VTAP

Application Note - Access Control (Wiegand) cards, passes and formats

Firmware from v2.2.6.0

VTAP100-PAC-W

Revised September 2024 v1.01

DOT ORIGIN

If you need help to set up or use your VTAP reader, beyond what is contained in this Application Note, 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 a VTAP reader for access control

This Application Note discusses configuration settings to manipulate the format of data from passes and cards, when it is sent from the VTAP100 Wiegand reader into your access control system.

There are two VTAP100 PAC (Physical Access Control) readers that support the technologies commonly used for access control:

- A VTAP100 Wiegand reader (VTAP100-PAC-W) can interface directly with a Wiegand controller, as a door or turnstile reader in an access control system.
- A VTAP100 RS-485/OSDP reader (VTAP100-PAC-485) will support the Open Supervised Device Protocol, as a Peripheral Device (PD) in secure communication with an Access Control Unit (ACU).

Refer to the VTAP Application Note on Using OSDP for more about the use of a VTAP100 RS-485/OSDP reader. When OSDP is used, some of the Wiegand data manipulation described here may be needed, but the use of OSDP on VTAP readers is not limited to access control.

2 Using a VTAP Wiegand reader for access control

A VTAP100 Wiegand reader (VTAP100-PAC-W) can interface directly with a Wiegand controller, as a door or turnstile reader in an access control system.

Note: The Wiegand interface is only available on VTAP100-PAC-W. The `Wiegand`... settings will not have any effect on any other VTAP model.

Many customers want to replace an existing access control reader with a VTAP reader, in order to add the capability to use NFC wallet passes in addition to using an existing population of cards or tags. This may be during a temporary migration phase in the roll out of mobile wallet passes, or to create a permanent hybrid solution, able to accept both cards/tags and NFC mobile passes.

There are several steps in adding a VTAP reader to an access control system with a Wiegand interface:

1. **Check your access control system data requirement:**

   Check the format of data expected by your access control system from existing cards or tags, if any. Your access control system may already accept credentials from more than one type of card or tag. For each type of card or tag, find out:

   - Does your system use the card UID or block data?
   - How many bits of data are transferred?
   - Are digits used by the access control system decimal or hexadecimal?
   - In which order are the bytes read?
   - Is there any site or facility code?
   - Are there any checksum bits?

   A good understanding of your access control system in this step, will make it much easier later to achieve a configuration where interchangeable, equivalent data can be extracted from both mobile passes and cards/tags.

2. **Mobile pass design:**

   Design a mobile pass containing the type of data required by your access control system. If this will be used in an existing or hybrid access control system, ensure the pass design includes either the ASCII, hex or decimal version of the binary data you store on existing cards/tags.

3. **VTAP reader configuration:**

   Connect the VTAP reader to a PC using a USB cable.

   Edit the VTAP `config.txt` file to **Send pass payload over a Wiegand interface**, in the format expected.

   If cards/tags will be presented to the VTAP reader in addition to mobile passes, don't forget to set up the additional configuration needed to **Send UID or block data from a card/tag over Wiegand**.

   There may then be some fine tuning of that configuration required, to ensure you **Format pass and card/tag data to match over Wiegand** so that mobile passes and cards/tags are truly equivalent in your system.

   You may also need to delay startup of the Wiegand interface by a number of milliseconds to allow the power supply to stabilise. We recommend that you use a value such as 5000ms (setting `StartupDelayMS=5000` in `config.txt`) when using an external power supply, to prevent possible file system corruption during installation if VTAP could be wired up to a live external power supply.

4. **Wire the VTAP reader Wiegand interface:**

   Safely remove the VTAP reader from the PC and connect the VTAP to a controller to your access control system following the wiring instructions in the VTAP100-PAC-W Installation Guide.

   Depending on the type of controller you are using, you may need to use controller configuration software to identify and link the reader.

