VTAP

Settings and Commands Reference

Firmware from v2.2.2.1

VTAP50 and VTAP100

Revised November 2023 v3.52

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

Settings and commands which have been added since the last general release software v2.2.0.2 are highlighted as NEW. Settings and commands with additional options or changed defaults in this version are highlighted as NEW OPTIONS

You must be running the latest features firmware v2.2.2.1 or later to access all of these highlighted commands or settings. Visit **https://vtapnfc.com** if you need to download the latest firmware version.

To update the firmware on a VTAP reader:

1. Copy the 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: Configuration is exactly the same for VTAP50 and VTAP100, but the firmware files are specific to the type and version of VTAP reader and not interchangeable.

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

   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 the Firmware field in `BOOT.TXT` to see if the firmware version is as you expect.

If you are new to using the VTAP reader, you will find much more help in the VTAP Configuration Guide.

2 Settings

Settings are all of the parameters that can be set in a text file on your VTAP reader to control its operation. The settings are grouped according to which of the text files they can be used in. Most settings are used in the `config.txt` file.

2.1 config.txt

This section lists all valid settings for `config.txt`. `Config.txt` is a file which must contain the string `!VTAPconfig` at the start.

2.1.1 Apple VAS

Note: In these settings # is a number from 1 to 6, showing which settings form a group for reading each of 1 to 6 VAS pass types that might be presented to this VTAP reader. If you have multiple IDs, the lowest numbered ID will be requested first, then continuing in ascending numeric order.

These settings are necessary to provide the VTAP reader with pairs of `VAS#MerchantID` and `VAS#KeySlot` that identify you as an Apple merchant entitled to read particular passes, and point to the appropriate private key to decode the data. You will separately need to upload the key files to those key slots. Help with this is included in the VTAP Configuration Guide.

Setting	Definition	Default value	Example value	Options
`VAS#KeySlot`	An instruction to use the private key saved to a particular slot on the VTAP, when reading passes of type #.	`=0`	`=2`	1 to 6, identifying key file. If 0 or omitted (default), all available keys will be compared with the 4 byte hash of the public key for the data, to choose the right key. If the data received by VTAP cannot be decrypted, the phone will register a pass read, but the data will not be output.
`VAS#MerchantID`	An identifier supplied by a pass provider or customer to uniquely identify your passes in Apple Wallet applications	`N/A`	`=pass.com.pronto.fictionplc.demo`	

Setting	Definition	Default value	Example value	Options
VAS#MerchantURL	A website URL to be visited when a pass of type # is presented. Not currently supported by iOS for VAS-only transactions.	N/A	=https://dotorigin.com/	
VASDefaultPassesEnabled	Used to restrict the number of VAS pass types to be checked, reducing the time spent testing each pass presented.	=1,2,3,4,5,6	=2,3	In this example, only VAS2 and VAS3 pass types will be enabled at startup

2.1.2 Google Smart Tap

A list of all valid configuration settings relating to Google Smart Tap controlling parameters in the `config.txt` file.

> **Note:** In these settings # is a number from 1 to 6, showing which settings form a group for reading each of 1 to 6 Smart Tap pass types that might be presented to this VTAP reader.

These settings are necessary to provide the VTAP reader with pairs of `ST#CollectorID` and `ST#KeySlot` that identify you as the Google merchant entitled to read particular passes, and point to the appropriate private key to decode the data. You will separately need to upload the key files to those key slots. Multiple collector IDs are not supported by Android, which means you cannot request more than one Collector ID from Google. Only one set should be live at any one time. Help with this is included in the VTAP Configuration Guide.

Setting	Definition	Default value	Example value	Options
`ST#CollectorID`	An identifier supplied by Google to uniquely identify your passes in Google Wallet Smart Tap applications	`N/A`	`=97598013`	
`ST#KeySlot`	An instruction to use the private key saved to a particular slot on the VTAP, when reading passes of type #.	`=0`	`=4`	1 to 6, identifying key file. If omitted or set =0 (default) Google Wallet Smart Tap data will still be received and sent by the VTAP, only if the pass does not require authentication by the terminal.
`ST#KeyVersion`	An instruction to use the private key with a particular version number saved on the VTAP, when reading passes of type #	`=0`	`=10`	
`STDefaultPassesEnabled`	Used to restrict the number of ST pass types to be checked, reducing the time spent testing each pass presented.	`=1,2,3,4,5,6`	`=5`	In this example, only ST5 pass type will be enabled at startup

2.1.3 NFC card or tag

A list of all valid configuration settings which relate to extracting pass information from NFC cards or tags for the `config.txt` file.

The setting `NFCType2`, , `NFCType4` or `NFCType5` is used to direct the VTAP reader to recognise and read particular types of data from the different NFC cards that may be presented. The settings prefixed `NDEFTag...` restrict the types of data that will be read.

The `NDEFTagType4AID` and `NDEFTagType4AIDOnly` settings allow you to decide which NFC ISO application IDs (AIDs) should be tried when attempting to read NFCForum NDEF data from a Type4 card/tag (with `NFCType4=N`). The default behaviour is to try to select the NDEF-Tag application AID (D2760000850101h) as defined by the NFC Forum, but if `NDEFTagType4AID` is set then this AID will be tried first, in preference to the default. If the `NDEFTagType4AID` is not found, the VTAP will then try the default NDEF-Tag application AID. This behaviour can be overridden by setting `NDEFTagType4AIDOnly=1`, so the VTAP will not attempt to use the default NDEF-Tag application AID after failing to select the specified `NDEFTagType4AID`.

If you want to read blocks of data from NFC cards or tags you will need to use settings prefixed `TagRead...` described in section **2.1.4** .

Setting	Definition	Default value	Example value	Options
`NFCType#`	Permits the cards or tags of an NFC Type # to be read, and defines which part of the card/tag data will be read.	`=0`	`=U`	For NFC Types: Use =U or 1 to permit UID to be read (4 or 7 bytes usually in the first block). Use =N or 2 to read NDEF records (NFCType 2 or 4). Use =B or 3 to read or decode up to 16 bytes of a memory block (similar to MIFAREClassic for MIFARE cards). Use this with **MIFARE card or tag** settings. Use =D (NFC Type 4 only) to read DESFire secure data with **DESFire card or tag** settings. =0 is disabled (default). An order of precedence allows multiple selections from Pass, DESFire secure, (Block data from MIFARE cards), NDEF, UID. So =NU or =UN would read UID if an NDEF record cannot be read.

Setting	Definition	Default value	Example value	Options
IgnoreRandomUID	Filter out random tag reads. Some devices presented to the VTAP can appear as a Type 4 tag with random UID. May be needed if a VTAP is configured to read both passes and UID from NFC Type 4 tags.	=0	=1	=1 to filter out random NFC Type 4 tag reads. Only useful if you have set NFCType4=U.
NDEFTagExtractType	Read only NDEF records of one type, such as type T (text). The output will be the payload of the first matching NDEF record found (by searching the NDEF records recursively). Where nothing is found that matches the constraints set, nothing will be output. If no constraints are set, all NFC data content will be output.	N/A	=T	=T for text records (or omit setting). Only useful if you have set NFCType2=N, NFCType3=N or NFCType4=N.

Setting	Definition	Default value	Example value	Options
NDEFTagExtractID	Read only NDEF records with a particular ID, such as 'name'. The output will be the payload of the first matching NDEF record found (by searching the NDEF records recursively). Where nothing is found that matches the constraints set, nothing will be output. If no constraints are set, all NFC data content will be output.	N/A	=name	=name (or omit setting). Only useful if you have set NFCType2=N, NFCType3=N or NFCType4=N.
NEW NDEFTagType4AID	Allows you to set a preferred AID to try when reading an NDEF Type 4 record.	N/A	=f1234567890abcd	The AID set here will be tried before trying the two standard NFC Forum application IDs (d2760000850101 followed by d2760000850100). The standard AIDs are always tried when this setting is not used.
NEW NDEFTagType4AIDOnly	Allows you to restrict the VTAP reader to only use the AID provided as the NDEFTagType4AID setting.	=0	=1	=1 will prevent the VTAP reader from trying the standard NFC Forum application IDs (d2760000850101 followed by d2760000850100), so that only the value set for NDEFTagType4AID will be tried. =0 allows those standard AIDs to be tried if NDEFTagType4AID is not successful.

2.1.4 MIFARE card or tag

A list of all valid configuration settings for the `config.txt` file, which relate to extracting pass information from MIFARE cards or tags.

The setting `MIFAREClassic` is used to direct the VTAP reader to read various types of data from any MIFARE Classic cards that are presented. If you want to read blocks of data from MIFARE Classic or NFC cards or tags you will also need to use settings prefixed `TagRead...`

If `MIFAREClassic=B` to read block data, you need to specify `TagReadBlockNum`, `TagReadKey` and `TagReadKeyType` to define which block to read and provide the key information needed to decode that block.

