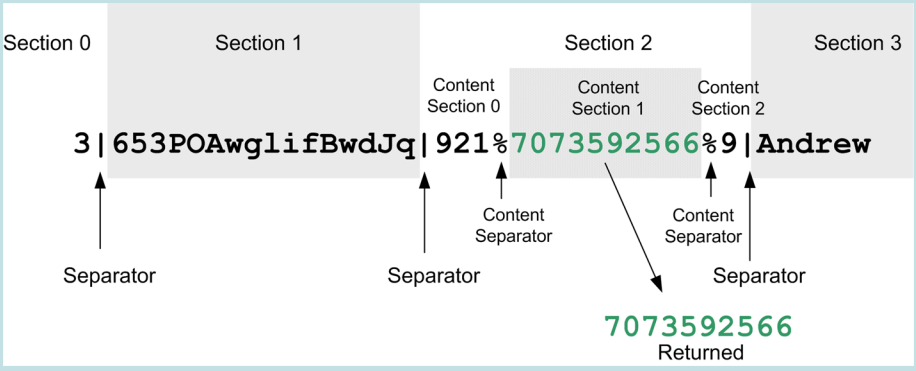


Figure 3-3 Separator |, Section 2, Content Separator %, Section 1

Example: Changes to `config.txt` to extract part of the full pass payload for Wiegand interface [VTAP100-PAC-W only]

```
!VTAPconfig

WiegandMode=1           ; Enable Wiegand interface
WiegandPassMode=1       ; Choose to extract only
                        ;  a part of the pass payload
WiegandPassSeparator=|  ; Set the separator character the VTAP should
                        ;  use to separate the payload into sections
WiegandPassSection=2    ; Section number to read based on that
                        ;  WiegandPassSeparator
PassWiegandBits=32
```

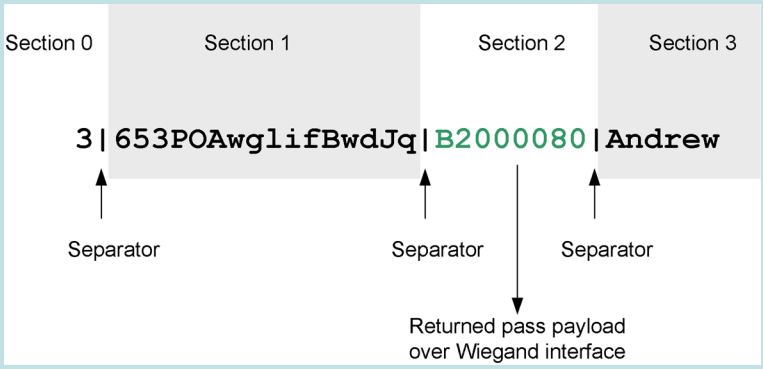


Figure 3-4 Separator |, Section 2 for Wiegand data (on VTAP100-PAC-W only)

Full pass payload:
3|653POAwglifBwdJq|B2000080|Andrew

Pass payload sent over the Wiegand interface:
B2000080

3.4 Control LED and buzzer feedback

The VTAP100 has two diagnostic LEDs, red and green, which provide a heartbeat on the board, for factory use only. There are also user feedback LEDs, on the reverse side of the board, which are visible through a window in the case. These can show any hex RGB colour. There is also a buzzer on both VTAP50 and VTAP100. You control how the LEDs and buzzer react to passes, cards or tags by setting parameters including `LED` or `Beep` in the `config.txt` file.

Note: Not all VTAP50 have built in LEDs.

Example: Control of LED and buzzer in default `config.txt`

```
!VTAPconfig

LEDSelect=1
LEDDefaultRGB=1EEBCF
PassLED=00FF00,100,100,2
PassBeep=100,100,2
TagLED=00FF00,100
TagBeep=100
```

`LEDSelect=1` is to make use of the pair of LEDs that are visible through a compact case (CC) for user feedback, but you could use `LEDSelect=2` to use the LED that is visible through a square case (SQ) instead. (When driving VTAP50 external LEDs use `LEDSelect=0` for external RGB (common cathode), `LEDSelect=1` or `=2` for external RGB (common anode), or `LEDSelect=3` for external or on-board serial LEDs where fitted.) `LEDDefaultRGB=1EEBCF` sets the colour that the feedback LED(s) show constantly, but can be set `=0` if you would rather LEDs were off when a pass is not being read.

`PassLED` chooses the hex colour for the LEDs to flash when there is a mobile pass read, along with the flash on time, interval between flashes and number of repeats, the example chooses two 100ms flashes with 100ms intervals in between. `TagLED` sets the LED response to a card/tag read in the same way. In the example the response to a card/tag is set to be the same colour, but a single 100ms flash (interval between flashes and number of repeats can be omitted if not needed). `StartLED` is set up in the same way to define an LED action on startup.

`PassBeep` sets a beep on time, interval between beeps, and a number of repeats for the buzzer when there is a mobile pass read. Buzzer frequency can be changed on hardware from VTAP100 v5 or VTAP50 v2 onwards. More complex beep sequences are discussed in the section **Playing beep sequences or tunes** below. In this example there will be a double beep for a pass read. `TagBeep` sets the buzzer response to a card/tag read in the same way and `StartBeep` is set up in the same way to define a beep on startup. The example sets a single beep response to a card/tag read. (Again you can omit the interval between beeps and number of repeats, for a single beep of the specified duration).