2.1 Send pass payload over a Wiegand interface

The Wiegand interface allows a mobile NFC pass payload to be passed straight to an access controller from your VTAP reader, like data from any other card reader.

Note: The Wiegand interface is only available if you are using the VTAP100-PAC-W model.

To enable the Wiegand interface you will need to make changes to the `config.txt` file.

Example: Changes to `config.txt` to enable the Wiegand interface

```
!VTAPconfig

WiegandMode=1         ; Enable Wiegand interface
PassWiegandBits=56    ; See note below, this must match bit length expected
                      ;   by controller and data must contain this number
                      ;   of bits, =56 is default if omitted
```

Here `WiegandMode=1` enables data transmission over the Wiegand interface.

`WiegandSource` controls which types of data (pass reads, card/tag reads, serial commands) will be sent to the Wiegand interface. The default value is `A1`, which allows sending of all NFC pass and card/tag data. Refer to the VTAP Commands Reference Guide for other options, if data sources need to be restricted for your application.

Additional settings are needed if you want to **Send only part of pass payload**, which are discussed in the following section. And there are a number of settings which will allow you to adjust the **Format of pass data** before it is sent over the Wiegand interface, discussed later in this section.

Note: `PassWiegandBits` should be set to match the bit length expected by the controller and the pass payload must contain sufficient data to provide this number of bits. `PassWiegandBits` defaults to 56, so the expected form of the pass payload is 14 hex digits, unless `PassWiegandBits` is set to another value in your configuration.

2.1.1 Send only part of pass payload over Wiegand

`WiegandPassMode` allows you to extract only a part of each mobile NFC pass payload to send over the Wiegand interface. Setting `WiegandPassMode=1` enables all of the other `WiegandPass...` settings, to extract a short character sequence from the pass payload. This can then be interpreted as a decimal or hexadecimal number and sent over the Wiegand interface as a bit sequence. These settings let you fetch the specific section of the pass payload needed by your access control system, as in this example:

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
```

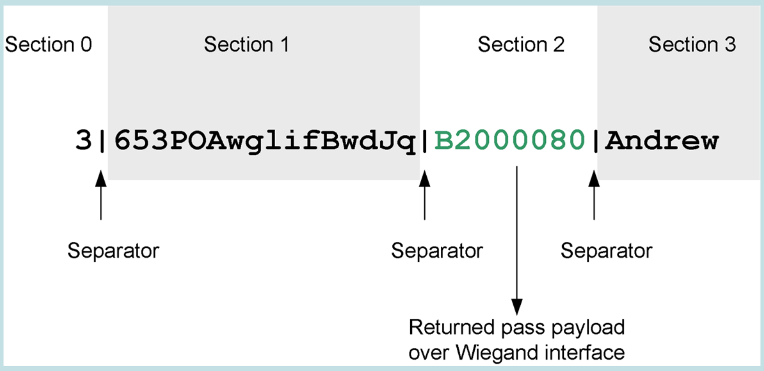
Section 0
Section 1
Section 2
Section 3
3|653POAwglifBwdJq|B2000080|Andrew
Separator
Separator
Separator
Returned pass payload over Wiegand interface

Figure 2-1 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

Refer to the VTAP Commands Reference Guide for all the possible `WiegandPass...` settings and options to extract even smaller parts of the pass payload.

2.1.2 Format of pass data sent over Wiegand

Wiegand data is usually a short bit pattern, rather than a sequence of characters. So there are several optional settings, to use in `config.txt`, which allow you to change the output format for any data read, in terms of bit length, parity bits and identification of pass type, for data transferred over a Wiegand connection:

- `PassWiegandBits=56` lets you specify the number of bits (1 to 255) to output over the Wiegand interface, from the start of the filtered pass payload. If omitted it defaults to 56.

  (`TagWiegandBits` does the same for card/tag data.)

- `PassFormat=d` is a setting to interpret ASCII pass payload characters as either hex (h) or decimal (d), when converting the pass payload to a Wiegand bit sequence.

  (For cards and tags containing a sequence of ASCII characters, you may want `TagWiegandASCIIFormat` set to hex (`h`), decimal (`d`) or the default ASCII (`a`), along with `TagReadFormat=a`.)

- `PassWiegandParity=1` adds a single 'parity bit' equivalent to pass payload. This makes it possible to use mobile pass data formats that include parity bits. Parity bit equivalents can be used if the parity bit(s) are not being tested for validity. `PassWiegandParity=2` adds calculated odd and even parity bits to the data. Either can be used if `PassFormat=d` or `PassFormat=h`. Again, the default `=0` turns this feature off.

  (Use `TagWiegandParity` to do the same operation for card and tag data, used with `TagReadFormat=a` and `TagWiegandASCIIFormat=d` or `=h` to interpret the tag byte data as an ASCII string representing a decimal or hex number, and to convert this to the corresponding Wiegand bit sequence by adding extra parity bits, which might be expected by the controller.)

- `WiegandPassTypeIdent=1` inserts an additional leading byte of pass type identifier (01 for Apple VAS, or 02 for Google ST) in the Wiegand output. This makes it possible to distinguish between cards/tags and mobile wallet passes. This setting overrides `PassWiegandBits` and results in a Wiegand bit length of 64 bits. The default `=0` turns this feature off.

2.2 Send UID or block data from a card/tag over Wiegand

Card/tag data can be passed straight to an access controller by your VTAP reader, like data from any other card reader. The most simple scenario is where the data required is the card/tag UID. Your `config.txt` file will need to contain settings like those used in this example:

Example: Reading UID from a MIFARE card/tag