`TagReadOffset` and `TagReadLength` are optional, but often used together, to define where to start extracting data from a data block and how much data to read.

If you choose `TagReadFormat=d` to require a decimal output of block data you can use `TagReadRightShift` to drop parity bits or `TagReadMinDigits` to add leading zeroes, when a fixed width output is required.

There is a simple example in an Application Note.

`TagByteOrder` can be used on UIDs to reverse the byte order, if that suits your application.

Setting	Definition	Default value	Example value	Options
`MIFAREClassic`	Permits MIFARE cards or tags to be read, and defines which part of the card/tag data will be read. This can be the UID (4 or 7 bytes usually in the first block) or up to 16 bytes of data from another specified block. (Similar to NFCType for NFC cards).	`=0`	`=U`	Use =U or 1 to permit MIFARE UID to be read, =N or 2 to read NFC data (not currently supported on MIFARE Classic), =B or 3 to read or decode a memory block. =0 is disabled (default).An order of precedence allows multiple selections, from MIFARE just from Pass, Block data or UID. So =BU or =UB would read UID if an block data cannot be read.

Setting	Definition	Default value	Example value	Options
TagByteOrder	Option to reverse the byte order of UID or block data read from a card/tag. A UID of 1A2B3C4D becomes 4D3C2B1A.	=0	=1	Example =1 reverses the byte order. It may be used in conjunction with TagReadFormat to produce hex-standard, hex-reversed decimal and decimal-reversed output of UIDs.
TagReadBlockNum	Select a block number to read.	=0	=56	MIFARE Classic 1K cards have 64 blocks (16 sectors of 4 blocks each) and 4K cards have 256 blocks (32 sectors of 4 blocks and 8 sectors of 16 blocks). These are numbered in decimal from 0 to 255. (Must also set NFCType#=B or 3, or MIFAREClassic=B or 3).
TagReadKey	The key required to read block data from a MIFARE Classic card or tag.	N/A	=123ABC456DEF	6 bytes in hexadecimal
TagReadKeyType	This is the key type, for the key set in TagReadKey.	=A	=B	=A or =B to describe the MIFARE Classic key type to be used with block data
TagReadMinDigits	Require fixed-width UID output by adding leading zeros as necessary.	=0	=10	1 to 20 digits, corresponding to the maximum number of decimal digits in a 64 bit value. Omitted or =0 has no effect. =A for automatic padding with leading zeros to length for UID or tag: 10 digits for 32bit UID, 17 digits for 56bit UID or 20 digits for 64bit UID. =X for extended automatic padding: 10 digits for 32bit UID, 18 digits for 56bit UID or 20 digits for 64bit UID.
TagReadOffset	The byte offset within block or UID data. Describes the position to start reading data from, within a block.	=0	=5	=0 to =15. If this offset is used together with TagReadLength, TagReadOffset + TagReadLength must be less than or equal to 16.

Setting	Definition	Default value	Example value	Options
TagReadLength	The number of bytes of block or UID data to read from a tag or card. (Except for DESFire tags or cards which have an overriding DESFireReadLength setting.)	=16	=4	=1 byte to =16 bytes. This number should not exceed 4 if TagReadFormat is 'd'ecimal, as no more than 4 bytes are used in this case. If this offset is used together with TagReadOffset, TagReadOffset + TagReadLength must be less than or equal to 16. (If it is not, the length will be reduced by truncating data.)
TagReadFormat	Choose to format the extracted block data as ASCII, hex or decimal.	=h	=d	=a for ASCII characters (each byte is an ASCII character); =d for decimal (interpret binary data as a 64 bit decimal value and output as ASCII decimal digits), in this case TagReadLength should not exceed 4 bytes; =h for hexadecimal (convert binary data to ASCII hex digits with 2 digits per byte).
TagReadRightShift	Number of bits to right shift data. It may be used to remove any parity bits included at the end of extracted binary data.	=0	=1	Number of bits to right shift decimal 64 bit block data. (Only relevant if TagReadFormat=d.)

2.1.5 DESFire card or tag

A list of all valid configuration settings which relate to extracting pass information from DESFire cards or tags for the `config.txt` file.

DESFire cards may contain a number of applications, identified by an application ID. Each application may contain a number of data files, each identified by a file number, which may be individually protected. The VTAP reader supports a number of formats to read, decode or output the secure data. The format might be HID 10301 26-bit or HID 10301 37-bit. Reading data, from a DESFire card which contains secured data, therefore includes uploading the app key file, and providing information about the application ID and the key number to be used for authentication, along with the file number and the crypto algorithm for decoding each file and bit format.

To read DESFire cards will require setting `NFCType4=D`, uploading a suitable `appkey#.txt` file with the relevant application key, and using all of these settings prefixed `DESFire...`. Only DESFire cards which are unformatted or Key-ID 26-bit HID 10301 data are currently supported.

Note: In these settings # is a number from 1 to 6, showing which settings form a group for reading each of 1 to 6 values from separate files and or applications on a DESFire card or tag. If you use multiple DESFire#... settings the values read will be output together, spaced by the `DESFireSeparator` string. The lowest numbered DESFire read will be first in the output string, then continuing in ascending numeric order. (For Wiegand data multiple reads are not supported, so only the lowest numbered DESFire#... settings will be used.) If no number is used the setting will be treated as set 1.

For examples refer to Application Notes.

Setting	Definition	Default value	Example value	Options
NEW OPTIONS `DESFire#AppID`	Hex number identifying your DESFire application	`N/A`	`=F56400`	
NEW OPTIONS `DESFire#Crypto`	Identifies the cryptographic method used for DESFire cards or tags	`=3`	`=1`	=3 identifies AES (default), =1 identifies 3DES cryptography, =0 for no cryptography

Setting	Definition	Default value	Example value	Options
NEW OPTIONS DESFire#FileID	Number identifying the file within your DESFire application to read	N/A	=1	Use a value from 1 to 255
NEW OPTIONS DESFire#KeyNum	Number identifying the application key needed to read your DESFire file	N/A	=1	
NEW OPTIONS DESFire#KeySlot	Identifies which uploaded appkey#.txt file contains the key for accessing the DESFire file	N/A	=1	1 to 4, to refer to the application key files uploaded as appkey1.txt through appkey4.txt
NEW OPTIONS DESFire#Format	Identifies which bit format is used to store the data.	=0	=1	=0 means no format (specify DESFire#ReadLength and TagReadFormat to determine how the data is output), =1 means KEY-ID format (26 bit facility code and number format, H10301 compatible),
NEW OPTIONS DESFire#ReadLength	The number of bytes of data to read from DESFire cards, distinct from TagReadLength which applies to other cards and tags.	=3	=4	=1 byte to =16 bytes. (Default =3 suits Key-ID encoded cards, however the setting not required if DESFire#Format has selected a Key-ID format, as length is then automatic.
NEW DESFireSeparator	Defines a string to include in between the data obtained from separate DESFire card reads, when there is more than one.	=","|	="|"	Choose any separator that suits your application, up to 16 characters. (Note: Pre/postfix strings can still also be applied to the combined DESFire read output over a given interface.)

2.1.6 Keyboard/barcode emulation

A list of all valid configuration settings which relate to sending pass information over a keyboard/barcode emulation interface for the `config.txt` file.

If `KBLogMode=1` and `KBSource` is set, you can open an application on the computer connected to the VTAP reader, and pass or card data that is read will be typed to the screen. `KBSource` determines whether keyboard wedge/barcode mode is triggered only by mobile pass reads, or cards/tag reads too. You might need a `KBDelayMS` value to insert a delay between read and transmit, to suit your receiving application.

`KBPrefix` and `KBPostfix` can be optionally used to add extra characters before or after a pass read, to suit your application.

When `KBPassMode=1` you can use all of the other settings beginning `KBPass...` to extract only part of the mobile pass data. To do this you have to start by identifying a separator character in the data as `KBPassSeparator`. This allows the data to be regarded as a set of sections that can be identified by number. By identifying a `KBContentSeparator` this process can be repeated at a lower level, within a section. There is a simple example in the VTAP Configuration Guide.

Setting	Definition	Default value	Example value	Options
`KBDelayMS`	Inserts a number of milliseconds delay between key down and key up in the keyboard output.	`=5`	`=10`	Adjust to suit your receiving application between 5ms and 255ms.
`KBLogMode`	Turns keyboard emulation on or off.	`=0`	`=1`	=1 starts keyboard emulation, =0 switches it off [default]
`KBPassContentMode`	Turns on a subdivision of a data section into smaller content sections.	`=0`	`=1`	=0 off (default), =1 on. To use this mode you must also supply KBPassContentSeparator and KBPassContentSection.