3.4.1 VTAP50 serial LEDs

On VTAP50 v2 boards with firmware from v2.1.11.2, a chain of serial LEDs can be connected to the serial LED connection on the expansion header.

A chain of serial LEDs supports more complex LED options and requires a special text file to control their behaviour, to control the colour of each LED separately and create combinations or sequences of LED patterns.

`LEDMaxSerial` defines the length of the attached LED chain. The current maximum value for external serial LEDs is 255. Please contact **vtap-support@dotorigin.com** if your requirement is for more than 255 LEDs.

Note: Where status LEDs are fitted alongside a chain of serial LEDs, they will duplicate the behaviour of the first two LEDs in the chain.

There is an example `leds.ini` file in the Appendix.

Be aware that, if your requirement is very simple, a small section of the `leds.ini` file can be included in the `config.txt` file, to avoid the need for a separate file. For instance:

Example: Changes to `config.txt` to set a simple sequence for VTAP50 v2 serial LEDs without using an `leds.ini` file

```
[leds.blue]
000010=1-10
[end.leds]

LEDDefaultRGB=000000:leds.blue@config.txt
```

This example defines a sequence where serial LEDs 1 through 10 in the series show a dark blue (#000010).

Example: To set a simple sequence for VTAP50 v2 serial LEDs in an `leds.ini` file

Type these lines in a text file called `leds.ini`.

```
[leds.red]
FF0000=1-10

[leds.green]
00FF00=1-10

[leds.blue]
0000FF=1-10

[seq.rgb]
repeat=forever
frame=500,leds.red
frame=500,leds.green
frame=500,leds.blue
```

Add these lines to `config.txt` to call the text file `leds.ini`.

```
LEDDefaultRGB=FFFFFF:seq.rgb@leds.ini
```

This example defines a sequence where serial LEDs 1 through 10 in the series show red, then blue, then green on a loop.

3.4.2 Playing beep sequences or tunes

On VTAP100 v5 or VTAP50 v2 hardware onwards, the buzzer frequencies can be changed by specifying a sequence, in a very similar way to specifying sequences for serial LEDs.

A configuration section name and/or file name can follow any of the standard buzzer commands (`TagBeep`, `PassBeep`, `PassErrorBeep` or `StartBeep` to describe a sequence of beeps.

When a section is specified but no file, it is assumed the beep sequence will be defined in an `leds.ini` file, potentially alongside serial LED sequence definitions.

If your requirement is very simple it can be defined within the `config.txt` file, to avoid the need for a separate file. This is shown in the first example below. A more complex beep sequence can be more easily managed in a separate file. The second example uses a text file named `beeps.ini`, though any short file name could be used (following the `xxxxxxxx.xxx` format, up to 8 characters with a 3 character optional extension).

Example: Changes to `config.txt` to set a simple sequence for the TagBeep response, without using another file

```
[tagbeep]
repeat=2 ; repeat sequence twice
tempo=120 ; set to 120 beats-per-minute
note=100,440 ; 440Hz for 100ms
note=50,220 ; 220Hz for 50ms
[end.tagbeep]

TagBeep=100:tagbeep@config.txt
```

This example plays a short 4 note sequence. (Note: If the `tagbeep` section could not be found, for example if the section name was mistyped, the `TagBeep` action would default to a single 100ms beep at default frequency, simply ignoring the sequence part of the configuration line.)

Example: To set a simple beep sequence in an `beeps.ini` file

Type these lines in a text file called `beeps.ini`.

```
[tagbeep]
repeat=2 ; repeat whole sequence twice
tempo=120 ; set to 120 beats-per-minute
note=w,C6 ; whole note of C6
note=16s,A#5 ; 16 sixteenth note lengths of A#5 (same as 1w)
tempo=90 ; slow down the tempo for the remaining notes
note=250 ; rest for 250ms
note=w ; rest for one whole note
note=7e,Gb6 ; seven eighth notes of Gb6
note=100,440 ; 440Hz for 100ms
note=100,C4 ; C4 for 100ms
note=q,440 ; 440Hz for one quarter note
```

Add these lines to `config.txt` to call this section in the text file `beeps.ini`.

```
TagBeep=100:tagbeep@beeps.ini
```

(Note: If the `beeps.ini` file or `tagbeep` section cannot be found, the `TagBeep` action would default to a single 100ms beep at default frequency, simply ignoring the sequence part of the configuration line.)

This example includes a variety of notes, specified in a variety of different ways. The syntax used is described below.

Key elements in a note sequence:

- The `repeat` is a number of times to play the whole sequence, or use `forever` to continue until another sequence is started.

- The `tempo` sets the beats-per-minute (defaults to `100`).
- Each `note` is played in the order listed. Each one is specificed as `note=<duration>,<frequency>,<volume>`
  - Duration can be specified as number in milliseconds, or as a fraction of a beat, calculated from the tempo setting, denoted by `w` (whole), `h` (half), `q` (quarter), `e` (eighth) or `s` (sixteenth). You can have multiple of the fractional units - so `3h` would be a duration equivalent to 3 half notes.
  - Frequency can be specified as number in Hz or as a note name such as `C#6` where the note here `C` is named as a letter, `#` added for sharp or `b` for flat, and the number refers to the octave, from `0` to `8` (defaults to `6` if omitted).