```
!VTAPconfig

MIFAREClassic=U          ; Read UID data from MIFARE Classic card/tag
WiegandMode=1            ; Enable Wiegand interface
TagWiegandBits=0         ; Automatically determines the bit length from the
                         ;  extracted data.
```

Output on ComPort or other ASCII interface:
Hex (4-byte): C699E142

Output on Wiegand interface:
Hex (4-byte): C699E142
Binary: 11000110 10011001 11100001 01000010

When `TagWiegandBits=0`, the Wiegand bit length will be determined from the number of bytes of data that have been read. We recommend this automatic method where you have a mixed population of cards: For example a MIFARE Classic UID could be either 4 bytes or 7 bytes and will then produce 32 bit or 56 bit Wiegand data appropriately.

Setting a specific bit length value will cause a larger bit length UID to be truncated, or a smaller bit length UID zero-padded, to the achieve the set bit length. This may be better if your access controller only supports a small bit length data (eg 32 bit).

Additional settings are needed if you want to **Read block data from MIFARE Classic card/tags** or **Read DESFire card/tags**, which are discussed in the following sections.. And there are a number of settings which will allow you to adjust the **Format of card/tag data** before it is sent over the Wiegand interface, discussed at the end of this section.

2.2.1 Read block data from MIFARE Classic card/tag

If your application requires reading block data from a MIFARE Classic card/tag (`MIFAREClassic=B` or `=3`), you need to set the following additional parameters to define which block to read, and specify the key and its type, in order to read that block.

- `TagReadBlockNum` specifies the block number to read. Set a value between 0 to 255 (default is `TagReadBlockNum=0`). MIFARE Classic 1K cards have 64 blocks; 4k cards have 256 blocks, which are numbered in decimal from 0 to 255.
- `TagReadKey` is a hex value key, needed to read the block data. Set a 6-byte hexadecimal value corresponding to the key required to read the block chosen by `TagReadBlockNum`. There is no default for this.

  Alternatively, you can save the key needed to read the block data in a file called `appkey#.txt` on the VTAP reader and set `TagReadKeySlot=#` instead, where `#` is the key number, an integer from `1` to `9`.
- `TagReadKeyType` is the key type for the key identified in `TagReadKey`. Set `=A` or `=B` corresponding to the MIFARE Classic key type. Default is `TagReadKeyType=A`.

MIFARE block data specific settings are used in the following example:

Example: Read data from MIFARE Classic block 3 and sending over Wiegand and ASCII interfaces

