Technical Reference Guide
For Gemini 2000 Payment System
version 1.18

Address: Unit 7 Vitrage Technical Park, 27 Witney Road,
Poole, Dorset, BH17 0GL, United Kingdom
Phones: +44 1202 666 700
Email: info@gemini2k.com

Revision History

Version	Date	Author	Description
1.0	26 July 2021	PA	Initial
1.1	26 Jan 2022	JM	Formatting, wording
1.2	28 Jan 2022	PA	Remove logins, update serial pins description
1.3	16 Feb 2022	PA	Added section Development-specific requirements
1.4	23 Mar 2022	PA	Commands for the graphical display.
1.5	20 Jun 2022	PA	DNS and NTP requirements
1.6	14 Sep 2022	PA	Expand on section 3.4. Add modem configuration
1.7	20 Oct 2022	PA	Clarify response codes for EMVSALE. IP address no longer required. Update Notification Codes
1.8	18 Nov 2022	PA	Terminal sends heartbeat to BO even if reader disconnected
1.9	13 Jan 2023	PA	Update metric value for modem usage
1.10	14/03/2023	SU	Added token authorization related parameters
1.11	14 Apr 2023	PA	Clarify use of CANCEL and EMVCONFIRM
1.12	16 May 2023	TA	Added direct_display_command to Table 2
1.13	26 May 2023	PA	Examples for direct_display_command
1.14	12 Jun 2023	PA	Clarify A1/C1 notification codes and cancelling EMVSALE
1.15	29 June 2023	PA	Use of systemd networking. Reformatting.
1.16	12 Sep 2023	PA	Handling loss of BO communication
1.17	27 Nov 2023	SU	Added token_local parameter
1.18	28 Feb 2024	JM	Updated photos

Contents

Introduction

This document describes how to connect an EPOS to the Gemini 2000 terminal and the use of the messaging protocol between them.

Audience

Gemini 2000 customers who want to use our Telemetry terminal and contactless reader to integrate a contactless payment facility into their EPOS devices.

Prerequisites

An understanding of Linux, networking, Mosquitto MQTT.

List of Terms

Term used	Meaning
Telemetry terminal	A generic term for one of our single-board computer models. This runs Linux and the Terminal software.
Reader	A generic term for one of our card reader models. This connects to the Telemetry terminal.
Terminal	Collective term for Telemetry terminal connected to Reader
Terminal software	Python scripts and proprietary binaries running on the Telemetry terminal
EPOS	Electronic Point-of-sale device, such as vending machine or EV charger
PSP	Payment Service Provider, a banking intermediary that clears transactions
SCA	Strong Customer Authentication, typically a request for a secondary authentication on top of the card tap
APN	Access Point Name for modem use. Supplied by the simcard provider.

1. Basic operation overview

The EPOS is connected to the Telemetry terminal, which is in turn connected to the Reader.

To perform payments processing, the Telemetry terminal must have an Internet connection, which can be done via an Ethernet cable or the optional WiFi module or 4G modem.

1. The EPOS initiates operations by sending commands to the Telemetry terminal.
2. The Telemetry terminal in turn sends commands to the Reader.
3. The Reader reads payment card information, encrypts it, and sends the encrypted data back to the Telemetry terminal.
4. The Telemetry terminal forwards the encrypted data to the PSP via the internet for payment processing.
5. On receiving a reply from the PSP, the Telemetry terminal responds accordingly to the EPOS.

For readability this document will from now onwards refer to the Telemetry terminal connected to a Reader as the ‘Terminal’.

2. The communication flow

Communication between the Telemetry terminal and Reader is via a dedicated serial connection. The Reader on its own does not initiate any communications, it only responds to commands sent by the Telemetry terminal, which in turn responds to commands sent by the EPOS.

Communication between the EPOS and Terminal is carried out by MQTT messages over a TCP/IP connection. The central point in the EPOS/Terminal data exchange is played by the MQTT broker running as a server on the Terminal.

Both the EPOS and the Terminal software are ‘clients’ of the MQTT broker, publishing and subscribing to specific topics.

The Terminal implements an API to enable the exchange of messages with the EPOS. The API is based on asynchronous message exchange between the EPOS and the Terminal.

Both the EPOS and Terminal can send and receive messages at any time. There is no hierarchy like client/server. However, the Terminal is most likely to receive commands from the EPOS and subsequently send responses back to it. The Terminal can also send notifications to the EPOS about events that might be of interest to it, for example, when connectivity with the back office has been lost or restored.

If the Terminal is configured to use the back office, then it will require internet access. All card payment requests are temporarily stored on the Terminal until their receipt has been acknowledged by the back office. Therefore, if the EPOS makes any payment requests when there is no back office connectivity, these will be stored and later sent to the back office once connectivity resumes.