    Setting frequency to `0` switches the buzzer off, which can turn a note into a rest of a specified duration.
  - Volume can be set between `0` and `127`, though this is not particularly sensitive (defaults to the maximum `127` if omitted).

VTAP can alternatively play very simple MIDI files. It is limited to one channel of a format 0 file, containing a single track. To use a MIDI file, use the section name 'midi' and specify a MIDI file as the file name.

Example: To play a simple single channel MIDI file

Add these lines to `config.txt` to call a MIDI file called `song.mid`.

```
PassBeep=100:midi@song.mid
```

This example will play channel 0 of the midi file. Using the section name `midi4@` would instead play channel 4 from the named MIDI file. (Note: If the MIDI file cannot be found, the `PassBeep` action would default to a single 100ms beep at default frequency, simply ignoring the sequence part of the configuration line.)

4 Read cards or tags

VTAP50 and VTAP100 are primarily designed to read mobile passes, not cards and tags. You can choose to allow VTAP reader to read particular NFC or MIFARE® Classic card or tag types in addition to mobile passes, by making changes in `config.txt`.

Example: Changes to `config.txt` to allow NFC or MIFARE Classic card or tag types to be read

```
!VTAPconfig

NFCType2=N
NFCType4=D
NFCType5=U
MIFAREClassic=B

; Use =U or 1 to permit a particular card type UID to be read
; =N or 2 to read NDEF records (NFC Type 2 or 4, not MIFARE Classic)
; =B or 3 to read/decode memory blocks on MIFARE Classic, NFC Type 2 or 5
; =D can be used for NFCType4 to read DESFire cards
; =0 is disabled (default)
```

Note: Don't forget that you also need a setting such as `KBSource=A1` to send this card/tag data, in addition to mobile pass data, over your preferred interface.

This table will help you find valid setting combinations:

Card/tag type	Compatibility			
	UID (=U)	NDEF (=N)	Block data (=B)	Secure data (=D)
NFC Type-2/MIFARE Ultralight/NTAG	✓	✓	✓ (AES)	
NFC Type-4/DESFire/ Host Card Emulation	✓	✓		✓ (EV1, DES/AES)
NFC Type-5/ICODE	✓		✓ (up to 255 bytes)	
MIFARE Classic	✓		✓ (single block)	

Table 4-1 Types of data from different card/tag types

MIFARE is a registered trademark of NXP B.V.

There are advanced settings such as `NDEFTagExtractType` and `NDEFTagExtractID` described in the VTAP Commands Reference Guide, which allow you to further extract the data that will be read.

When blocks of data are read from cards or tags, you may also want to set some of the parameters prefixed `TagRead` to **Extract only part of the card or tag data**.

You can also upload keys to read secured data from DESFire cards. This option is addressed in VTAP Application Notes.

4.1 Extract only part of the card or tag data

You can extract parts of the card or tag data, or choose to send UID, NDEF records or block data, by setting values in `config.txt`.

There are three types of data you might want to extract from a card or tag:

- **UID** - The unique identifier of the card/tag. This is typically either 4 or 7 bytes of data (32 or 56 bits). It is usually stored in a fixed location within the first block of the card or tag's memory depending on tag type.
- **NDEF** - A stream of NDEF (NFC Data Exchange Format) records encoded and stored in the card or tag's memory area according to the NFC Forum specification for the tag type. Extracted by NDEF record type or ID.
- **Block data** - A sequence of up to 16 bytes of data (or 255 bytes for NFC Type 5) stored in a particular location in the card or tag's memory area and accessed directly by specifying the block number, offset and length within the block.

The simplest situation is to set `NFCType#=U` or `MIFAREClassic=U` to send the whole UID or `NFCType#=N` to send all NDEF records.

To extract part of the block data you need `NFCType=B` or `MIFAREClassic=B`, then identify which data block to read, using settings with a `TagRead` prefix.

The following example extracts data in this way:

Example: Changes to `config.txt` to extract a fixed length of data from a particular start point

```
!VTAPconfig

MIFAREClassic=B
TagReadBlockNum=8
TagReadKey=123ABC456DEF
TagReadKeyType=B
TagReadOffset=5
TagReadLength=4
TagReadFormat=d
```

Here you specify which block to read (with the key and key type to use, if needed), the offset within the block to start taking data, counting from 0, and the length of data to take. The output format is set to 'd'ecimal. So, in this case:

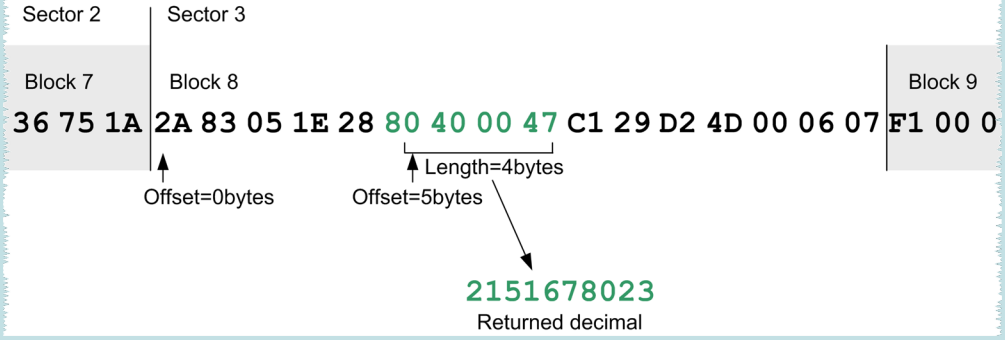


Figure 4-1 Block 8, Offset 5, Length 4 bytes

5 Connect to other systems

If you want to receive the pass data collected by the VTAP reader in other systems, you will need to use a virtual COM port, or serial RS-232, RS-485 or Wiegand interface, if your hardware permits.