```
!VTAPconfig

WiegandMode=1
MIFAREClassic=B            ; Read block data from MIFAREClassic card/tag
TagReadBlockNum=3          ; Read data from block 3
TagReadKey=AC42E155C409    ; Key to read the block 3
```

Outputs:

Hex (16-byte): 00000000000000000000000001378684

Binary: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000001 00110111 10000110 10000100

Note: Since `TagReadLength` is not defined, the default value of 16 bytes is used.

Remember that there are a number of additional settings which will allow you to adjust the **Format of card/tag data** before it is sent over the Wiegand interface if needed, discussed at the end of this section.

2.2.2 Read DESFire cards or tags over Wiegand interface

The VTAP Application Notes on DESFire contains a detailed discussion on reading DESFire cards/ tags and handling the output.

DESFire specific settings are used in the following example:

Example: Settings in config.txt to read data from DESFire cards or tags containing ASCII data and sending over Wiegand and ASCII interfaces

```
!VTAPconfig

WiegandMode=1
NFCType4=D                    ; Read DESFire data from NFCType4 cards
DESFire1AppID=92E672          ; 24 bit DESFire application ID to read
                              ;  (6 hex digits)
DESFire1FileID=26             ; The file ID to read, within the
                              ;  application (decimal 1 to 255)
DESFire1KeyNum=1
DESFire1KeySlot=3             ; Use the key saved in appkey3.txt
                              ;  on the VTAP reader
DESFire1Crypto=1              ; 3DES cryptography
DESFire1Format=1              ; Use Key-ID format (26bit facility code
                              ;  and number format) to read the
                              ;  stored data
```

Outputs:
Hex (3-byte): 323135
Binary: 10011010100110001001100100

If you use `DESFire#Format=1` to select Key-ID 26-bit format, you need to be aware that different sections of the data serve different functions, as separate facility code and card number, which affects how that data converts from binary to decimal or hex:

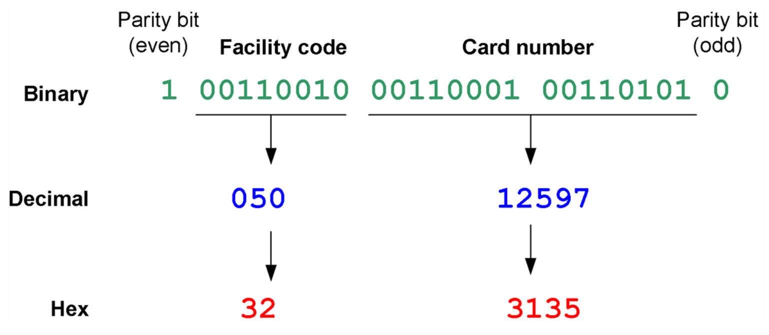

Figure 2-2 Translating Wiegand 26-bit binary data to decimal or hex as facility code + card number

In this case, if an output was sent to an ASCII interface, such as the COMport, that output would be a concatenation of the decimal facility code and decimal card number, 05012597.

Remember that there are a number of settings which will allow you to adjust the **Format of card/tag data** before it is sent over the Wiegand interface, discussed in the next section. In particular, if reading DESFire cards or tags when the `DESFire#Format` is not set (`=0`), you may want to use `TagWiegandASCIIFormat` (in conjunction with `TagReadFormat=a` and `DESFire#ReadLength`) since the output data format and length is not otherwise constrained.

Note: When using the Wiegand interface, multiple reads are not supported. In this case, only the lowest numbered `DESFire#...` settings will be used, which might not be `DESFire1...`. If only `DESFire3...` and `DESFire4...` settings are defined in `config.txt`, the `DESFire3...` settings would then be used for output over Wiegand.

2.2.3 Format of card/tag data sent over Wiegand

Before UID or block data is sent to a Wiegand interface you may need to perform a number of operations on that data, such as changing byte order, truncation, right shifting the bits, padding with zeroes or addition of parity bits. The settings required to perform these operations are discussed here:

- `TagByteOrder` will reverse the byte order of the block data or UID read from a card or tag. (Or use `TagByteOrderTypes` if you only want to reverse byte order for some card or tag types.)