See *Figure 1 Communication flow*

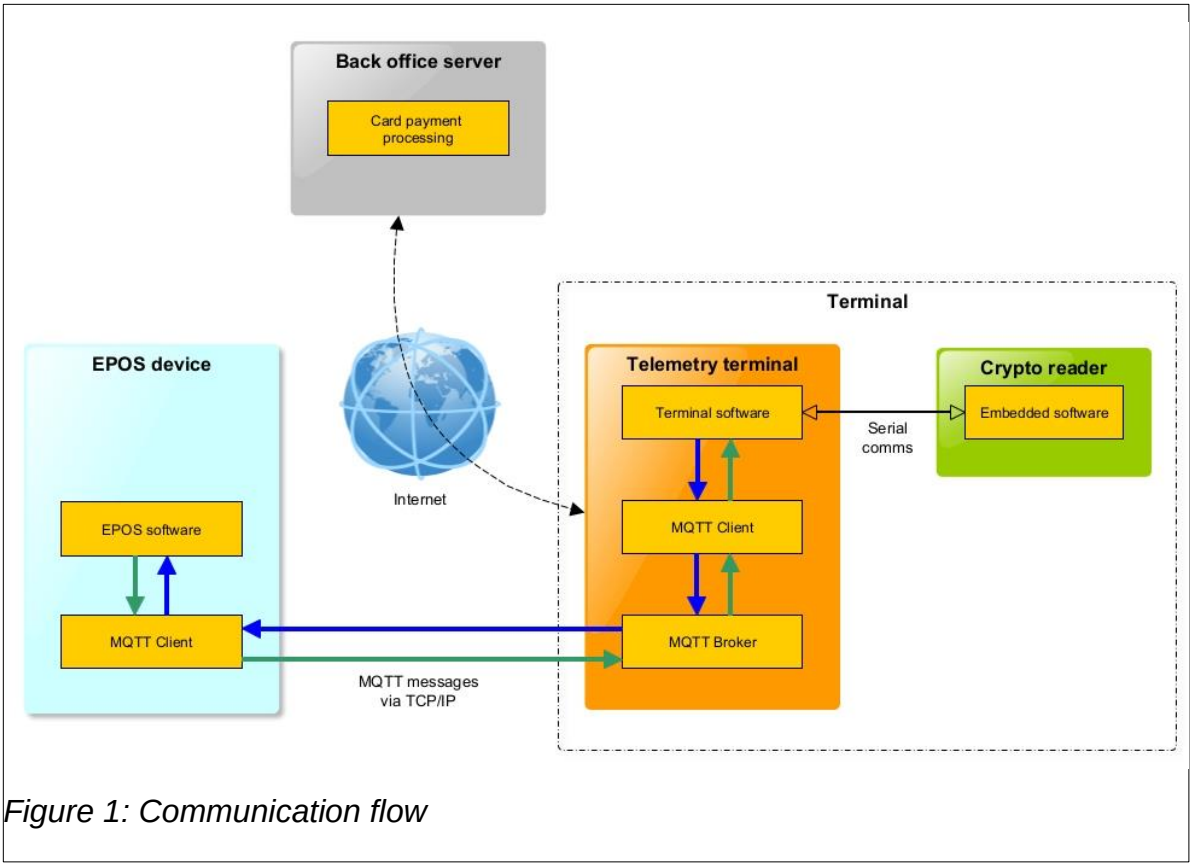

Figure 1: Communication flow

3. The Testing and Production environments

Gemini 2000 provides a testing environment to enable the simulation of card payments. We will supply you with test payment cards with which to carry out these transactions.

The Telemetry and Reader are both configured to operate in either the test or production environments. The Telemetry automatically connects to the appropriate environment. Although the Reader can be swapped to any other Telemetry at any point, it will not work if swapped across the test and production environments.

4. The Terminal Hardware

4.1 Connectivity options

The telemetry terminal offers the following connectivity:

- Ethernet
- WiFi module (optional)
- 4G modem (optional)
  - micro-SIM holder present
- RJ10 port
  - Not required for general operation and is currently unused, but is available for customisation, e.g. it could be brought out to pins that get set under certain conditions, such as a successful payment transaction.
- RJ12 port
  - Reserved solely for connecting to the Reader.
  - A reader is not tied to a specific Terminal and can be swapped at any point.
- Pins for serial communication
  - Only used for serial connections to a PC via a suitable cable and terminal client software e.g. an FTDI cable and PuTTY client.
  - Remove the top cover to locate P8, a 3 pin male header inside the unit
  - The serial cable must be connected with GND (black lead) nearest the outside edge of the PCB (**Error! Reference source not found.**)