Setting	Definition	Default value	Example value	Options
KBPassContentSection	Identifies the content section of interest, by finding KBPassContentSeparator characters. First content section before a separator is 0.	=0	=1	Used only if KBPassContentMode=1 and a KBPassContentSeparator is set.
KBPassContentSeparator	Identifies the content separator character in the data which bounds content sections.	=%	=%	Used only if KBPassContentMode=1 and a KBPassContentSection is set.
KBPassLength	Number of characters of data to extract, starting from the position defined by KBPassStart.	=0	=10	Defaults to 0 meaning do not truncate data, send all available characters from the pass data, or extracted section of pass data.
KBPassMode	Sets whether to extract parts of the data from the mobile NFC pass.	=0	=1	=0 send data unchanged [default] so any other KBPass... settings are ignored (except KBPassStart or KBPassLength used to truncate the data). =1 means extract data based on other KBPass... settings
KBPassSection	Identifies the section of interest, based on finding KBPassSeparator characters. First content section before a separator is 0.	=0	=2	Used only if KBPassMode=1 and a KBPassSeparator is set.
KBPassSeparator	Identifies the separator character in the data which bounds sections.	=|	=|	Used only if KBPassMode=1 and a KBPassSection is set.

Setting	Definition	Default value	Example value	Options
KBPassStart	Number of characters into pass data to start reading, where first character is 0.	=0	=5	Used with KBPassLength.
NEW OPTIONS				
KBPostfix	Adds special characters after data, if needed.	=%0A	=end%0D	Use standard ASCII hex characters to describe the keystroke to append. In the example %0D adds a carriage return after the data, %0A adds a new line after every pass or tag entry. $t would include the pass type (a, g, 0, 2, 4, or 5, indicating Apple, Google, MIFARE (0) , NFC type 2, 4 or 5). (Use $$t if '$t' is required in the output.) tn will follow the pass type identifier with a digit for VAS merchant ID/ST collector ID index (1 to 6) and a digit for keyslot used (1 to 6).
NEW OPTIONS				
KBPrefix	Adds special characters before the data, if needed.	[Empty]	=%0Astart	Use standard ASCII hex characters to describe the keystroke to append. In the example %0A adds a new line linefeed before the data. $t would include the pass type (a, g, 0, 2, 4, or 5, indicating Apple, Google, MIFARE (0) , NFC type 2, 4 or 5). (Use $$t if '$t' is required in the output.) tn will follow the pass type identifier with a digit for VAS merchant ID/ST collector ID index (1 to 6) and a digit for keyslot used (1 to 6).

Setting	Definition	Default value	Example value	Options
NEW OPTIONS				
KBSource	Controls which types of event data (pass reads, card/tag reads, serial commands) are sent as a keyboard entries.	=A1	=80	The values are determined from a bitwise combination of the following hexadecimal values. Source options include these bit values for mobile passes and tag UIDs: Bit 7 (0x80) = Mobile Pass (Apple VAS/Google Wallet Smart Tap) Bit 6 (0x40) = STUID Bit 5 (0x20) = Write to card emulation mode (CardEmulationMode=1 see **VTAP NFC tag emulation settings**) Bit 4 (0x10) = RFU Bit 3 (0x80) = RFU Bit 2 (0x40) = RFU Bit 1 (0x20) = Command interface messages (>interface:type:message see **Dynamic configuration commands**) Bit 0 (0x10) = Card/Tag UID So, for example setting KBSource=80 (hex) will send only mobile NFC pass data to the keyboard, use =A1 to send data from NFC passes and NFC cards/tags and card emulation or =83 to send passes, cards/tags and serial commands

2.1.7 Virtual COM port

A list of all valid configuration settings which relate to sending pass information over a virtual COM port interface for the `config.txt` file.

If `ComPortEnable=1`, `ComPortMode=1` and `ComPortSource` is set, pass or card data read by the VTAP reader will be sent over the virtual COM port. This is the virtual COM port in active mode. `ComPortSource` determines whether data is sent over the COM port interface only when there are mobile pass reads, or cards/tag reads too.

After setting `ComPortEnable=1` in `config.txt` the unit needs to be rebooted or power cycled. On a Windows PC, if you check the Device Manager, you will see under 'Ports (COM & LPT)' that there is now a VTAP reader that has been assigned a COM port. (If it says 'USB Serial Device' rather than 'VTAP100', you should right click on the `VTAP100.inf` or `VTAP.inf` file on the VTAP file system to install the correct driver.)

The COM port is described as being in either active or passive mode. Active mode means that pass or card data read by the VTAP reader will be sent over the COM port immediately, whenever it is read. Passive mode means that the VTAP reader will only send on that data in response to a command. These are listed in section **3**.

By default the VTAP reader is in active mode. Use `PassiveInterfaces=1` to enable passive mode and `CommandInterfaces=1` to allow commands to be received on the command interface. Set `InvalidDataCacheMS` for the time you want pass read data to be retained by the VTAP reader, while it is waiting for a Com Port request to send it onwards. There is a simple example in the VTAP Serial Interface Guide.

`ComPortPrefix` and `ComPortPostfix` can be optionally used to add extra characters before or after a pass read, to suit your application.

When `ComPortPassMode=1` you can use all of the other settings beginning `ComPortPass...` to extract only part of the mobile pass data. To do this you have to start by identifying a separator character in the data as `ComPortPassSeparator`. This allows the data to be regarded as a set of sections that can be identified by number. By identifying a `ComPortPassContentSeparator` this process can be repeated at a lower level, within a section. There is a simple example in the VTAP Configuration Guide.

Setting	Definition	Default value	Example value	Options
CommandInterfaces	Enables or disables your ability to send commands over particular interfaces.	=7	=1	=1 enables the ComPort, =2 enables the Serial port, =4 enables the Serial2 port. (Add values together to enable multiple ports, so =3 enables both ComPort and Serial port.) Default enables all ports.
ComPortEnable	Enable a virtual COM port for the USB interface so the VTAP is treated by the connected PC as a COM port, as well as a mass storage device.	=1	=0	=1 is enabled [default] and =0 is disabled
ComPortMode	Activate the COM port as an interface for receiving mobile pass data.	=1	=0	Set this to 1 [default] if you want VTAP data sent over the COM port ; = 0 is disabled
ComPortPassContentMode	Turns on a subdivision of a data section into smaller content sections.	=0	=1	=0 off (default), =1 on. To use this mode you must also supply ComPortPassContentSeparator and ComPortPassContentSection.
ComPortPassContentSection	Identifies the content section of interest, based on finding ComPortPassContentSeparator characters. First content section before a separator is 0.	=0	=1	Only used if ComPortPassContentMode=1 and ComPortPassContentSeparator is set.
ComPortPassContentSeparator	Identifies the content separator character in the data which bounds content sections.	=%	=%	Only used if ComPortPassContentMode=1 and ComPortPassContentSection is set.

Setting	Definition	Default value	Example value	Options
ComPortPassLength	Number of characters of data to extract, starting from the position defined by ComPortPassStart.	=0	=10	Defaults to 0 meaning do not truncate data, send all available characters from the pass data, or extracted section of pass data.
ComPortPassMode	Sets whether to extract parts of the data from the mobile NFC pass.	=0	=1	=0 send data unchanged [default] so any other ComPortPass... settings are ignored (except ComPortPassStart or ComPortPassLength used to truncate the data), =1 means extract data based on other ComPortPass... settings .
ComPortPassSection	Identifies the section of interest, based on finding ComPortPassSeparator characters. First content section before a separator is 0.	=0	=2	Used only if ComPortPassMode=1 and a ComPortPassSeparator is set.
ComPortPassSeparator	Identifies the separator character in the data which bounds sections.	=|	=|	Only used if ComPortPassMode=1 and a ComPortPassSection is set.
ComPortPassStart	Number of characters into pass data to reading, where first character is 0.	=0	=5	Used in conjunction with ComPortPassLength.
NEW OPTIONS				
ComPortPostfix	Adds special characters after data, if needed.	=%0D%0A	=end%0D	Use standard ASCII hex characters to describe the keystroke to append. In the example %0D adds a carriage return after the data, %0A adds a new line after every pass or tag entry. $t would include the pass type (a, g, 0, 2, 4, or 5, indicating Apple, Google, MIFARE (0) , NFC type 2, 4 or 5). (Use $$t if '$t' is required in the output.) tn will follow the pass type identifier with a digit for VAS merchant ID/ST collector ID index (1 to 6) and a digit for keyslot used (1 to 6).