- `TagReadFormat` sets the format of the extracted binary block data as ASCII (`a`), hex (`h`), or decimal (`d`).
  - When `TagReadFormat=a` (ASCII), each byte is an ASCII character.

    This will output the binary data from card/tag as an ASCII character sequence over ASCII interfaces (COM, Serial, Serial2, keyboard emulation). Use this setting along with `TagWiegandASCIIFormat` if the tag data is an ASCII sequence representing the digits of a decimal or hexadecimal value. If `TagReadFormat` is not set `=a`, binary data will be output to the Wiegand interface.
  - `TagReadFormat=d` will interpret the binary data as 64 bit decimal value for output to ASCII interfaces.

    If the tag data is a binary sequence (not an ASCII sequence) then use `TagReadFormat=d` or `=h` along with any other settings required to adjust the binary data for output over Wiegand.

  - `TagReadFormat=h` (default) converts the binary data to hex digits with 2 digits per byte for output to ASCII interfaces.

- `TagWiegandASCIIFormat` sets how to interpret tag data that contains an ASCII character sequence, representing a decimal or hexadecimal number, for output as a bit sequence over Wiegand. This setting only works when `TagReadFormat=a`.
  - Set `TagWiegandASCIIFormat=a` (default) to send the ASCII character byte values directly as a sequence of bits over Wiegand.
  - `TagWiegandASCIIFormat=d` to interpret the ASCII character sequence as a decimal number (up to 64 bits) when converting to a Wiegand bit sequence.
  - `TagWiegandASCIIFormat=h` to interpret the ASCII character sequence as a hexadecimal number when converting to a Wiegand bit sequence.
- `TagWiegandBits=56` lets you specify the number of bits (1 to 255) to output over the Wiegand interface, from the start of the extracted tag data. If omitted it defaults to 0, meaning that the Wiegand bit length is automatic: The number of bits is determined automatically from the number of bytes of tag data that have been read.
- `TagWiegandParity=1` adds a single 'parity bit' equivalent to the resulting bit sequence that is sent over Wiegand. This makes it possible to use tag or card data formats that include parity bits. Parity bit equivalents can be used if the parity bit(s) are not being tested for validity. `TagWiegandParity=2` adds calculated odd and even parity bits to the data. Either can be used with `TagReadFormat=a` and `TagWiegandASCIIFormat=d` or `=h` to get decimal or hex data. Again, the default `=0` turns this feature off.
- `TagReadOffset` sets a byte offset within the block or UID data to start reading from, using a value between 1 to 15. This setting is often used with `TagReadLength` to specify starting point of the desired data to be extracted and how many bytes to read from that starting point. If using `TagReadOffset` along with `TagReadLength`, the `TagReadOffset + TagReadLength` values must be less than or equal to 16, otherwise the length will be reduced by truncating data. Default `TagReadOffset=0` (no offset).

  Note: `TagReadOffset` and `TagReadLength` are not applicable when reading secure DESFire data where `DESFireReadLength` is used instead.

- `TagReadRightShift` will right shift decimal 64bit data. It can be used to remove any parity bits at the end of extracted binary data. Use a value from 1 to 63 to enable this setting, with a default is `TagReadRightShift=0`. `TagReadFormat` must be set to decimal (`=d`).

The following example shows how using `TagByteOrder` will change the UID card/tag data output from a VTAP reader over all interfaces.

Example: Reading UID from a MIFARE card/tag and sending reversed byte order over Wiegand and ASCII interfaces