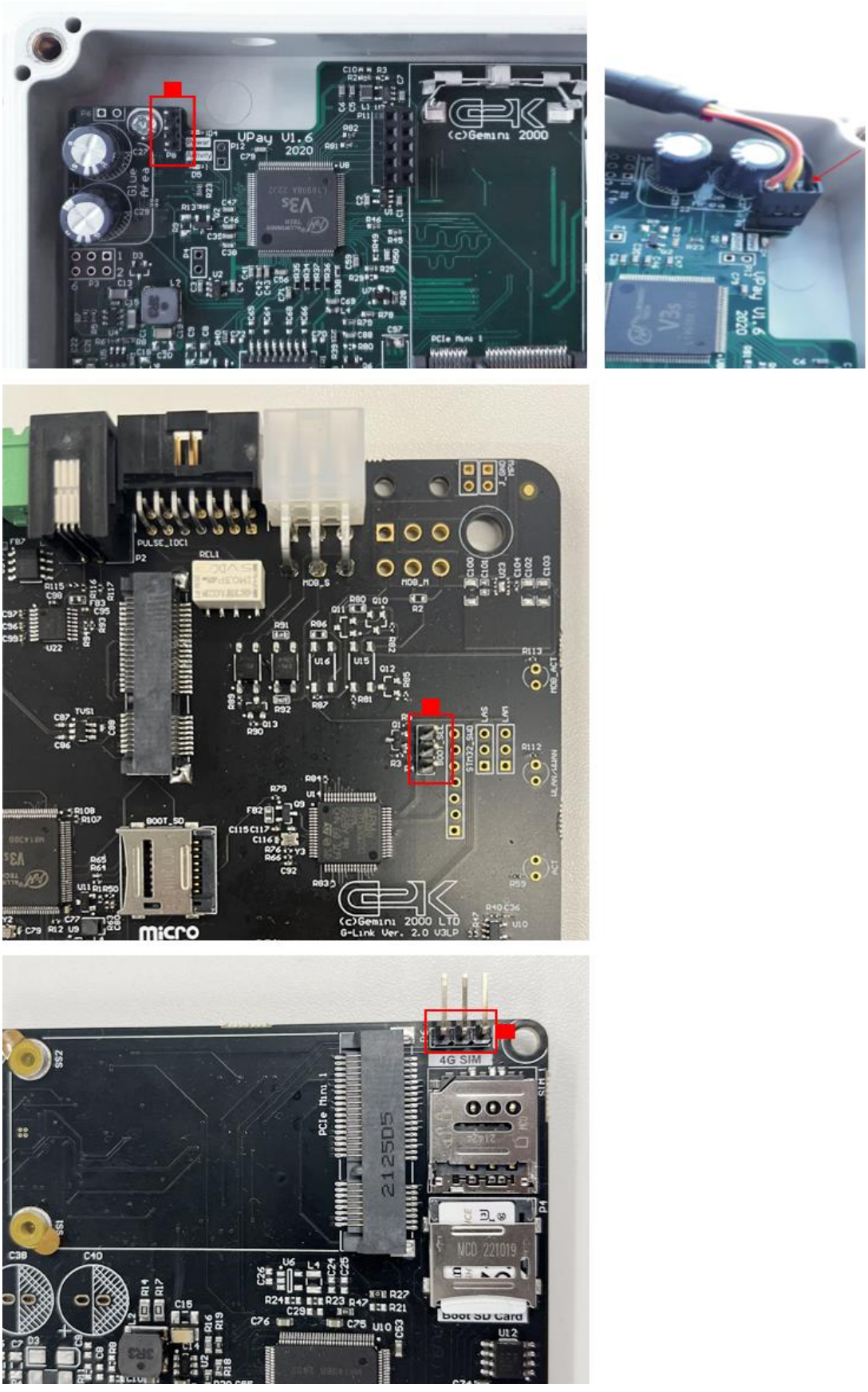

Figure 2: Pins for serial communication on (top to bottom) VPay, GLink and VLink telemetry boards. The red notch indicates location for black wire of console cable for correct polarity.

4.2 The microSD

Contains the operating system and software. In order to access the Gemini 2000 back office, the Terminal is assigned a digital certificate which is linked to it's unique hardware ID. Access to the back office will not be possible if SD cards are swapped between Terminals.

5. Options for connecting the Terminal to the EPOS

As the Terminal requires an Internet connection, and also a TCP/IP connection to the EPOS, there are several connection options available. There is no DHCP server on the Terminal.

5.1 Option 1: Use a network switch/router for internet access

- Requires a network switch/router to be installed at the EPOS
- The switch/router provides access to the internet and connects the EPOS and Terminal

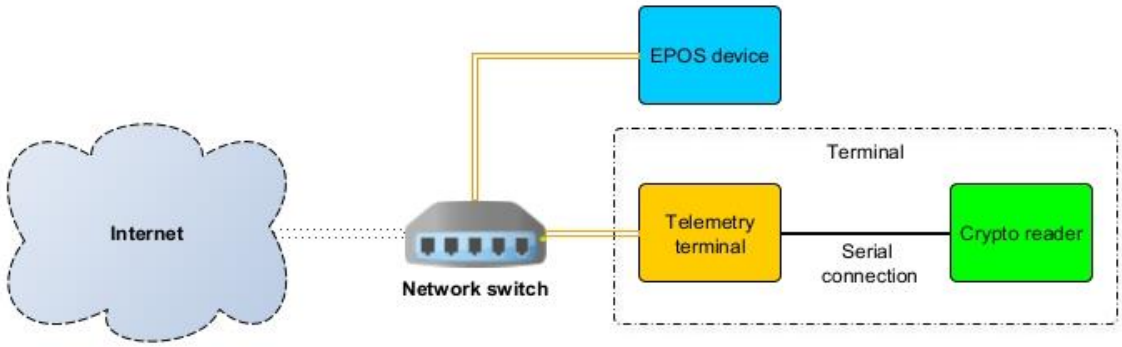

Figure 3: Using a network switch

5.2 Option 2: use WiFi for internet access, and ethernet to link Terminal and EPOS

- WiFi access point provides access to the internet
- WiFi module installed in the Terminal provides access to the WiFi access point
- EPOS connects to Terminal via direct Ethernet cable connection.
- EPOS and Terminal network interfaces to be set up for direct cable connection
- Alternatively, if the EPOS has Wifi capability, it can connect to the Terminal via the WiFi access point, instead of via ethernet

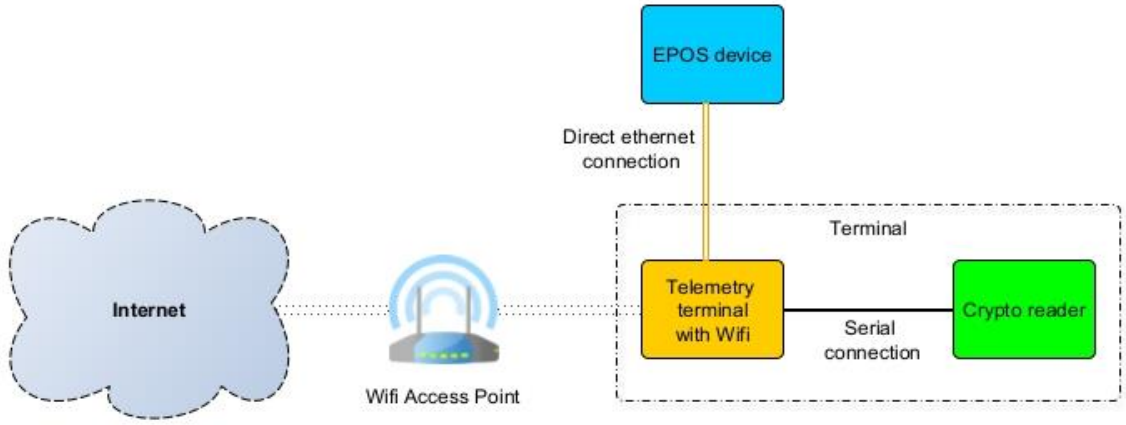

Figure 4: Using the WiFi module

5.3 Option 3: use the modem for internet access, and ethernet to link Terminal and EPOS

- Modem module installed in the Terminal provides access to the internet
- EPOS connects to Terminal via direct Ethernet cable connection.
- EPOS and Terminal network interfaces to be set up for direct cable connection