Setting	Definition	Default value	Example value	Options
NEW OPTIONS `ComPortPrefix`	Adds special characters before the data, if needed.	`[Empty]`	`=%0Astart`	Use standard ASCII hex characters to describe the keystroke to append. In the example %0A adds a new line linefeed before the data. $t would include the pass type (a, g, 0, 2, 4, or 5, indicating Apple, Google, MIFARE (0) , NFC type 2, 4 or 5). (Use $$t if '$t' is required in the output.) tn will follow the pass type identifier with a digit for VAS merchant ID/ST collector ID index (1 to 6) and a digit for keyslot used (1 to 6).
NEW OPTIONS `ComPortSource`	Controls which types of event data (pass reads, card/tag reads, serial commands) are sent to the COM port interface.	`=A1`	`=80`	The values are determined from a bitwise combination of the following hexadecimal values. Source options include these bit values for mobile passes and tag UIDs: Bit 7 (0x80) = Mobile Pass (Apple VAS/Google Wallet Smart Tap) Bit 6 (0x40) = STUID Bit 5 (0x20) = Write to card emulation mode (`CardEmulationMode=1` see **VTAP NFC tag emulation settings**) Bit 4 (0x10) = RFU Bit 3 (0x80) = RFU Bit 2 (0x40) = RFU Bit 1 (0x20) = Command interface messages (`>interface:type:message` see **Dynamic configuration commands**) Bit 0 (0x10) = Card/Tag UID So, for example setting ComPortSource=80 (hex) sends mobile NFC pass data only, use =A1 to send data from NFC passes, NFC cards/tags and card emulation or =83 to send passes, cards/tags and serial commands

Setting	Definition	Default value	Example value	Options
`InvalidDataCacheMS`	A period to retain data after reading in milliseconds. Set to retain data in memory until requested, but without undue security risk to that data.	`=3000`	`=3000`	Example says retain data for 3000ms after a mobile pass, card/tag read. Used with interfaces in passive mode only.
`PassiveInterfaces`	Enable passive mode at startup for any of the command interfaces.	`=0`	`=3`	Example =3 enables passive mode on Serial and COMport. The value is any combination of bit values (as per CommandInterfaces values) where COMPort=1, Serial=2, Serial2=4; Default =0 means passive mode is disabled on all interfaces.

2.1.8 Wiegand interface

A list of all valid configuration settings relating to the Wiegand interface for use in the `config.txt` file.

Note: The Wiegand interface is only available on VTAP100-PAC-W hardware.

If `WiegandMode=1` and `WiegandSource` is set, you can open an application on the computer connected to the VTAP reader, and pass or card data that is read will be sent over the Wiegand interface. `WiegandSource` determines whether data is sent over the Wiegand interface only when there are mobile pass reads, or cards/tag reads too.

`PassFormat` allows you to choose to interpret ASCII pass data as hex or decimal. By default 56 bits of data will be passed over the Wiegand interface, but you can change this by setting `PassWiegandBits`. The setting `PassWiegandParity` allows you to pad mobile pass data with zeroes, to imitate the inclusion of parity bits. This makes the pass data interchangeable that from with cards/tags which include parity bits, as long as parity is not being tested for validity.

When `WiegandPassMode=1` you can use all of the other settings beginning `WiegandPass...` to extract only part of the mobile pass data. To do this you have to start by identifying a separator character in the data as `WiegandPassSeparator`. This allows the data to be regarded as a set of sections that can be identified by number. By identifying a `WiegandPassContentSeparator` this process can be repeated at a lower level, within a section. There is a simple example in the VTAP Configuration Guide.

Setting	Definition	Default value	Example value	Options
`WiegandMode`	Activate the Wiegand interface for receiving mobile pass data.	`=0`	`=1`	=1 sends VTAP data to the Wiegand interface, =0 switches it off [default]
`WiegandInputEnable`	Control the VTAP response to Wiegand red, green and beep input signals, from a door controller.	`=81`	`=0`	=81 (default) means that the VTAP will receive both beep and red and green LED inputs, =0 prevents any use of these inputs. To drive VTAP responses (beep and LEDs) you must also have WiegandMode=1.

Setting	Definition	Default value	Example value	Options
WiegandPassContentMode	Turns on a subdivision of a data section into content sections.	=0	=1	=0 off (default), =1 on. To use this mode you must also supply WiegandPassContentSeparator and WiegandPassContentSection.
WiegandPassContentSection	Identifies the content section of interest, based on finding WiegandPassContentSeparator characters. First content section before a separator is 0.	=0	=1	Only used if WiegandPassContentMode=1 and WiegandPassContentSection is set
WiegandPassContentSeparator	Identifies the content separator character in the data which bounds content sections.	=%	=%	Only used if WiegandPassContentMode=1 and WiegandPassContentSection is set.
WiegandPassLength	Number of characters of data to extract, starting from the position defined by WiegandPassStart.	=0	=10	Defaults to 0 meaning do not truncate data, send all available characters from the pass data, or extracted section of pass data.
WiegandPassMode	Sets whether to extract parts of the data from the mobile NFC pass.	=0	=1	=0 send data unchanged [default] so any other WiegandPass... settings are ignored (except WiegandPassStart or WiegandPassLength used to truncate the data), =1 extract data based on other WiegandPass... settings .
WiegandPassSection	Identifies the section of interest, based on finding WiegandPassSeparator characters. First content section before a separator is 0.	=0	=2	Only used if WiegandPassMode=1 and a WiegandPassSeparator is set.

Setting	Definition	Default value	Example value	Options
WiegandPassSeparator	Identifies the separator character in the data which bounds sections.	=|	=|	Only used if WiegandPassMode=1 and WiegandPassSection is set.
WiegandPassStart	Number of characters into string to start extracting data, where first character is 0.	=0	=5	Used in conjunction with WiegandPassLength.
WiegandPassTypeIdent	Inserts an additional leading byte of pass type identifier information in the Wiegand output. Either 01 for Apple VAS, or 02 for Google ST. This allows the controller or application receiving the Wiegand data (via a door controller) to distinguish between cards/tags and mobile wallet passes, as well as identifying the wallet type (Google or Apple) for reporting purposes.	=0	=1	=0 disabled or =1 add the pass type identifier. When this option is used, the Wiegand bit length is fixed to 56 bits + 8 bits type identifier = 64 bits total, so PassWiegandBits is ignored, if this option is enabled.
PassFormat	Choose to interpret ASCII pass data characters as either hex or decimal, when converting the pass data to a Wiegand bit sequence.	=h	=d	=d for a decimal 64 bit number, from which the appropriate number of bits are output to Wiegand ; =h for hexadecimal, converted to a byte sequence from which the appropriate number of bits are output. (These lengths are both controlled by WiegandPassBits).
PassWiegandBits	Specify the number of bits to output over the Wiegand interface from the start of the filtered pass data, where it otherwise defaults to 56.	=56	=64	= the number of bits required from 1 to 255

Setting	Definition	Default value	Example value	Options
PassWiegandParity	Adds the equivalent of parity bits to decimal pass data. This makes it possible to use mobile pass number formats that include parity bits, as long as the parity bit(s) are not being tested for validity. This is the same as bitwise left-shifting the data by the given number of bits within the 32 bit result.	=0	=1	Adds a the given number of bits of 0 padding (for example in lieu of parity bits) to the least significant end of the data. Use together with PassFormat=d for decimal data.
TagWiegandASCIIFormat	Defines how to interpret ASCII tag or card data for output over a Wiegand interface.	=a	=d	=d is decimal and =h is hexadecimal, =a means ASCII (default)
TagWiegandBits	Specify the number of bits to output over the Wiegand interface from the start of the extracted tag data, where it otherwise defaults to 56. Or allow the number of bits to be automatically determined from the data.	=56	=64	= 0 to automatically determine the bit length from the extracted data, or set to the number of bits required from 1 to 255

Setting	Definition	Default value	Example value	Options
TagWiegandParity	Adds the equivalent of parity bits to decimal tag data. This makes it possible to use tag or card number formats that include parity bits, as long as the parity bit(s) are not being tested for validity. This is the same as bitwise left-shifting the data by the given number of bits.	=0	=1	Adds a the given number of bits of 0 padding (for example in lieu of parity bits) to the least significant end of the data. Used with TagReadFormat=a and TagWiegandASCIIFormat=d to get decimal data.
NEW OPTIONS				
WiegandSource	WiegandSource controls which types of event data (pass reads, card/tag reads, serial commands) go to the Wiegand interface.	=A1	=80	The values are determined from a bitwise combination of the following hexadecimal values. Source options include these bit values for mobile passes and tag UIDs: Bit 7 = Mobile Pass (Apple VAS/Google Wallet Smart Tap) Bit 6 = STUID Bit 5 = Card emulation data Bit 4 = RFU Bit 3 = RFU Bit 2 = RFU Bit 1 = Serial Bit 0 = Card/Tag UID So, for example setting =80 sends mobile NFC pass data only, use =81 to send data from NFC passes and NFC cards/tags or =83 to send passes, cards/tags and serial commands

2.1.9 Serial RS232 interface

A list of all valid configuration settings for the `config.txt` file, which relate to sending pass information over the Serial, RS232 interface on the VTAP50 or VTAP100 J1 connector.