> **Note:** The serial RS-232 interface is an option on the -OEM hardware only. The Wiegand interface is only available on the VTAP100-PAC-W model and an RS-485 interface is only available on the VTAP100-PAC-485 model.

The basic steps to **Send pass data over virtual COM port** and for **Using a VTAP reader with the latest Read-a-Card software** are included in this guide.

There are a number of other advanced features which may be needed by integrators, for more complex applications. For information about other serial interfaces, and using those interfaces in passive mode, you will need to refer to the VTAP Serial Integration Guide.

5.1 Send pass data over a virtual COM port

A virtual COM port setting for the USB interface will mean that the VTAP reader is treated by the connected PC as a COM port, as well as a mass storage device. The COM port will be active and will send any pass data received as soon as it is read.

To help receive pass data over the virtual COM port in a particular format, you can add extra prefix or postfix characters to the data that is sent, using `ComPortPrefix` or `ComPortPostfix` settings. You can also set `ComPortPassMode=1` and choose to **Extract only part of the pass data** using the same settings used to manipulate pass data sent over keyboard emulation, but this time starting the settings with `ComPortPass...` rather than `KBPass...`.

> **Note:** If you would rather receive pass and status data only in response to a request over the COM port refer to the VTAP Serial Integration Guide on serial control and using the virtual COM port in passive mode.

5.2 Using a VTAP reader with the latest Read-a-Card software

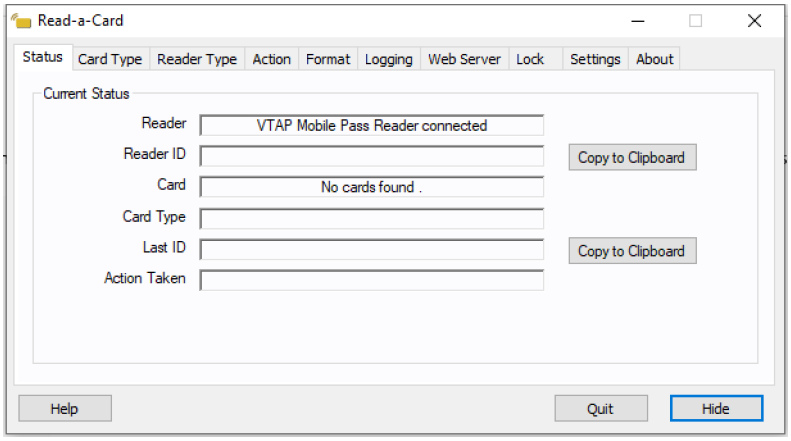
Read-a-Card software from Dot Origin can act as a VTAP software agent, providing a simple means of transferring pass or card read information from your VTAP reader to your web site or application, when you enable a Read-a-Card Web Server.

Read-a-Card is free to use with any VTAP reader. When a VTAP reader using firmware v2.1.12.7 or later is connected to a PC running Read-a-Card software v3.4.5 or later, Read-a-Card will be automatically licensed for use.

If you have older VTAP firmware or Read-a-Card software, we suggest that you update to the latest versions. You can download the latest version of Read-a-Card here: **https://vtapnfc.com/downloads/tools/Read-a-Card.msi**

(If you are unable to move to the latest VTAP firmware or Read-a-Card software, please contact **vtap-support@dotorigin.com** for help.)

When a VTAP reader is connected to your PC and you open Read-a-Card software, you should see "VTAP Mobile Pass Reader connected":



Note: If you see "Waiting for reader" on the Read-a-Card **Status tab** and you should check any `ComPort...` settings included in your `config.txt` file and adjust them, if needed, to match the example below. Save the updated `config.txt` file, then reboot the VTAP reader, by briefly removing power. The VTAP reader needs its virtual COM port enabled to work with Read-a-Card.

For VTAP firmware from v1.1.9.1 onwards, you don't need to add any special lines to `config.txt` for the VTAP reader to work with Read-a-Card. But if any `ComPort...` settings are included in your `config.txt` file, they should be set to the values in the example below.

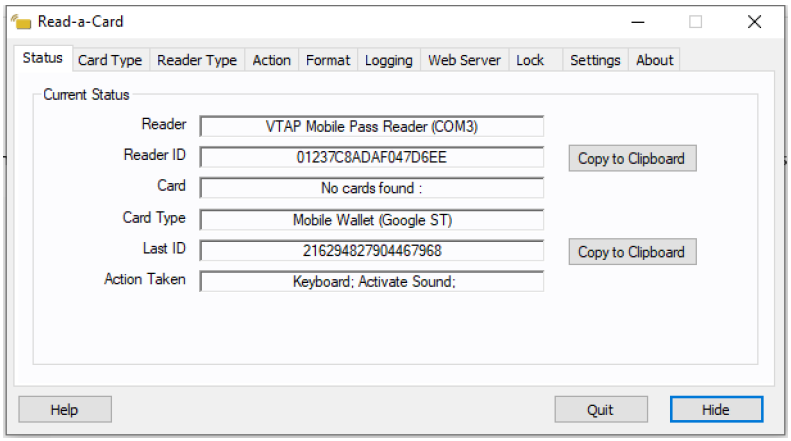
The example below shows how you might want to configure your VTAP reader to work well with Read-a-Card.