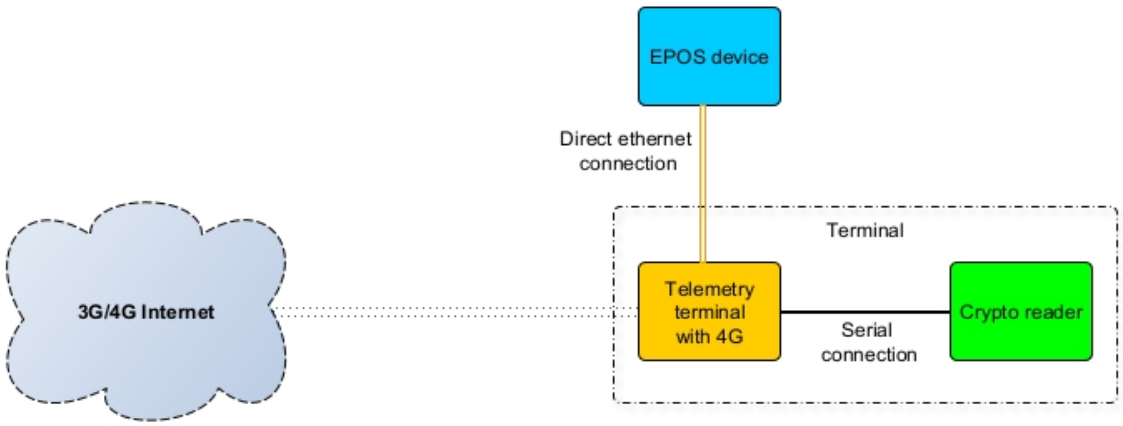

Figure 5: Using the 4G module + direct ethernet

5.4 Option 4: use the modem for internet access, and wifi to link Terminal and EPOS

- Modem module installed in the Terminal provides access to the internet
- WiFi module installed in the Terminal provides access to the wireless network
- EPOS connects to the Terminal via the wireless network

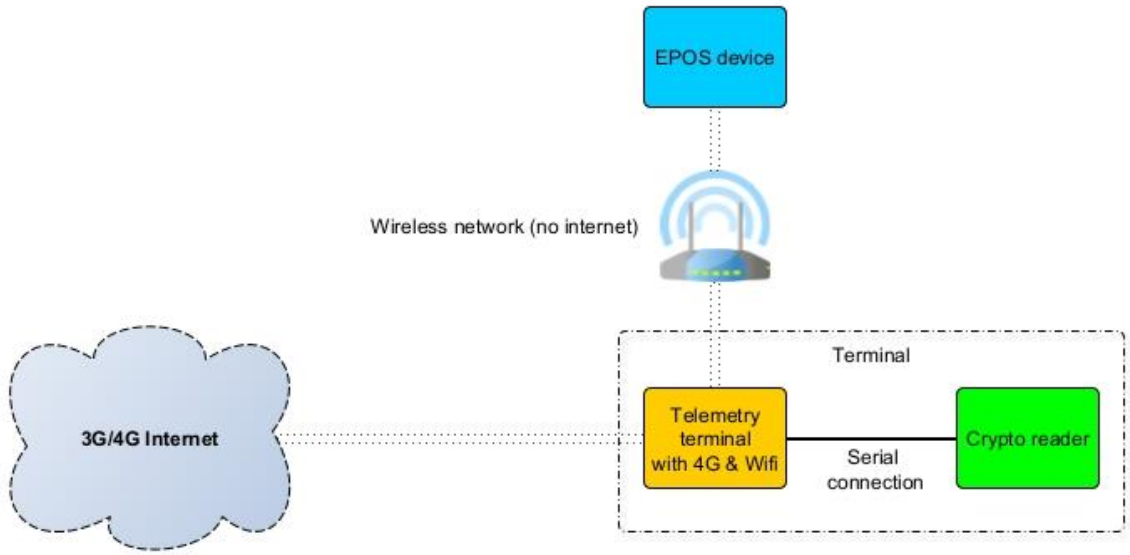

Figure 6: Using the 4G and WiFi modules together

6. Network configuration

The Terminal's internet connection must have:

1) access to a public DNS
2) access to an NTP server

The following configuration files are used. Only the ethernet and/or wifi interfaces will require configuring. The modem interfaces can be left as is. Editing them requires root privileges.

Configuration file	Description
/etc/systemd/network/eth0.network	Ethernet
/etc/systemd/network/wlan0.network	WiFi
/etc/wpa_supplicant/wpa_supplicant-wlan0.conf	WiFi credentials
/etc/systemd/network/eth1.network	modem (used by Huawei)
/etc/systemd/network/usb0.network	modem (used by Quectel)
/etc/systemd/network/wwan0.network	modem (used by Telit)
/opt/4G_modems/apn.cfg	APN for modem

- By default, the Terminal is shipped with it's ethernet and wifi interfaces set to **DHCP.**
- The Terminal runs an MQTT broker, so must be configured with a **static IP address.**
- There must not be more than one default gateway IP address specified.

Access to the Terminal's command line can done via serial cable connection, or SSH if the IP address is known. Please contact us for the username and password.

The detailed configuration for each connection option is described below. Change the IP addresses to suit your network.