Note: The Serial RS232 port is available on VTAP50, or on VTAP100 v3a hardware onwards.

If `SerialMode=1` and `SerialSource` is set, you can open an application on the computer connected to the VTAP reader, and pass or card data that is read will be sent over the Serial RS232 interface. `SerialSource` determines whether the Serial RS232 interface is used to send mobile pass reads only, or cards/tag reads too.

You might need `SerialSettings` to change the standard parameters for data transfer from the 9600, n, 8, 1 default. `SerialStartup` can be used to set a message to transfer when the Serial interface is first enabled.

`SerialPrefix` and `SerialPostfix` can be optionally used to add extra characters before or after a pass read, to suit your application.

When `SerialPassMode=1` you can use all of the other settings beginning `SerialPass...` to extract only part of the mobile pass data. To do this you have to start by identifying a separator character in the data as `SerialPassSeparator`. This allows the data to be regarded as a set of sections that can be identified by number. By identifying a `SerialContentSeparator` this process can be repeated at a lower level, within a section. There is a simple example in the VTAP Configuration Guide.

Setting	Definition	Default value	Example value	Options
`SerialMode`	Activate the Serial RS232 interface for receiving mobile pass data.	`=1`	`=0`	Set this to =1 [default] if you want VTAP data sent over the Serial RS232 interface, =0 switches it off
`SerialPassContentMode`	Turns on a subdivision of a data section into content sections.	`=0`	`=1`	=0 off (default), =1 on. To use this mode you must also supply SerialPassContentSeparator and SerialPassContentSection.

Setting	Definition	Default value	Example value	Options
SerialPassContentSection	Identifies the content section of interest, based on finding SerialPassContentSeparator characters. First content section before a separator is 0.	=0	=1	Only used if SerialPassContentMode=1 and SerialPassContentSeparator is set.
SerialPassContentSeparator	Identifies the content separator character in the data which bounds content sections.	=%	=%	Only used if SerialPassContentMode=1 and SerialPassContentSection is set.
SerialPassLength	Number of characters of data to extract, used in conjunction with SerialPassStart.	=0	=10	Defaults to 0 meaning do not truncate data, send all available characters from the pass data, or extracted section of pass data.
SerialPassMode	Sets whether to extract parts of the data from the mobile NFC pass.	=0	=1	=0 send data unchanged [default] so any other SerialPass... settings are ignored (except SerialPassStart or SerialPassLength used to truncate the data), =1 extract data based on other SerialPass... settings .
SerialPassSection	Identifies the section of interest, based on finding SerialPassSeparator characters. First content section before a separator is 0.	=0	=2	Used only if SerialPassMode=1 and SerialPassSeparator is set.
SerialPassSeparator	Identifies the separator character in the data which bounds sections.	=|	=|	Only used if SerialPassMode=1 and SerialPassSection is set.

Setting	Definition	Default value	Example value	Options
SerialPassStart	Number of characters into string to start extracting data, where first character is 0.	=0	=5	Used with SerialPassLength.
NEW OPTIONS				
SerialPostfix	Adds special characters after data, if needed.	=%0D	=end%0D	Use standard ASCII hex characters to describe the keystroke to append. In the example %0D adds a carriage return after the data. $t would include the pass type (a, g, 0, 2, 4, or 5, indicating Apple, Google, MIFARE (0) , NFC type 2, 4 or 5). (Use $$t if '$t' is required in the output.) tn will follow the pass type identifier with a digit for VAS merchant ID/ST collector ID index (1 to 6) and a digit for keyslot used (1 to 6).

Setting	Definition	Default value	Example value	Options
NEW OPTIONS				
SerialPrefix	Adds special characters before the data, if needed.	[Empty]	=%0Astart	Use standard ASCII hex characters to describe the keystroke to append. In the example %0A adds a new line linefeed before the data. $t would include the pass type (a, g, 0, 2, 4, or 5, indicating Apple, Google, MIFARE (0) , NFC type 2, 4 or 5). (Use $$t if '$t' is required in the output.) tn will follow the pass type identifier with a digit for VAS merchant ID/ST collector ID index (1 to 6) and a digit for keyslot used (1 to 6).
SerialSettings	Optional setting used to control the transfer of data over a serial connection.	=9600,n,8,1	=9600,n,8,1	Port settings: baud rate, parity, data bits, stop bits. Defaults to 9600,n,8,1 if not specified. Baud rate can be set to 115200 instead of 9600. Parity could be o for odd, e for even or n for no parity. 8 data bits means that groups of 10 data bits will be sent over the serial link, a start bit, 8 data bits and a stop bit. The stop bit can be transferred for 1, 1.5 or 2 times the time used to transfer a normal bit.

Setting	Definition	Default value	Example value	Options
NEW OPTIONS				
SerialSource	SerialSource controls which types of event data (pass reads, card/tag reads, serial commands) go to the Serial interface.	=A1	=80	The values are determined from a bitwise combination of the following hexadecimal values. Source options include these bit values for mobile passes and tag UIDs: Bit 7 (0x80) = Mobile Pass (Apple VAS/Google Wallet Smart Tap) Bit 6 (0x40) = STUID Bit 5 (0x20) = Write to card emulation mode (CardEmulationMode=1 see **VTAP NFC tag emulation settings**) Bit 4 (0x10) = RFU Bit 3 (0x80) = RFU Bit 2 (0x40) = RFU Bit 1 (0x20) = Command interface messages (>interface:type:message see **Dynamic configuration commands**) Bit 0 (0x10) = Card/Tag UID So, for example setting SerialSource=80 (hex) will send only NFC mobile pass data to the serial interface. Use =A1 to send data from NFC passes, NFC cards/tags and card emulation or =83 to send passes, cards/tags and serial commands
SerialStartup	Optional setting for transfer of data over a serial connection.	="VTAP100/<version>\r"	=startup message	Defaults to "VTAP100/<version>\r". May be disabled by setting it to empty.

2.1.10 Serial2 interface

A list of all valid configuration settings for the `config.txt` file, which relate to sending pass information over the Serial2 port, which supports an RS485 connection on a VTAP100 with a suitable expansion board..

Note: The Serial2 port is only available on VTAP100 v4 hardware onwards.

If `Serial2Mode=1` and `Serial2Source` is set, you can open an application on the computer connected to the VTAP reader, and pass or card data that is read will be sent over the Serial2 interface. `Serial2Source` determines whether the Serial2 interface is used to send mobile pass reads only, or cards/tag reads too.

You might need `Serial2Settings` to change the standard parameters for data transfer from the 9600, n, 8, 1 default. Serial2RS485 is used to set up RS485 transmission over the Serial2 interface, rather than RS232.

When `Serial2PassMode=1` you can use all of the other settings beginning `Serial2Pass...` to extract only part of the mobile pass data. To do this you have to start by identifying a separator character in the data as `Serial2PassSeparator`. This allows the data to be regarded as a set of sections that can be identified by number. By identifying a `Serial2ContentSeparator` this process can be repeated at a lower level, within a section. There is a simple example in the VTAP Configuration Guide.

Setting	Definition	Default value	Example value	Options
`Serial2Mode`	Activate the Serial2 interface for receiving mobile pass data.	`=1`	`=0`	Set this to =1 [default] if you want VTAP data sent over the Serial2 interface, =0 switches it off
`Serial2PassContentMode`	Turns on a subdivision of a data section into content sections.	`=0`	`=1`	=0 off (default), =1 on. To use this mode you must also supply Serial2PassContentSeparator and Serial2PassContentSection.

Setting	Definition	Default value	Example value	Options
Serial2PassContentSection	Identifies the content section of interest, based on finding Serial2PassContentSeparator characters. First content section before a separator is 0.	=0	=1	Only used if Serial2PassContentMode=1 and Serial2PassContentSeparator is set.
Serial2PassContentSeparator	Identifies the content separator character in the data which bounds content sections.	=%	=%	Only used if Serial2PassContentMode=1 and Serial2PassContentSection is set.
Serial2PassLength	Number of characters of data to extract, used in conjunction with Serial2PassStart.	=0	=10	Defaults to 0 meaning do not truncate data, send all available characters from the pass data, or extracted section of pass data.
Serial2PassMode	Sets whether to extract parts of the data from the mobile NFC pass.	=0	=1	=0 send data unchanged [default] so any other Serial2Pass... settings are ignored (except Serial2PassStart or Serial2PassLength used to truncate the data), =1 extract data based on other Serial2Pass... settings.
Serial2PassSection	Identifies the section of interest, based on finding Serial2PassSeparator characters. First content section before a separator is 0.	=0	=2	Only used if Serial2PassMode=1 and Serial2PassSeparator is set.
Serial2PassSeparator	Identifies the separator character in the data which bounds sections.	=|	=|	Only used if Serial2PassMode=1 and Serial2PassSection is set.