Example: Extra lines for `Config.txt` file
to make VTAP reader send demo pass UID over virtual COM port

```
ComPortPassMode=1
ComPortPassSeparator=|
ComPortPassSection=2
```

Note: The last three lines of this example assume the demo pass format generated by Pronto, where sections are separated by '|' characters and the UID is in section 2. Passes from other providers may use other separators, or the entire payload as a number.

When you present a pass to your VTAP reader. You should see that it is detected on the Read-a-Card **Status tab**, like this:



If you need a keyboard emulation output, when using a VTAP reader with Read-a-Card, it is recommended that you use Read-a-Card's keyboard emulation feature.

> **Note:** If you enable a keyboard emulation output from your VTAP reader (your `config.txt` file contains `KBLogMode=1`) each pass or card read will interact with the Read-a-Card user interface, which could be confusing and cause Read-a-Card to minimise into the system tray.

The VTAP reader is recognised by Read-a-Card software as a proprietary reader, 'VTAP Mobile Pass Reader' type on the **Reader Type** tab. This gives you an expanded facility to check and manipulate data stored in passes, tags or cards.

To set up web server functionality consult the Read-a-Card User Guide, which opens when you select the 'Help' button within Read-a-Card.

5.3 More advanced features

There are several advanced features which may be needed by integrators, for more complex applications, which are explained in detail in the VTAP Serial Integration Guide and VTAP Application Notes. These include:

- Use the virtual COM port in passive mode

  If you want the VTAP reader to send information from pass/card reads only in response to requests (not automatically), you will need a virtual COM port in passive mode.

  There are a set of commands available, that the VTAP reader will understand, when received over the virtual COM port in passive mode. They can request particular data, direct data to be sent over another interface, alter configuration of the VTAP reader, lock or unlock the device with a password, or drive the user feedback LEDs [VTAP100 or VTAP50 v2 only] and buzzer.

- Transfer files in passive mode

  If you cannot access the VTAP reader as a mass storage device on your PC, you can use the Zmodem protocol to transfer files, such as a firmware update, key update or configuration file. By default the VTAP reader is always listening for Zmodem commands.

  You could use a free application such as TeraTerm: Simply choose to **Transfer** a file, select **Zmodem**, **Send** and choose the file to send. It is anticipated that developers will prefer to implement their own code, with actions that trigger a Zmodem transfer. Example code for these situations can be provided in C/C++/Python/Java.

- Use serial interfaces

  An RS-232 serial interface is available on some OEM hardware and RS-485 on others. These interfaces are used in a similar manner to others, but again with some special settings.

- Use a Wiegand interface

  There is a model called the VTAP100-PAC-W, which can be integrated into access control applications. Control of the Wiegand interface also requires specialist settings.

6 Maintenance features

This section addresses less frequent operations such as checking status, updating firmware and resetting the device.

6.1 Check status in BOOT.TXT

Inspecting BOOT.TXT will give you essential information about your VTAP reader set up, at time of last reboot, which might be helpful when troubleshooting.

```
VTAP50
Boot time: 1970/01/01 00:00:00
Firmware: V2.2.4.0
Storage: Dataflash
Status: 0
Hardware: 2.00
Expansion: None
VCP enabled
NCI: 0471125005-8C00
Serial number: 563230-29F1F2ADC2261743BDB1B8405FFD603E
API level: 4
AppKeys used: 123------
KeySlots used: ----56
```

Figure 6-1 Example VTAP100 v5VTAP50 v2 BOOT.TXT file

You are most likely to need:

- 'Serial number' (or 'ATCA' on VTAP100 v4a or earlier) - the hardware serial number for your VTAP reader.
- 'VTAP label' (if set) - the assigned serial number for your VTAP reader, which matches that on its label. This will not show if not set.
- 'Firmware' - the VTAP reader core firmware version in use. You will find the latest firmware versions at **https://www.vtapnfc.com/download/**
- 'Hardware' - the VTAP reader hardware version in use.
- 'API level' - indicates which serial or OSDP API commands are supported.
- 'KeySlots used:' - Indicates the ECC private keys loaded on the VTAP to access VAS or Smart Tap passes. Helps you check whether you have uploaded the necessary ECC private keys, which can be unclear as the files are deleted when they are uploaded. These two examples show how to read this information:
  - 'KeySlots used:------' shows that no keys have been uploaded.
  - 'KeySlots used: 12--56' shows that key files 1 and 2 have been successfully uploaded, in addition to the defaults 5 and 6.

- 'AppKeys used:' Indicates the application keys (if any) uploaded to the VTAP for any other applications, such as keys loaded to use with DESFire applications.
- 'VCP enabled', if included - indicates that the virtual COM port has been enabled.
- 'Status' - should be 0 if operating normally, anything else indicates an error state.
- 'Expansion:' shows the name of the expansion board (if any) connected to the VTAP, for example: 'VTAP100W' for a Wiegand expansion board.
- 'Boot time' - The time at boot, which defaults to 1970/00/00 00:00:00 if power is removed to reboot.