Reboot the device after making any changes : **sudo reboot**

6.1 Option 1: Use a network switch/router for internet access

1) Configure **/etc/systemd/network/eth0.network** to have a static IP eg.

```
[Match]
Name=eth0

[Network]
Address=192.168.7.247/24
Gateway=192.168.7.1
DNS=192.168.7.1
```

2) Connect the Terminal's ethernet to a network switch that has internet access
3) Connect the EPOS to the network switch

6.2 Option 2: use WiFi for internet access, and ethernet to link Terminal and EPOS

1) Configure **/etc/systemd/network/wlan0.network** to have a static IP eg.

```
[Match]
Name=wlan0

[Network]
Address=192.168.7.247/24
Gateway=192.168.7.1
DNS=192.168.7.1
```

2) Supply wifi credentials in **/etc/wpa_supplicant/wpa_supplicant-wlan0.conf**

```
ctrl_interface=/var/run/wpa_supplicant
ap_scan=1
fast_reauth=1

network={
    ssid="your-ssid"
    psk="your-password"
    priority=1
}
```

3) Connect the EPOS and Terminal directly together with ethernet. The EPOS will also require a static IP that uses the same subnet as the Terminal eg. if Terminal is 192.168.7.247, then EPOS must be 192.168.7.nnn

6.3 Option 3: use the modem for internet access, and ethernet to link Terminal and EPOS

1) Configure **/etc/systemd/network/eth0.network** file to have a static IP, with no gateway IP eg.

```
[Match]
Name=eth0

[Network]
Address=192.168.7.247/24
```

2) Ensure that no gateway IP is specified in any other configuration file
3) Specify the modem's APN in **/opt/4G_modems/apn.cfg**
4) Connect the EPOS and Terminal directly together with ethernet. The EPOS will also require a static IP that uses the same subnet as the Terminal eg. if Terminal is 192.168.7.247, then EPOS must be 192.168.7.nnn

6.4 Option 4: use the modem for internet access, and wifi to link Terminal and EPOS

1) Configure **/etc/systemd/network/wlan0.network** file to have a static IP, with no gateway IP eg.

```
[Match]
Name=wlan0

[Network]
Address=192.168.7.247/24
```

2) Supply wifi credentials in **/etc/wpa_supplicant/wpa_supplicant-wlan0.conf**

```
ctrl_interface=/var/run/wpa_supplicant
ap_scan=1
fast_reauth=1

network={
    ssid="your-ssid"
    psk="your-password"
    priority=1
}
```

3) Ensure that no gateway IP is specified in any other configuration file
4) Specify the modem's APN in **/opt/4G_modems/apn.cfg**
5) Configure the EPOS to access the Terminal via the wireless network

7. The MQTT broker

The MQTT broker installed on the Terminal is Mosquitto. It runs as a Linux service **(mosquitto.service)** and automatically starts at boot.

Please contact us for the username and password that MQTT clients must use.

The address and port for MQTT clients to access the broker is :

IP Address	**:**	**(the static IP of the Terminal)**
Port	**:**	**1883**

The broker configuration file is located on the Terminal at:

/etc/mosquitto/mosquitto.conf

8. The MQTT client

In order to publish/subscribe to the broker, an MQTT client library must be used. The library must be supplied with the login credentials to make a connection to the broker.

There are libraries available for a wide range of programming languages. The Terminal software makes use of the Paho Python client library. Similarly, the EPOS device will also need to use a client library, the choice of which depends on the programming language being used by the EPOS developer.

See https://mqtt.org/software/ for a list of client libraries.

To simulate MQTT client connections from a PC to the Terminal, you can use tools such as MQTTX, MQTT Explorer, etc to publish/subscribe to the MQTT broker.

9. How MQTT is used

The Terminal and EPOS exchange messages by publishing and subscribing to specific topics on the broker. A heartbeat message gets published by the Terminal at a specific interval and the EPOS can use this to detect if it has lost connection to the Terminal.

The Terminal :

- subscribes to **EPOS/requests**
- publishes to **G2KT/responses** and **G2KT/notifications**
- publishes heartbeat messages to **G2KT/HB**

The EPOS :

- subscribes to **G2KT/responses** and **G2KT/notifications**
- subscribes to **G2KT/HB** to receive heartbeat messages
- publishes to **EPOS/requests**

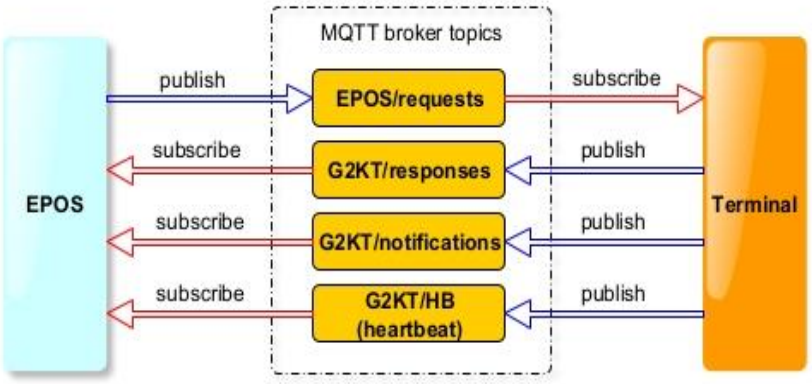

Figure 7: MQTT topics

10. The message structure

All messages are strictly JSON formatted strings in UTF-8 encoding, following a common structure definition:

{

"**version**":<message version string>,

"**type**": <message type>,

"**payload**": <message payload>

}

There are 3 message types, each containing different payload values:

- **Request**
- **Response**
- **Notification**

An example message flow between the Terminal and EPOS to illustrate the use of the 3 message types:

1. EPOS posts EMV Authorisation **request** message
   - Terminal receives the **request**
2. Terminal posts PRESENT CARD **notification** message, and turns on the reader
   - EPOS receives the **notification**
3. If card is read OK,
   1. Terminal posts CARD READ OK **notification** message
      - EPOS receives the **notification**
   2. Terminal posts CONNECTING **notification** message
      - EPOS receives the **notification**
4. Terminal sends EMV Authorisation to the BO, then on receiving the result, it posts EMV Result **response** message, with appropriate status code
   - EPOS receives the **response**
5. At a pre-defined interval, Terminal also posts heartbeat **notification** message
   - EPOS receives the **notification**