Setting	Definition	Default value	Example value	Options
Serial2PassStart	Number of characters into string to start reading data, where first character is 0.	=0	=5	Used with Serial2PassLength.
NEW OPTIONS Serial2Postfix	Adds special characters after data, if needed.	=%0A	=end%0D	Use standard ASCII hex characters to describe the keystroke to append. In the example %0D adds a carriage return after the data. $t would include the pass type (a, g, 0, 2, 4, or 5, indicating Apple, Google, MIFARE (0) , NFC type 2, 4 or 5). (Use $$t if '$t' is required in the output.) tn will follow the pass type identifier with a digit for VAS merchant ID/ST collector ID index (1 to 6) and a digit for keyslot used (1 to 6).
NEW OPTIONS Serial2Prefix	Adds special characters before the data, if needed.	[Empty]	=%0Astart	Use standard ASCII hex characters to describe the keystroke to append. In the example %0A adds a new line linefeed before the data. $t would include the pass type (a, g, 0, 2, 4, or 5, indicating Apple, Google, MIFARE (0) , NFC type 2, 4 or 5). (Use $$t if '$t' is required in the output.) tn will follow the pass type identifier with a digit for VAS merchant ID/ST collector ID index (1 to 6) and a digit for keyslot used (1 to 6).
Serial2Settings	Optional setting used to control the transfer of data over the serial2 connection.	=9600,n,8,1	=9600,n,8,1	Port settings: baud rate, parity, data bits, stop bits. Defaults to 9600,n,8,1 if not specified.

Setting	Definition	Default value	Example value	Options
NEW OPTIONS				
`Serial2Source`	Serial2Source controls which types of event data (pass reads, card/tag reads, serial commands) go to the Serial2 interface.	`=A1`	`=80`	The values are determined from a bitwise combination of the following hexadecimal values. Source options include these bit values for mobile passes and tag UIDs: Bit 7 (0x80) = Mobile Pass (Apple VAS/Google Wallet Smart Tap) Bit 6 (0x40) = STUID Bit 5 (0x20) = Write to card emulation mode (`CardEmulationMode=1` see **VTAP NFC tag emulation settings**) Bit 4 (0x10) = RFU Bit 3 (0x80) = RFU Bit 2 (0x40) = RFU Bit 1 (0x20) = Command interface messages (`>interface:type:message` see **Dynamic configuration commands**) Bit 0 (0x10) = Card/Tag UID So, for example setting Serial2Source=80 (hex) will send only NFC mobile pass data to the serial2 interface. Use =A1 to send data from NFC passes, NFC cards/tags and card emulation or =83 to send passes, cards/tags and serial commands
`Serial2RS485`	Automatically enables transmit driver for RS485 transmission, rather than RS232 over the serial2 interface.	`=1`	`=0`	=1 indicates the serial2 interface is used for RS485 (on suitable hardware)

2.1.11 LED control

A list of all valid configuration settings which relate to LED behaviour for use in the `config.txt` file.

Note: VTAP100 and some VTAP50 readers have factory fitted LEDs, but these settings can also control an external RGB or serial LEDs fitted.

The VTAP100 has diagnostic LEDs for factory use and user feedback LEDs.

Use `LEDMode` to control the behaviour of any diagnostic LEDs. `LEDSelect` chooses to use user feedback LEDs in a position compatible with the case around the VTAP100. `LEDDefaultRGB` sets the default colour for those user feedback LEDs, when pass reads or errors are not being signalled.

`PassLED` chooses the colour to flash on user feedback LEDs, with duration and repeats, when a mobile pass is read. `TagLED` does the same for cards/tags. `PassErrorLED` chooses the colour to flash on user feedback LEDs, duration and repeats, when an error occurs during a mobile pass read.

There is a simple example in the VTAP Configuration Guide.

Setting	Definition	Default value	Example value	Options
`LEDMode`	Chooses a default mode of operation for diagnostic LEDs	`=0`	`=1`	=0 Factory use LEDs 1-4 will flash in sequence but all on during NFC activity =1 Factory use LEDs all off but all on during NFC activity =2 Factory use LEDs will flash in sequence but no reaction to NFC activity =3 Factory use LEDs always off

Setting	Definition	Default value	Example value	Options
LEDSelect	Selects which RGB LED to use in response to pass reads, to suit the type of case around a VTAP100. For a VTAP50 using an external RGB LED board, use it to select the appropriate common cathode or common anode connection.	=1	=2	=0 External RGB LED (common cathode), =1 on VTAP100: on-board LED pair shows through compact case window; On VTAP50: external RGB LED (common anode), =2 on VTAP100: on-board LED pair shows through square case window; On VTAP50: external RGB LED (common anode), =3 on VTAP50 only: External and on-board serial LEDs, where fitted.
PassErrorLED	Chooses LED colour to flash on presentation of mobile NFC pass, if an error occurs	N/A	=FF0000,500	LED colour (use any hex RGB colour value) , LED on ms, LED off ms, number of repeats Example sets a long 500ms flash in red on presentation of mobile NFC pass, if an error occurs
PassLED	Chooses LED colour to flash on successful presentation of mobile NFC pass	N/A	=FF8000,100,50,2	LED colour (use any hex RGB colour value), LED on time in ms, LED off time in ms, number of repeats Example sets two 100ms orange LED flashes, spaced by 50ms, on presentation of mobile NFC pass
TagLED	Chooses LED colour to flash on presentation of a card/tag	N/A	=00FF00,500	LED colour (use any hex RGB colour value), LED on time in ms, LED off time in ms, number of repeats Example sets 1 500ms green LED flash on presentation of card/tag

Setting	Definition	Default value	Example value	Options
LEDDefaultRGB	Chooses the default colour of the RGB LED. [On VTAP50 v2 only: A serial LED pattern can be chosen.]	=FFFFFF	=FF8000 =FFFFFF:seq.comet@leds.ini	Set =FF8000 sets the LED to a warm orange colour as its default state, use any hexadecimal RGB value. The second option sets the LED to white on most VTAPs, but on a VTAP50 v2 with serial LEDs will follow the seq.comet section of the leds.ini file.
LEDMaxSerial	Limits the maximum number of LEDs in a serial LED chain or matrix. [VTAP50 v2 only] Not required for external RGB LEDs.	=24	=24	The VTAP50 v2 has a limit of 1 to 255 serial LEDs in a chain. If future versions permit longer chains, there may be a need to limit the number used, for backward compatibility.

2.1.12 Buzzer control

A list of all valid configuration settings which relate to buzzer behaviour for use in the `config.txt` file.

`PassBeep` chooses the buzzer duration and repeats to trigger when a mobile pass is read, following the same pattern as LED controls. `TagBeep` does the same for cards/tags. `PassErrorBeep` chooses the buzzer duration and repeats to trigger when an error occurs during a mobile pass read.

There is a simple example in the VTAP Configuration Guide.

Setting	Definition	Default value	Example value	Options
`PassBeep`	Sets a beep on successful presentation of mobile NFC pass	`N/A`	`=100,50,2`	Beep on ms, beep off ms, number of repeats Example sets two 100ms beeps, spaced by 50ms, on presentation of mobile NFC pass
`PassErrorBeep`	Sets a beep on presentation of mobile NFC pass, if an error occurs.	`N/A`	`=500`	Duration in ms Example sets a long 500ms beeps on presentation of mobile NFC pass if ; an error occurs
`TagBeep`	Sets a beep on presentation of card/tag	`N/A`	`=100`	Beep on ms, beep off ms, number of repeats Example sets 1 100ms beep on presentation of a card/tag

2.1.13 VTAP NFC tag emulation

A list of all configuration settings which relate to VTAP NFC tag emulation for the `config.txt` file.

VTAP can emulate an NFC Forum Type 4 card or tag, containing NDEF encoded data. If you tap a phone on a VTAP in card emulation mode, a URL could be launched or Text data displayed on the tapping phone.

The VTAP emulated tag can also be written to, for example by an Android or iOS app. Any NDEF record data written by the app will not overwrite the emulated tag data, but instead is intercepted by the VTAP and could be output over any of the VTAP comms interfaces, treated as though it was a data read from an NFC tag.

`CardEmulationMode` lets you enable VTAP NFC tag emulation and `CardEmulationData` sets the data that will be passed when the VTAP is in a card emulation mode. These settings can be adjusted dynamically using the `?card` and `?cardmode` commands in the section **Dynamic configuration commands**.