If the configuration has been locked the `BOOT.TXT` file will end with the words LOCKED S/W or LOCKED H/W.

6.2 Update firmware

To update the firmware on a VTAP reader:

1. Copy the core firmware image file `vtapware.dat` to the VTAP mass storage device. (It will be called `firmware.dat` for VTAP100 if the hardware version is v4 or earlier. You can identify the hardware revision if you check status in BOOT.TXT or use the `?b` command over a serial interface.)

   Note: The configuration process is the same for VTAP50 and VTAP100, but files on a VTAP reader can be specific to the type and hardware version of VTAP reader and are not always interchangeable.

2. Reboot the device, either by using the `?REBOOT` command over a serial interface, by briefly disconnecting power.

   There will be a delay of a couple seconds when the VTAP reader boots up again and performs the update, then it will continue to operate as normal.

3. You can **Check status in BOOT.TXT** to see if the firmware version is as you expect.

If the new firmware does not work, reload the previous firmware.

6.3 Software lock to prevent local firmware or configuration change

You can lock the VTAP reader so that its firmware and configuration cannot be changed. You can either do this in software, or simply disable the mass storage device in hardware. A **Hardware lock to prevent firmware or configuration change** would mean no files from the VTAP reader are visible.

A software lock will ensure you can still read `boot.txt` locally. You will still able to make changes to firmware or configuration over the command interface on any serial ports that have been enabled (including the USB virtual COM port), if you have the password. There is information about this in the VTAP Serial Integration Guide.

To set the software lock:

1. Place a text file called `lock.txt` on to the VTAP reader mass storage device containing the `lock` command.

   Example: Content of `lock.txt` to lock the VTAP reader and set a password for later unlocking

   ```
   !VTAPlock
   lock=APa55word

   ; this sets the password to "APa55word" and locks the VTAP reader
   ```

2. When you save this file, and **reboot** or power cycle the VTAP reader, it will enable the software lock.

Note: The `lock.txt` file content is never visible in the VTAP reader file system and the password is hashed and stored securely in the VTAP reader. `BOOT.TXT` tells you that the software is locked.

Note: It may appear that you can edit `config.txt` and other files, despite a software lock in place, but these edits will be silently ignored by the VTAP reader and will be lost from the Windows cache on remount, reboot or power cycle.

When you need to remove the software lock:

1. Place another text file called `lock.txt` on to the VTAP reader mass storage device containing the `unlock` command.

Example: Content of `lock.txt` to unlock the VTAP reader

```
!VTAPlock
unlock=APa55word

; this offers the password "APa55word" to unlock the VTAP reader
```

2. If the password quoted matches the one used when setting the lock, when you **reboot** or power cycle the VTAP reader, it will be unlocked.

6.4 Hardware lock to disable USB mass storage device

You can lock the VTAP reader so that its firmware and configuration cannot be changed. You can either do this in software, or simply disable the mass storage device in hardware. or VTAP100-PAC-485

A **Software lock to prevent firmware or configuration change** prevents changes but leaves the file system readable. A hardware lock means that the VTAP reader will no longer be detected as a USB mass storage device. (It will still behave as an HID keyboard device and, if enabled, the virtual COM port will behave as a composite USB device consisting of HID keyboard and USB virtual COM port.)

Note: On VTAP50 reader boards or modules, although the contact is present, connector pins and jumpers will need to be supplied.

Connect a jumper across LOCK on the PCB to lock the device, preventing firmware or configuration changes via the mass storage device. (You may still update firmware or configuration via command interfaces, virtual COM port or serial ports, if they are enabled.)

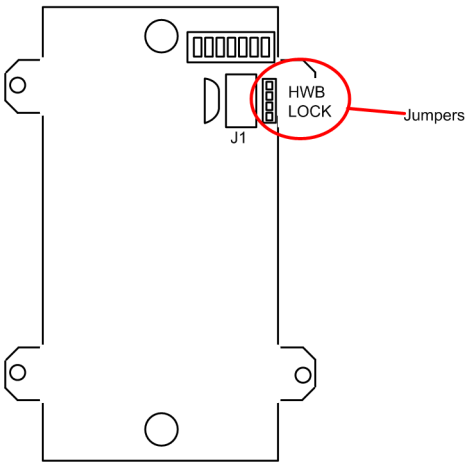


Figure 6-2 Jumper positions on VTAP100 PCB v4a or v5 hardware

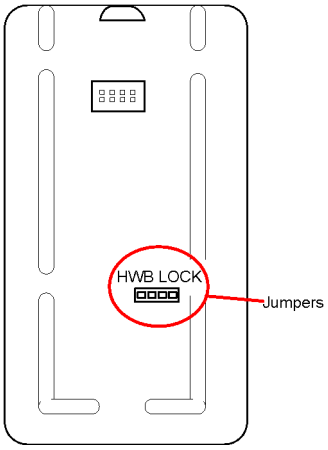


Figure 6-3 Jumper positions on VTAP50 PCB v1 or v2 hardware

Note: If you have a VTAP100 PCB version 3a or earlier the connections will be different, please **contact us** for manuals specific to your hardware. (If you are not sure which version PCB you have, just **Check status in BOOT.TXT**.)