10.1 The Request message

A **request** message is published by the EPOS in the **EPOS/requests** topic. On receiving the message, the Terminal validates it before publishing a response (where required) in the **G2KT/responses** topic. If a message fails validation, no response will be produced and an entry will be logged on the Terminal at **~/G2K_Terminal/terminal.log.**

For the **request** message, **type** must be set to **"request":**

{

"version": <*message version*>,

"type": "**request**",

"**payload**": {

"version": <*payload version*>,

"code": <*request code*>,

"**parameters**": {<*parameter key*> *:* <*parameter value*>}

}

}

The Terminal will validate the message as follows :

- <*message version*> and <*payload version*> are currently unused
- <*request code*> and <*parameter key*> must be from Table 1 Request codes
- <*parameter value*> type must match the Data type in Table 1 Request codes
- <*parameter key*> must be supplied if Mandatory in Table 1 Request codes

Table 1: Request codes

Request code	Parameter key	Data type	Description	Mandatory
EMVAUTH (EMV Authorisation)	cust_ref	String	Customer ref. To appear on the bank statement	N
	tr_ref	String	Transaction ref. Presumably a counter in the EPOS + some time base	Y
	amount	Integer	Minor currency units with 2 decimals implied e.g. 1000 = 10.00	Y
	cashback	Integer	Minor currency units with 2 decimals implied e.g. 1000 = 10.00	N
	currency	Integer	3 digit currency code e.g. 826 = GBP	Y
	ack	Integer	If present and not zero(0) then require immediate acknowledgment of the request	N
	token_enable	Integer	0: token authorization disabled 1: token authorization enabled When enabled the terminal will run the 'magic loop' first	N
	scheme_code	String	<parameter value> is a value that is specific to the payment scheme used by the EPOS vendor. It activates the reader and returns the private scheme card UID. Default scheme reports ISO14443 Type A card UID LSB first (UIDLSB)	N
	token_local	Integer	0: card data is reported to BO in TOKEN request; 1: card data is only returned to the EPOS in PRIVATESCH response	N

			(default)	
EMVCONFIRM (EMV confirmation)	tr_ref	String	Transaction ref. Same value as in the initial EMVAUTH request	Y
	amount	Integer	Minor currency units with 2 decimals implied e.g. 1000 = 10.00	Y
	void_reason	Integer	Void reason code	N
CANCEL	n/a	n/a	After EMVAUTH or EMVSALE, the reader is switched on and waits for a card to be presented until the timeout period is reached (30s). CANCEL is used to turn off the reader instead of waiting for the timeout.	n/a
TERMSTATUS (Terminal status)	reports_list	List of strings	<parameter value> can be one or more of: 'swver' - Terminal s/w version 'uid' - get Reader ID 'tid' - get Terminal ID	Y
PRIVATESCH (Private payment scheme)	scheme_code	String	<parameter value> is a value that is specific to the payment scheme used by the EPOS vendor. It activates the reader and returns the private scheme card UID.	Y
EMVSALE (when performing an Offline Authorisation, or an Online Authorise-and-Confirm)	cust_ref	String	Customer ref. To appear on the bank statement	N
	tr_ref	String	Transaction ref. Presumably a counter in the EPOS + some time base	Y
	amount	Integer	Minor currency units with 2 decimals implied e.g. 1000 = 10.00	Y
	cashback	Integer	Minor currency units with 2 decimals implied e.g. 1000 = 10.00	N

	currency	Integer	3 digit currency code e.g. 826 = GBP	Y
	token_enable	Integer	0: token authorization disabled 1: token authorization enabled When enabled the terminal will run the 'magic loop' first	N
	scheme_code	String	<parameter value> is a value that is specific to the payment scheme used by the EPOS vendor. It activates the reader and returns the private scheme card UID. Default scheme reports ISO14443 Type A card UID LSB first (UIDLSB)	N
	token_local	Integer	0: card data is reported to BO in TOKEN request; 1: card data is only returned to the EPOS in PRIVATESCH response (default)	N
RESET	n/a	n/a	Reboots the Terminal	n/a
CUSTID	n/a	n/a	Request customer identification by presenting an EMV card or private scheme card.	n/a
DISPLAY	command	String	<parameter value> is from the 'Command' column in Table 2 Graphical display codes	Y
	args	Dictionary	<parameter value> is from the 'Arguments' column in Table 2 Graphical display codes It is a dictionary with named arguments specific for each command. When no arguments are required, then supply an empty	Y

			dictionary as {}	

Table 2: Graphical display codes

NOTE: When using the Yuzu reader, only use the 'direct_display_command'. It is used to send commands described in the Yuzu Display Guide.

Command (string)	Arguments	Description
clear	-	Clears entire screen
set_cursor_position	x:Integer y:Integer	Sets current cursor position to horizontal(x) and vertical(y) coordinates. For uCrypto display: 0<=x<=127, 0<=y<=31
print	x:Integer y:Integer text:String	Prints the 'text' at x,y coordinates using the currently selected font
set_font	index:Integer	Font index. TIMESNR_8=0 CENTURY_8=1 ARIAL_8=2 COMICS_8=3 GLCDFONT=4 TEST=5 ARIAL_18PT=6 NI7SEG_24PT=7 MS_SANS_SERIF_8PT_8859_15=8 ARIAL_8PT_8859_15=9 ARIAL_10PT_8859_15=10 ARIAL_20PT_8859_15=11 TIMES_ROMAN_8PT_8859_15=12 TIMES_ROMAN_10PT_8859_15=13 TIMES_ROMAN_20PT_8859_15=14
invert	-	Inverts display
draw_line	x0:Integer y0:Integer	Draws a line between points with coordinates x0,y0 and x1,y1 with given color. Color is

	x1:Integer	optional and can be 0 or 1. Default is 1.
	y1:Integer	
	color:Integer	
idle	mode:Integer	Activate screen saver in given mode, e.g. horizontal scroll. Mode: 0 – exit idle mode – restore display content 1 – horizontal scroll - left 2 – horizontal scroll – right
clear_screen_area	x:Integer	Clears rectangular area of the screen with top left corner at x,y and given width(w) and height(h).
	y:Integer	
	w:Integer	
	h:Integer	
direct_display_command	args:List	Enables executing direct display commands.