There is more information in the VTAP Application Notes.

Setting	Definition	Default value	Example value	Options
NEW `CardEmulationMode`	This enables or ends a mode, where the VTAP emulates an NFC Forum Type 4 card or tag, containing NDEF encoded data.	`=0`	`=1`	=0 to leave card emulation mode, =1 to enter card emulation mode, =2 to enter mixed mode, where the VTAP reads tags and passes but will also emulate a card. For more detail refer to VTAP Application Notes.

Setting	Definition	Default value	Example value	Options
NEW CardEmulationData	This defines the NDEF data to be sent by a VTAP reader that is in card emulation mode.	N/A	=TEXT,Hello World! =URI,http://www.vtapnfc.com =RAW,D101055402656e4869 =FILE,tagdata.txt	Format is =<type><:lang>,<NDEF> <NDEF> data is assumed to be text if a <type> TEXT/URI/RAW/FILE is not included TEXT (T) type is assumed to be English (en) unless another 2 character <lang>uage code is used URI (U) requires a valid web address RAW (R) is a raw binary message part for an NDEF record. Its length will be automatically set. FILE (F) points to a text file on the VTAP containing one of the other command types. For more detail refer to VTAP Application Notes.

2.1.14 Other settings

Other valid configuration settings for use in the `config.txt` file .

`kbmap` optionally points to a separate file, where a list of key codes corresponding to particular characters, can be used to translate between the VTAP default US keyboard emulation and the keyboard language setting used by the host computer.

`MassStorageEnable` provides an alternative to a hardware lock on VTAP readers where jumpers may be difficult to access, but there is an enabled host connection via serial or USB.

`NFCDefaultEnable` allows the VTAP reader NFC field to be turned on or off to manage power consumption. `StartupDelayMS` provides a delay during start up to allow power to fully stabilise, for use primarily in situations with a Wiegand interface.

Setting	Definition	Default value	Example value	Options
NEW `kbmap`	Points to a keyboard map file and/or section of that file, which lists key codes corresponding to particular characters, in order to translate to a keyboard language setting used by the host computer.	`N/A`	`=en-uk@kbmap.txt`	The syntax for this setting is kbmap=<section>@<file_name>, where the <file_name> defaults to kbmap.txt if omitted. <section> refers to a particular keyboard language such as en-uk, as the file may contain maps for multiple languages although the kbmap setting can only use one at a time. If you need to use this feature please contact vtap-support@dotorigin.com for suitable kbmap.txt example files.

Setting	Definition	Default value	Example value	Options
NEW				
MassStorageEnable	Allows a remote host to completely remove or restore mass storage access to VTAP readers with a config.txt setting. An alternative to a hardware lock, which avoids the need for changes to jumpers on the PCB, which may not be readily accessible.	=1	=0	Default value is enabled. CAUTION: Only disable (=0) where there is an enabled host connection via serial or USB that can be used later to update the configuration. If you disable mass storage without enabled serial or USB command interfaces you will be locked out of your VTAP reader, and have to return it to Dot Origin for recovery!
NFCDefaultEnable	Determines whether NFC is enabled [default] or disabled at startup, in order to manage power consumption.	=1	=0	In the example NFC will be disabled at startup, to save power.
StartupDelayMS	Delay in milliseconds before fully starting up to allow the power supply to stabilise.	=1000	=5000	Use a value such as 5000 when using an external power supply. This could prevent possible file system corruption during installation, if VTAP could be wired up to a live external power supply (typically when using Wiegand or RS485 expansions).
ZmodemRxTimeout	Set the receive timeout to an appropriate duration.	=1000	=2000	Number of milliseconds

2.2 command.txt

This section lists all valid commands for `command.txt`. This is a file which must contain the string `!VTAPcommand` at the start.

There is a simple example in the VTAP Configuration Guide.

Command	Definition	Default value	Example value	Options
`reboot`	Power cycle the VTAP, as if the USB cable was disconnected momentarily.	`N/A`	`N/A`	
`refresh`	Force the VTAP to re-read `config.txt`. This useful if you have renamed a file to be `config.txt` as file renaming is not automatically recognized by the VTAP.	`N/A`	`N/A`	
`remount`	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.	`N/A`	`N/A`	

2.3 lock.txt

This section lists all valid commands for `lock.txt` which is needed to work with the software configuration lock. This is a file which must contain the string `!VTAPlock` at the start.

There is a simple example in the VTAP Configuration Guide.

Command	Definition	Default value	Example value	Options
`lock`	Sets the password to the given value and locks the VTAP.	`N/A`	`=APa55word`	
`unlock`	Offers the given password to unlock the VTAP.	`N/A`	`=APa55word`	

3 Commands

These are all of the commands that can be sent over a serial interface to control your VTAP reader.

Your command interface could be a virtual COM port or serial port. The commands can be sent using a program such as TeraTerm or PuTTY-nd for instance.

Situations where these might be used are described in the VTAP Serial Interface Guide. These include managing a VTAP reader in passive mode, changing `config.txt` or transferring files to the VTAP reader remotely.

For ease of use the commands are split into three types: **Data request commands**, **Remote management commands** and **Dynamic configuration commands**.

3.1 Data request commands

This section lists valid commands to send over a Command Interface, either a virtual COM port or serial port, using a program such as TeraTerm or PuTTY-nd for instance. You may also want to use commands in the **Remote management commands** and **Dynamic configuration commands** sections.

These commands request data stored in the VTAP reader without changing it in any way.

A `<setting>` refers to any of those listed in the section **2.1** .

Command	Definition	Example value	Options
`%<setting>`	Request the current value of any setting that could be included in the `config.txt` file.	`%TagLED`	Multiple settings can be requested in a single line command, by separating each setting with the | character. Using % on its own will return all current settings.
`?b`	Reads VTAP boot information including firmware version and unique serial number.		
NEW `?events`	Polls for any events (such as configuration change) that have happened in the VTAP reader since it was last polled for events.	`?events 00000001`	A 32-bit hex number supplied with the command is a bit mask to select only certain events be reported. The default without a bit mask is to report all events, equivalent to using bit mask ffffffff.Currently configuration change is the only event recorded. If 00000001 is returned by the command then the configuration has been updated since last poll. After reporting an event it is cleared. If the bit mask used prevents the reporting of any event, it will remain pending for a subsequent poll.

Command	Definition	Example value	Options
`?h`	Retrieve public key hash information for the VTAP ECC keyslots, which can be used to work out which private keys have been loaded. The public key hash is the first 4 bytes of the 32 byte SHA-256 hash over the 32 byte ECC public key.		The block of public key hash data returned by the command is 32 byte string. The string starts with `A5`. The public key data for the 6 key slots follows in order, 1 through 6. Each key slot is described by a 1 byte flag (00=key slot empty, 01=key slot in use) then the 4 byte public key hash for that slot. After all 6 keys the string ends with `5A`.
`?r`	Reads last NFC pass, card/tag data (as long as it was received inside the last `InvalidDataCacheMS` period).		This is only used in passive mode (selected with ?p)
`?t`	Returns the type of the most recent cached NFC pass, card/tag read as a single character (as long as it was received inside the last `InvalidDataCacheMS` period) over the COM port.	`?t COM`	Qualifiers instruct the VTAP to send an indicator of the type of NFC pass, card/tag data over a particular interface. `?t COM` sends that NFC pass, card/tag data over COM port. This is the default for `?t` when an interface is not specified. Returned values are A for an Apple VAS pass, G for a Google Smart Tap pass, 1 for 2 for NFC Forum type 2 (eg MIFARE Ultralight), 4 for NFC Forum type 4 (DESFire), 5 for NFC Forum type 5 (ISO15693), 0 for MIFARE Classic. Only used in passive mode (selected with ?p).
NEW `?temp`	Returns the internal temperature of the VTAP.		Returns a celsius value as reported by the processor.

Command	Definition	Example value	Options
`?type`	Returns the type of the most recent cached NFC pass, card/tag read as a single character (as long as it was received inside the last `InvalidDataCacheMS` period) over the COM port or serial port, in passive mode.	`?type`	Returned values are A for an Apple VAS pass, G for a Google Smart Tap pass, 2 for NFC Forum type 2 (eg MIFARE Ultralight), 4 for NFC Forum type 4 (DESFire), 5 for NFC Forum type 5 (ISO15693), 0 for MIFARE Classic. Only used in passive mode (selected with ?p). For Apple of Google passes the pass type is followed by VAS merchant ID/Smart Tap collector ID index and the key slot in use. In other cases -- follows the type.

3.2 Dynamic configuration commands

This section lists valid dynamic commands to send over a Command Interface, either a virtual COM port or serial port, using a program such as TeraTerm or PuTTY-nd for instance. You may also want to use commands in the **Remote management commands** and **Data request commands** sections.