If your VTAP reader board has a daughter board on top, as is the case for a VTAP50 and VTAP100, you will need to lift the daughter board off, to reach these jumpers.

When you start the VTAP reader, the presence of this jumper means the connected PC will not detect a USB mass storage device, only a keyboard (or keyboard and virtual COM port).

When you remove the jumper across LOCK and restart the VTAP reader, it will be detected as a USB mass storage device and you can make firmware or configuration changes again.

6.5 Reboot, remount, refresh commands

The VTAP reader supports a few direct commands, that are sent by creating and saving a text file called `command.txt`. They are `reboot`, `remount`, and `refresh`.

Example: `command.txt` file

```
!VTAPcommand

reboot
; Valid commands are reboot, remount or refresh
```

- Reboot will power cycle the VTAP reader, as if the USB cable was disconnected momentarily.
- Remount will remove the drive from the operating system briefly and then attach it again. This will force the operating system to re-read any changes that were made to the file system by the VTAP reader. Windows may not recognize the changes made to text files by the VTAP reader, but if you use the `remount` command, Windows will re-read all of the files.
- Refresh will force the VTAP reader to re-read `config.txt`. This is needed if you have renamed a file to be `config.txt`, as renaming is not otherwise noticed.

A Default Config.txt file

Copy and paste the text from the blue box to your `Config.txt` file to return to factory default configuration, for a VTAP100-OEM or VTAP100-USB-CC:

```
!VTAPconfig

VAS1MerchantID=pass.com.pronto.originpass.demo
VAS1KeySlot=6

ST1CollectorID=80644855
ST1KeySlot=6
ST1KeyVersion=1

NFCType2=1

KBLogMode=1
KBSource=A1
KBDelayMS=2
KBPassMode=0

LEDSelect=1
LEDDefaultRGB=1EBBCF
PassLED=00FF00,200,1,1
PassBeep=100,100,2
TagLED=00FF00,200
TagBeep=100
```

`Config.txt` variations for other VTAP models:

- VTAP100-USB-SQ is the almost the same, but with `LEDSelect=2`.
- VTAP50 models will have `LEDSelect=3` to control both external and on-board serial LEDs by default.

Note: Since firmware version v1.1.9.1 all serial communications interfaces are enabled by default, to ensure that serial communications are always possible even if the configuration file `config.txt` is missing or damaged. To take advantage of this, remove the line `ComPortEnable=0`.

Copy and paste the text from this box to your `Config.txt` file to return to factory default configuration, for a VTAP100-PAC-W-CC:

```
!VTAPconfig

VAS1MerchantID=pass.com.pronto.originpass.demo
VAS1KeySlot=6

ST1CollectorID=80644855
ST1KeySlot=6
ST1KeyVersion=1

NFCType2=U
MIFAREClassic=U

WiegandMode=1
WiegandSource=A1
WiegandPassMode=1
WiegandPassSeparator=|
WiegandPassSection=2

LEDDefaultRGB=1EBBCF
LEDSelect=1
PassBeep=50,50,2
TagBeep=100,100,1

KBLogMode=0
KBSource=A1
KBDelayMS=2
KBPassMode=0

StartupDelayMS=1000
```

The factory default configuration for a VTAP100-PAC-W-SQ is the almost the same, but with `LEDSelect=2`.

Note: Since firmware version v1.1.9.1 all serial communications interfaces are enabled by default, to ensure that serial communications are always possible even if the configuration file `config.txt` is missing or damaged. To take advantage of this, remove the line `ComPortEnable=0`.

Copy and paste the text from the blue box to your `config.txt` file to return your VTAP100-PRO-BW reader to Local mode factory default configuration for use with Dot Origin demonstration passses:

```
!VTAPconfig

VAS1MerchantID=pass.com.pronto.originpass.demo
VAS1KeySlot=6


ST1CollectorID=80644855
ST1KeySlot=6
ST1KeyVersion=1

CloudMode=0

NFCType2=U

KBLogMode=1
KBDelayMS=2
KBPassMode=0

LEDSelect=1
LEDDefaultRGB=1EBBCF

PassBeep=100,100,2
TagBeep=100

PassLED=00FF00,200,1,1
TagLED=00FF00,200

Serial2Settings=115200,n,8,1
BTKeyboardMode=1
```

Note: It is possible to 'break' the VTAP50 and VTAP100 if an incorrect `config.txt` or file is uploaded, for instance switching off Cloud mode when the reader is intended to work through VTAP Cloud. Contact **vtap-support@dotorigin.com** if you need Dot Origin to help recover your device.