(10.1.1) EMVAUTH vs EMVSALE

EMVAUTH:

- For situations where the final payment amount is not known at the time the customer presents their payment card.
- This is a pre-authorization request that must be followed by EMVCONFIRM (using the same **tr_ref**) in order to finalise the payment amount.
- The EMVCONFIRM amount has to be less than or equal to the EMVAUTH amount. You may send a zero amount to cancel/stop the payment from being taken.
- If no EMVCONFIRM is sent, the EMVAUTH transaction is cancelled by the PSP after a period of time, typically 5 days.
- If **token_enable** is set to 1 then the terminal will run 'magic loop' thus allowing acceptance of a non-EMV ISO14443 Type A card as a 'token'. If a non-EMV card is detected then the terminal will retrieve the card UID and will send a request for token authorization to the BO. In this case the **amount** is ignored**.**

EMVSALE:

- For situations where the final payment amount is known at the time the customer presents their payment card.
- No EMVCONFIRM is required as the request contains the final payment amount.
- Unlike EMVAUTH, attempting to send EMVCONFIRM afterwards with a zero amount will not cancel/stop the payment from being taken, under normal circumstances. The exceptional circumstance is discussed in section **Error! Reference source not found.Error! Reference source not found.**
- If **token_enable** is set to 1 then the terminal will run 'magic loop' thus allowing acceptance of a non-EMV ISO14443 Type A card as a 'token'. If a non-EMV card is detected then the terminal will retrieve the card UID and will send a request for token authorization to the BO. In this case the **amount** is ignored**.**

(10.1.2) Supported scheme codes for PRIVATESCH and TOKEN

- **UIDMSB**– ISO 14443 Type A card UID **MSB** first
- **UIDLSB** - ISO 14443 Type A card UID **LSB** first

10.2 Example JSON of Request messages

(10.2.1) Send an EMVAUTH

```
{"version": [1, 0],
"type": "request",
"payload": {
"version": [1, 0],
"code": "EMVAUTH",
"parameters": {
"cust_ref": "A00987B",
"tr_ref": "100020210611",
"amount": 10,
"cashback": 0,
"currency": 826,
"ack": 1}
}}
```

(10.2.2) Clear the display (uCrypto only)

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "clear",
"args": {}
}}}
```

(10.2.3) Print text on the display (uCrypto only)

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "print",
"args": {"x":5,"y":10,"text":"Present card"}
}}}
```

(10.2.4) Using User Mode (Yuzu only):

In this mode, user-defined graphics can be displayed on the Yuzu.

NOTE: screen co-ordinates are defined as X,Y, where X is specified as a 1-byte value and Y is specified as a 2-byte value in the format LSB, MSB. For example, location (55,270) would be specified in the command as (55,14,1). The Y value 270 = 0x10E = 14,1

(10.2.4.1) First set the display into User mode

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "direct_display_command",
"args": {"args": [128, 0, 0]}
}}}
```

(10.2.4.2) Clear display (optional, as previous step will clear it)

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "direct_display_command",
"args": {"args": [6]}
}}}
```

(10.2.4.3) Set 28p font

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "direct_display_command",
"args": {"args": [24,0]}
}}}
```

(10.2.4.4) Print 'TESTS' at x/y (0,0)

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "direct_display_command",
"args": {"args": [23, 0, 0, 0, 84, 69, 83, 84, 83, 0]}
}}}
```

(10.2.4.5) Set to 36p font

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "direct_display_command",
"args": {"args": [24,17]}
}}}
```

(10.2.4.6) Print 'TESTS' at x/y (55,270)

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "direct_display_command",
"args": {"args": [23, 55, 14, 1, 84, 69, 83, 84, 83, 0]}
}}}
```

(10.2.4.7) Set colour to blue

NOTE: Colour is defined as red, green, blue with range 0-255 each

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "direct_display_command",
"args": {"args": [22, 0, 0, 255]}
}}}
```

(10.2.4.8) Draw line from x/y(239,0) to x/y(50,270)

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "direct_display_command",
"args": {"args": [9, 239, 0,0, 50, 14,1]}
}}}
```

(10.2.4.9) Draw bitmap

See the Yuzu Display Guide for the bitmap index to use for a particular logo eg. the following draws the G2K logo at x/y(0,0).

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "direct_display_command",
"args": {"args": [12, 0, 0, 0,0]}
}}}
```

(10.2.4.10) Draw rectangle around boundary, from (0,0) to (239,139)

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "direct_display_command",
"args": {"args": [14, 0, 0,0, 239, 63,1]}
}}}
```

(10.2.4.11) Draw circle at (120,160) with radius 50

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "direct_display_command",
"args": {"args": [13, 120, 160,0, 50]}
}}}
```

(10.2.5) Using Payment Mode (Yuzu only):

In this mode, pre-defined graphics can be displayed, along with user-defined text.

Payment Mode is the default startup mode. Once activated, the Yuzu can be switched into several pre-defined states, each with it's own logo and optional text fields.

(10.2.5.1) Switch into Payment mode

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "direct_display_command",
"args": {"args": [128,0,1]}
}}}
```

(10.2.5.2) Set Idle state, showing no text at the top or bottom

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "direct_display_command",
"args": {"args": [128,1]}
}}}
```

(10.2.5.3) Set Idle state, showing no text at the top and 'TESTS' at the bottom

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "direct_display_command",
"args": {"args": [128,1, 84, 69, 83, 84, 83, 0]}
}}}
```

(10.2.5.4) Set Tap To Pay state, showing default 'Tap To Pay' at the top and £1.23 at the bottom