Dynamic commands are used to manage the behaviour of a VTAP reader in passive mode, in real time. They support close integration of your VTAP reader with other systems. These commands are not saved in `config.txt` so they will only be valid until the next `?x` command, **reboot or refresh**, when the VTAP reader will return to following the settings in `config.txt`.

A `<setting>` refers to one of those listed in the section **2.1** .

Command	Definition	Example value	Options
NEW OPTIONS			
`>interface:type:message`	Send a message (for example as a virtual pass or tag payload) from any ComPort or serial interface to any of the other interfaces. Note: Sending to a Wiegand interface >W is a special case described separately.	`>K:A:hello`	*Interface* is one of k, c, s or t, where k=Keyboard, c=ComPort, s=Serial, t=Serial2. *Type* is one of a, g, 0, 2, 4, or 5, indicating Apple, Google, MIFARE (0) , NFC type 2, 4 or 5 . The example here will send “hello”, as an Apple pass payload, to the keyboard emulation interface. Or set the value of *Interface*Source to 83, to enable serial commands over the corresponding interface. Example: When sending a message to ComPort, set ComPortSource=83 to enable pass read, card/tag reads and serial commands over the virtual COM port.

Command	Definition	Example value	Options
NEW OPTIONS			
`>W:T:YY:XXXXXXXX`	Send a message (for example as a virtual pass or tag payload) from any of the serial interfaces to a Wiegand interface [VTAP100-PAC-W only]. This is slightly different from the general case, as the message must take the form of a bit pattern and requires a bit length value.	`>W:A:06:B7A207`	*T* is type a, g, 0, 2, 4, or 5, indicating Apple, Google, MIFARE (0) , NFC type 2, 4 or 5 , *YY* is the Wiegand bit length in hex and *XXXXXXXX* is the Wiegand data in hex. (This is left justified, which means any padding bits to make a multiple of 8 will be in the LS bits). The example here will send the 6 bit “B7A207”, as an Apple pass payload, to the Wiegand interface.
`?a`	Change COM port to active mode. Mobile pass, card/tag data will then be sent over the COM port automatically, whenever it is received.		
DEPRECATED			
`?BEEP`	The buzzer can be driven by this command over any serial command interface. *Use of ?BEEPR now preferred.*	`?BEEP 200,200,2`	Beep on ms, beep off ms, number of repeats In the example, the buzzer will beep twice for 200ms on and 200ms off.
`?BEEPR`	The buzzer can be driven by this command over any serial command interface.	`?BEEPR 200,200,2`	Beep on ms, beep off ms, number of repeats. In the example, the buzzer will beep twice for 200ms on and 200ms off. Response provided OK<newline> or FAIL<newline>.

Command	Definition	Example value	Options

NEW
`?card <type><:lang>,<NDEF>`
	Set the `CardEmulationData` value dynamically over a serial interface. This defines the NDEF data to be sent by a VTAP reader that is in card emulation mode.	`?card TEXT,Hello World!`
		`?card URI,http://www.vtapnfc.com`
		`?card RAW,D101055402656e4869`
		`?card FILE,tagdata.txt`
			<NDEF> data is assumed to be text if a <type> is not included
			TEXT (T) type is assumed to be english (en) unless another 2 character <lang>uage code is used
			URI (U) requires a valid web address
			RAW (R) is a raw binary message part for an NDEF record. Its length will be automatically set.
			FILE (F) points to a text file on the VTAP containing one of the other command types.
			For more detail refer to VTAP Application Notes.

NEW `?cardmode <#>`	Set the `CardEmulationMode` value dynamically over a serial interface. This enables or ends a mode, where the VTAP emulates an NFC Forum Type 4 card or tag, containing NDEF encoded data.	`?cardmode 1`	=0 to leave card emulation mode, =1 to enter card emulation mode, =2 to enter mixed mode, where the VTAP reads tags and passes but will also emulate a card. For more detail refer to VTAP Application Notes. After this command the VTAP returns the number of its current `CardEmulationMode`. Omitting a mode number <#> will also return the current `CardEmulationMode`.

Command	Definition	Example value	Options
DEPRECATED			
?k	Consumes any key files (such as privateX.pem or appkey#.txt) into secure storage, from the file system, without requiring a restart. Then remount the USB mass storage device after loading keys, forcing the USB host, such as Windows, to update its cache of the filesystem directory. *Use of ?KEYLOAD now preferred.*		
?KEYLOAD	Consumes any key files (such as privateX.pem or appkey#.txt) into secure storage, from the file system, without requiring a restart. Then remount the USB mass storage device after loading keys, forcing the USB host, such as Windows, to update its cache of the filesystem directory.		Response provided OK<newline> or FAIL<newline>.
?l	To lock the VTAP configuration using a password over any serial command interface.	?l APa55word	The example will lock the VTAP configuration using the password APa55word.

Command	Definition	Example value	Options
DEPRECATED			
?LED	The LED [Not VTAP50 v1] can be driven with this command over any serial command interface. Sequences of RGB LED flashes can be triggered, specifying the hexadecimal RGB colour. [VTAP50 v2 only: A serial LED pattern can be chosen.] *Use of ?LEDR now preferred.*	?LED FF0000,100,100,5 ?LED FFFFFF:seq.test@leds.ini	LED colour (use any hex RGB colour value), LED on ms, LED off ms, number of repeats. In the example, the LED will flash red 5 times for 100ms on and 100ms off. [VTAP50 v2 only: follow the second example to instruct the LEDS to follow the seq.test portion of the leds.ini file in driving the chain of serial LEDs].
?LEDR	The LED [Not VTAP50 v1] can be driven with this command over any serial command interface. Sequences of RGB LED flashes can be triggered, specifying the hexadecimal RGB colour. [VTAP50 v2 only: A serial LED pattern can be chosen.]	?LEDR FF0000,100,100,5 ?LED FF0000:seq.test@led.ini	LED colour (use any hex RGB colour value), LED on ms, LED off ms, number of repeats. In the example, the LED will flash red 5 times for 100ms on and 100ms off. [VTAP50 v2 only: follow the second example to instruct the LEDS to follow the seq.test portion of the leds.ini file in driving the chain of serial LEDs]. Response provided OK<newline> or FAIL<newline>.
?NFC	To enable/disable the NFC field, to reduce power consumption. If no value is supplied with the command, it will read the current state of the NFC field, returning ON or OFF.	?NFC 1	The example will enable the NFC field

Command	Definition	Example value	Options
`?p`	Change the COM port to passive mode. The COM port will then only send data in response to specific commands.		
`?REBOOT`	To reboot the VTAP over any serial command interface.		Response provided OK<newline> or FAIL<newline>.
`?STPasses`	To dynamically change the Google Smart Tap pass enabled over any serial command interface. Avoids the need for frequent edits to `config.txt` when pass types often need to be changed.	`?STPasses 4`	The example will enable only Google SmartTap pass type ST4, if it is present in `config.txt`
`?u`	To unlock the VTAP configuration using a password over any serial command interface.	`?u APa55word`	The example will unlock the VTAP configuration, using the password APa55word
`?VASPasses`	To dynamically change the Apple VAS passes enabled over any serial command interface. Avoids the need for frequent edits to `config.txt` when pass types often need to be changed.	`?VASPasses 3,4,5`	The example will enable only Apple VAS passes VAS3, VAS4 and VAS5, if they are present in `config.txt`
DEPRECATED			
`?x`	To reboot the VTAP over any serial command interface. *Use of ?REBOOT now preferred.*		

3.3 Remote management commands

This section lists the valid remote management commands to send over a Command Interface, either a virtual COM port or serial port, using a program such as TeraTerm or PuTTY-nd for instance. You may also want to use commands in the **Data request commands** and **Dynamic configuration commands** sections.

These commands directly change `config.txt` or transfer files to the VTAP reader remotely.

A `<setting>` refers to one of those listed in the section **2.1** .

Command	Definition	Example value	Options
`$<setting>=<value>`	Set any setting that could be included in the `config.txt` file. Note: The mass storage device will be temporarily dismounted after this action, to refresh operation of the VTAP.	`$TagLED=FF00FF,300`	Multiple settings (up to 10) can be made in a single line command, by separating each setting with the <tab> character. It is not recommended that you use this setting too frequently as it can impact flash memory life. If you require frequent changes in behaviour, use **dynamic commands**.
`!<filename>`	This instructs the VTAP to send a copy of its file. Having sent this command to the VTAP you need to run zmodem receive in the client to collect this file. It will continue being sent until a zmodem acknowledgement is received.	`!boot.txt`	In the example a copy of the file `boot.txt` will be sent over zmodem.