```
!VTAPconfig

MIFAREClassic=U          ; Read UID data from MIFARE Classic card/tag
WiegandMode=1            ; Enable Wiegand interface
```

Output on ComPort or other ASCII interface:
Hex (4-byte): C699E142

Output on Wiegand interface:
Hex (4-byte): C699E142
Binary: 11000110 10011001 11100001 01000010

Now, add this extra line in `config.txt` to reverse the byte order:

```
TagByteOrder=1           ; Reverse the byte order of the tag data
```

Output for MIFARE card/tag on ComPort or other ASCII interface:
Hex (4-byte): 42E199C6

Output for MIFARE card/tag on Wiegand interface:
Hex (4-byte): 42E199C6
Binary: 01000010 11100001 10011001 11000110

If you only want to reverse the byte order on certain types of card/tag data, refer to the VTAP Commands Reference Guide for more about the alternative `TagByteOrderTypes`.

The following example shows how using the card/tag data format settings can progressively change the way MIFARE Classic block data is output from a VTAP reader over all interfaces.

Example: Read data from MIFARE Classic block and send data over Wiegand and ASCII interfaces

```
!VTAPconfig

WiegandMode=1
MIFAREClassic=B            ; Read block data from MIFAREClassic card/tag
TagReadBlockNum=3          ; Read data from block 3
TagReadKey=AC42E155C409    ; Key to read the block 3
TagReadOffset=12           ; Start reading data after byte 12 in the
                           ;  block data
TagReadLength=4            ; Read 4 bytes starting from 13th byte in the
                           ;  block TagReadOffset+TagReadLength=16 (12+4)
```

Outputs:

Hex (4-byte): 01378684

Binary: 00000000 10011011 11000011 01000010

Adding two further settings in `config.txt`:

```
TagReadRightShift=1         ; Right shift the data by 1 bit to remove
                            ;  parity bit
TagReadFormat=d             ; Convert extracted binary block data into
                            ;  decimal
```

Outputs:

Decimal: 10208066

Binary: 00000000 10011011 11000011 01000010

The right shift by 1 bit causes an extra bit (0) to be padded on the left, resulting in a whole byte of 0 on the left of the ‘payload of interest’. When outputting to ASCII in decimal, any leading zeros are not printed, however the Wiegand output is still 32 bits (including leading zeros).

And adding another two settings in `config.txt`:

```
TagReadMinDigits=9         ; Fix the number of digits to be read to 9.
TagWiegandBits=32          ; Bit length for Wiegand data from card/tags
```

Output for MIFARE card/tag on ComPort or other ASCII interface:

Decimal (3-byte): 010208066

Output for MIFARE card/tag on Wiegand interface:

Hex (4-byte): 009BC342

Binary: 00000000 10011011 11000011 01000010

The following example shows how using the card/tag data format settings can change the way secure DESFire data is output from a VTAP reader over all interfaces.

Example: Read from DESFire card or tag data containing ASCII data and send data over Wiegand and ASCII interfaces

```
!VTAPconfig

WiegandMode=1
NFCType4=D                     ; Read DESFire data from NFCType4 cards
DESFire1AppID=92E672           ; 24 bit DESFire application ID to read
                               ;  (6 hex digits)
DESFire1FileID=26              ; The file ID to read, within the
                               ;  application (decimal 1 to 255)
DESFire1KeyNum=1
DESFire1KeySlot=3              ; use the key saved in appkey3.txt
                               ;  on the VTAP reader
DESFire1Crypto=1               ; 3DES cryptography
DESFire1Format=0               ; No format, so must set DESFireReadLength
                               ;  and TagReadFormat
DESFire1ReadLength=8           ; Number of bytes to read
                               ;  when DESFire1Format=0
```

Outputs:
Hex (8-byte): 3531323334333336
Binary: 00110101 00110001 00110010 00110011 00110100 00110011 00110011 00110110

And adding another setting in `config.txt` to output ASCII:

```
TagReadFormat=a                ; Keep the ASCII data read from DESFire
                               ;  card/tag for ASCII interfaces. If not
                               ;  set, the data will be converted to hex
```

Outputs:
ASCII (8-characters) 51234336
Binary: 00110101 00110001 00110010 00110011 00110100 00110011 00110011 00110110

And adding further settings in `config.txt` to alter the Wiegand output only:

```
TagWiegandASCIIFormat=d        ; Interpret the ASCII data from
                               ;  TagReadFormat as decimal
TagWiegandBits=32              ; Set the Wiegand output bit length to 32
```

Output for DESFire card/tag on ComPort or other ASCII interface:
ASCII: 51234336
Output for DESFire card/tag on Wiegand interface:
Hex: 030DC620

2.3 Format pass and card/tag data to match over Wiegand

The following example shows how you might combine these settings to get matching data formats output from both mobile passes and card/tag UID data output from a VTAP reader over all interfaces.

Example: Reading a MIFARE Classic UID and the matching value from a mobile wallet pass payload to send over Wiegand and ASCII interfaces

```
!VTAPconfig

MIFAREClassic=U
WiegandMode=1
WiegandPassMode=1
WiegandPassSeparator=|
WiegandPassSection=2
PassWiegandBits=32          ; Output 32bit mobile wallet pass payload
                              over Wiegand
TagWiegandBits=0            ; Automatic bit length of tag data over Wiegand
```

Output for MIFARE card/tag on ComPort or other ASCII interface is in the default format, hexadecimal, since `TagReadFormat` is not set:
Hex (4-byte): A6D7D15D

Output for MIFARE card/tag on Wiegand interface:
Hex (4-byte): A6D7D15D
Binary: 10100110 11010111 11010001 01011101

Full mobile wallet pass payload: 3|09uwlh2jh|A6D7D15D|Testing

Output for mobile wallet pass on ComPort or other ASCII interface, after the `WiegandPass...` settings :
Hex (4-byte): A6D7D15D

Output for mobile wallet pass on Wiegand interface:
Hex (4-byte): A6D7D15D

The following example shows how using the card/tag data format settings can change the way secure DESFire data is output from a VTAP reader over all interfaces.

Example: Settings to read data from both DESFire cards/tags and one section of mobile pass payload containing ASCII data

Start with settings to read the DESFire card/tag data:

```
!VTAPconfig

WiegandMode=1
NFCType4=D                 ; Read DESFire data from NFCType4 cards
DESFire1AppID=92E672       ; 24 bit DESFire application ID to read
                           ;  (6 hex digits)
DESFire1FileID=26          ; The file ID to read, within the
                           ;  application (decimal 1 to 255)
DESFire1KeyNum=1
DESFire1KeySlot=3          ; Use key saved in appkey3.txt on the VTAP
DESFire1Crypto=1           ; 3DES cryptography
DESFire1Format=0           ; No format, so must set DESFireReadLength
                           ;  and TagReadFormat
```

Output on both Wiegand and ASCII interfaces:

Hex (3-byte) 353132 Binary: 00110101 00110001 00110010

Since `DESFire#Format` is not set, the default DESFire read length (3-byte) is read

Now add the line:

```
DESFire1ReadLength=8        ; Number (1 to 16) of bytes to read
                            ;  when DESFire1Format=0
```

Output on both Wiegand and ASCII interfaces:

Hex (8-byte) 3531323334333336

Binary: 00110101 00110001 00110010 00110011 00110100 00110011 00110011 00110110

Add the line:

```
TagReadFormat=a             ; Convert the data read from DESFire card/tag
                            ;  for ASCII interfaces
```

Output on ASCII interface: ASCII (8-character) 51234336

Binary: 00110101 00110001 00110010 00110011 00110100 00110011 00110011 00110110

Output on Wiegand: Hex (8-byte) 3531323334333336

Binary: 00110101 00110001 00110010 00110011 00110100 00110011 00110011 00110110

Continued overleaf ->

-> Continued from previous page

Add the line:

```
TagWiegandASCIIFormat=d      ; Interpret the ASCII data from
                             ;  TagReadFormat as decimal
```

Output on Wiegand:
Binary: 00000000 00000000 00000000 00000000 00000011 00001101 11000110 00100000
Hex: 00000000030DC620
Decimal: 51234336

Add the line:

```
TagWiegandBits=32           ; Set the Wiegand output bit length to 32
```

Final output for DESFire card/tag
ASCII interface (such as output over ComPort): 51234336
Wiegand interface:
Binary: 00000011 00001101 11000110 00100000
Hex: 030DC620
Decimal: 51234336

Now consider the mobile pass.
The full mobile wallet pass payload is: 3|653POAwbeS2|51234336|Testing
Here, we assume that the third section of pass payload contains the required data.

```
WiegandPassMode=1          ; Enable Wiegand Pass mode to extract only
                           ;  part of the entire pass payload
WiegandPassSeparator=|     ; Set the separator character VTAP will use
                           ;  to separate the pass payload sections
WiegandPassSection=2       ; Use third section (starting from 0)
PassFormat=d               ; Interpret the pass payload as a decimal
                           ;  number for conversion to a Wiegand
                           ;  bit sequence
PassWiegandBits=32
```

Final output for mobile wallet pass
ASCII interface, such as ComPort: 51234336
Wiegand interface:
Binary: 00000011 00001101 11000110 00100000
Hex: 030DC620
Decimal: 51234336

Note: When using the Wiegand interface, multiple reads are not supported. In this case, only the lowest numbered `DESFire#...` settings will be used, which might not be `DESFire1...`. If only `DESFire3...` and `DESFire4...` settings are defined in `config.txt`, the `DESFire3...` settings would then be used for output over Wiegand.

If you are getting different card/tag and pass outputs over Wiegand, always bear in mind that there are different settings in VTAP reader configuration to control the format of :

- pass data over serial interfaces,
- pass data over the Wiegand interface,
- card/tag data over serial interfaces and
- card/tag data over the Wiegand interface.

In some applications these may need to be different. Do not assume that the data format you receive over a serial interface is necessarily the same as that being sent over Wiegand. Check the data received by your access control system to see what is being sent over Wiegand. Please do contact **vtap-support@dotorigin.com** if you need further help adjusting your configuration to achieve a particular format of output.

2.4 Wiegand LED and beep signals sent to the VTAP reader

By default the VTAP reader will receive and act on any LED (red and green) or beep response signals from an access controller connected by a Wiegand interface.

Note: The Wiegand interface is only available if you are using the VTAP100-PAC-W model.

To disable control of the VTAP LEDs or buzzer input, you can either disconnect the wires which connect the buzzer and LED pins from the controller to the VTAP reader, or set `WiegandInputEnable=0`. Set `=01` to enable LED input only, or `=80` to enable buzzer input only. The default is `WiegandInputEnable=81` which enables LED and buzzer.

3 About Application Notes

Application Notes address topics of interest to small groups of customers, or topics around the use of a VTAP reader with third-party systems.

The main documents available to support your use of the VTAP100-PAC-W are the Installation Guide for your VTAP reader model and the VTAP Configuration Guide. You will find the latest versions of these, and other useful information at **https://vtapnfc.com**.

If you need further help do contact us by email to **vtap-support@dotorigin.com**, or by phone +44 (0) 1428 685861 from Europe and Asia, or +1 (562) 262-9642 from Northern and Latin America.