B Default leds.ini file [VTAP50 v2 only]

Copy and paste the text from the blue box to a file called `leds.ini`. Put that file on to a VTAP50 with serial LED chain, then amend the file to suit your own LED signalling preferences. (You will then need to add the line such as `LEDDefaultRGB=FFFFFF:seq.comet@leds.ini` in `config.txt` to call the sequence defined in this file.)

```
; The patterns of the LEDs can be specified in sections starting "[leds.".
; Each line in the section specifies a RGB colour on the left and a list of
; LED numbers on the right. The LEDs are numbered from 1 up to the maximum
; specified in the main config (with a limit of 255).
;
; Multiple LEDs can be specified on each line separated by a comma e.g.
;
;    ff0000=1,2,3,4,5,6
;
; Ranges can also be specified with two numbers separated by a dash e.g.
;
;    ff0000=1-5,10-15,19-24
;
; Individual numbers and ranges can be mixed e.g.
;
;    00ff00=1-4,7,8,9,10-24
;
; If any LED is specified more than once, the last wins (in the order they
; appear in this file).
;
; Any LED not specified will be given a RGB value of 000000 (i.e. off).
;
[leds.off]

[leds.red]
050000=1-24

[leds.alt_red_green]
050000=1,3,5,7,9,11,13,15,17,19,21,23
000500=2,4,6,8,10,12,14,16,18,20,22,24

[leds.alt_green_red]
000500=1,3,5,7,9,11,13,15,17,19,21,23
050000=2,4,6,8,10,12,14,16,18,20,22,24

[leds.half_red_half_green]
ff0000=1-12
00ff00=13-24

[leds.red_blue_mix]
ff0000=1-5,9,10,11
0000ff=15,16,17,18-24

; This is a special pattern that is used when in serial LED mode but only an
; RGB value has been specified (in config or in an ?LED command).
;
; The RGB value will be used as the last part of the section name to lookup
```

```
; the LED pattern for that RGB colour.
;
; Note that the colours specified in the section for the LEDs do not have to
; match the original RGB colour.
;
[leds.rgb.ff0000]
800000=1
00ff00=2

; If the above specific RGB value is not present for a given RGB value then
; this generic section is used. Any entries with name "rgb" will receive the
; colour of the original RGB value.
;
; Note: the rgb field does not need to exist.
;
; Note: Other fields can specify different colours for other LEDs.
;
[leds.rgb]
rgb=1,3,5
00ff00=9

; Sequence sections always start with "seq." and supply a sequence of frames
; that specify the LED pattern and how long it is shown for. Each frame must
; have the key name "frame". Frames are shown in the order they are
; specified in this file. The duration is in milliseconds. The pattern
; references a section in the same file. Ideally the section should start
; with "[leds." to indicate it is a pattern specification, but this is not
; enforced.
;
; Note if the pattern section doesn't exist, the LEDs will be switched off
; for the duration of the frame.
;
; The number of frames is only limited by available RAM.
;
; The "repeat" key specifies how many times to repeat the sequence
; (currently limited to 255 repeats). If the "repeat" entry is not given,
; the sequence is shown once. A value of "forever" or "0" for the repeat
; entry will show the sequence continuously (until another sequence is
; started).
;
[seq.flash_red]
repeat=3
frame=500,leds.red
frame=500,leds.off

[seq.flash_red_forever]
repeat=forever
frame=500,leds.red
frame=500,leds.off

[leds.comet1]
100000=1
300000=2
400000=3
500000=4
```

```
[leds.comet2]
100000=2
200000=3
300000=4
400000=5

[leds.comet3]
100000=3
200000=4
300000=5
400000=6

[leds.comet4]
100000=4
200000=5
300000=6
400000=7

[leds.comet5]
100000=5
200000=6
300000=7
400000=8

[leds.comet6]
100000=6
200000=7
300000=8
400000=9

[leds.comet7]
100000=7
200000=8
300000=9
400000=10

[leds.comet8]
100000=8
200000=9
300000=10
400000=11

[leds.comet9]
100000=9
200000=10
300000=11
400000=12

[leds.comet10]
100000=10
200000=11
300000=12
400000=13

[leds.comet11]
100000=11
```

```
200000=12
300000=13
400000=14

[leds.comet12]
100000=12
200000=13
300000=14
400000=15

[leds.comet13]
100000=13
200000=14
300000=15
400000=16

[leds.comet14]
100000=14
200000=15
300000=16
400000=17

[leds.comet15]
100000=15
200000=16
300000=17
400000=18

[leds.comet16]
100000=16
200000=17
300000=18
400000=19

[leds.comet17]
100000=17
200000=18
300000=19
400000=20

[leds.comet18]
100000=18
200000=19
300000=20
400000=21

[leds.comet19]
100000=19
200000=20
300000=21
400000=22

[leds.comet20]
100000=20
200000=21
300000=22
```

```
400000=23

[leds.comet21]
200000=21
150000=22
100000=23
050000=24

[leds.comet22]
100000=22
200000=23
300000=24
400000=1

[leds.comet23]
100000=23
200000=24
300000=1
400000=2

[leds.comet24]
100000=24
200000=1
300000=2
400000=3

[seq.comet]
repeat=forever
frame=100,leds.comet1
frame=100,leds.comet2
frame=100,leds.comet3
frame=100,leds.comet4
frame=100,leds.comet5
frame=100,leds.comet6
frame=100,leds.comet7
frame=100,leds.comet8
frame=100,leds.comet9
frame=100,leds.comet10
frame=100,leds.comet11
frame=100,leds.comet12
frame=100,leds.comet13
frame=100,leds.comet14
frame=100,leds.comet15
frame=100,leds.comet16
frame=100,leds.comet17
frame=100,leds.comet18
frame=100,leds.comet19
frame=100,leds.comet20
frame=100,leds.comet21
frame=100,leds.comet22
frame=100,leds.comet23
frame=100,leds.comet24
```