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "direct_display_command",
"args": {"args": [128,2, 163,49,46,50,51,0]}
}}}
```

(10.2.5.5) Set Authorise state, showing no text at the top and £1.23 at the bottom

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "direct_display_command",
"args": {"args": [128,3, 163,49,46,50,51,0]}
}}}
```

(10.2.5.6) Set Success state, with default text at top and bottom

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "direct_display_command",
"args": {"args": [128,4]}
}}}
```

(10.2.5.7) Set Declined state, with default text at top and bottom

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "direct_display_command",
"args": {"args": [128,5]}
}}}
```

(10.2.5.8) Set Out of Service state, with default text at top and bottom

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "direct_display_command",
"args": {"args": [128,6]}
}}}
```

(10.2.5.9) Set Tap To Pay state, showing 10 words from 'AAA' to 'JJJ' at the top and £1.23 at the bottom

Note: This demonstrates how to set custom top line text. It can be applied to any of the above states. The number 100 before each word indicates 1000ms display duration. The number 60 indicates 600ms pause between words (it is only specified once). These timings may be changed to suit. Every word is terminated with 0.

```
{"version": [1,0],
"type": "request",
"payload": {
"version": [1,0],
"code": "DISPLAY",
"parameters": {
"command": "direct_display_command",
"args": {"args": [128,2,163,49,46,50,51,0, 100,60, 65,65,65,0,
100,66,66,66,0, 100,67,67,67,0, 100,68,68,68,0, 100,69,69,69,0,
100,70,70,70,0, 100,71,71,71,0, 100,72,72,72,0, 100,73,73,73,0,
100,74,74,74,0]}
}}}
```

10.3 The Response message

A **response** message is published by the Terminal in the **G2KT/responses** topic, in reply to a request message sent earlier by the EPOS. The Terminal builds and validates it before publishing it in the **G2KT/responses** topic. If a message fails validation, no response will be produced and an entry will be logged on the Terminal at **~/G2K_Terminal/terminal.log.**

For the **response** message, **type** will be set to **"response"**.

{

"version":<*message version*>,

"type": "**response**",

"**payload**": {

"version": <*payload version*>,

"code": <*response code*>,

"**value**": {<*response key*> *:* <*response value*>}

}

}

For the message structure:

- <*message version*> and <*payload version*> are currently unused
- <*response code*> and <*response key*> will be from Table 3 Response codes
- <*response value*> will match the Data type in Table 3 Response codes
- <*response key*> will be supplied if Mandatory in Table 3 Response codes

Table 3: Response codes

Response code	Response key	Data type	Description	Mandatory
EMVRESULT (in response to EMVAUTH, EMVCONFIRM, EMVSALE)	tr_ref	String	Transaction ref. Same value as in the initial EMVAUTH/EMVSALE request	Y
	amount	Integer	Minor currency units with 2 decimals implied e.g. 1000 = 10.00	N
	currency	Integer	3 digit currency code e.g. 826 = GBP	N
	status	integer	0 = Success 2 = EMVAUTH or	Y

			EMVSALE Approved 4 = EMVAUTH or EMVSALE Declined 5 = EMVAUTH Voice Referral 6 = EMVCONFIRM Failed 7 – UID/Token provided 8 – UID/Token authorized/approved 9 – UID/Token not authorized/declined 99 = Unknown (error logged)	
	auth_code	string	Authorisation code from the PSP if transaction approved	N
	psp_tr_ref	string	Transaction ref from PSP	N
	pan	String	Taken from a bank card, may be a hash value	N
	method	Integer	Contact or contactless, default is contactless 0 = Contactless	N
	scai	Integer	SCA indicator from the PSP 1 = Fall Forward to Contact (contactless transaction declined, retry with EMV contact-only) 2 = Online PIN required (contactless transaction declined, retry with PIN)	N
	pan_masked	String	If returned by the PSP	N
	scheme	string	Card scheme if returned by payment service provider	N
	psp_timestamp	String	From payment service provider	N
	arc	String	Authorization Response Code	N
	token	String	Token ID, i.e. Mifare Classic UID. Present only if non EMV card is used.	N
TERMSTATUS	report	Dictionary	Returns key/value pair of	Y

(Terminal status)			the report(s) originally requested, along with value(s) e.g. {“swver”:”G2K_T v1.4”}	
CANCEL	status	Integer	Cancel current transaction	Y
PRIVATESCH (Private payment scheme)	status	integer	0 = Success 1 = Fail	Y
	sch_data	string	Holds the card UID when status = 0, otherwise will be spaces	N
RECEIPT **- not yet implemented**	tr_ref	String	tbc	Y
	receipt_params	Dictionary	Returns key/value pairs for receipt data.	N
OFFLINERES Offline result, e.g. Offline approved **- not yet implemented. Potentially in response to EMVSALE)**	tr_ref	String	tbc	Y
	amount	Integer	Minor currency units with 2 decimals implied e.g. 1000 = 10.00	Y
	currency	Integer	3 digit currency code e.g. 826 = GBP	Y
	status	integer	tbc	Y
	pan	String	Taken from a bank card, may be a hash value	N
CUSTID Either pan or crn is returned in the response, never both	status	integer	tbc	Y
	pan	string	Taken from a bank card, may be a hash value	N
	crn	string	Taken from a private card scheme, e.g. ISO14443 Type A card UID	N

10.4 Example JSON of the Response message

(10.4.1) An EMVRESULT response


```
{"version": [1, 0],
"type": "response",
"payload": {
"version": [1, 0],
"code": "EMVRESULT",
"value": {
"tr_ref": "TRN12345",
"amount": 15,
"currency": 826,
"status": 1,
"auth_code": "XY7",
"psp_tr_ref": "PSPTRN12345",
"pan": "123456XXXXX",
"method": 5,
"scai": 1,
"pan_masked": "123456xxxxx",
"scheme": "Scheme12345",
"psp_timestamp": "05MAY2020 18:33"}}}
```

10.5 The Notification message

A **notification** message is published by the Terminal in the **G2KT/notifications** topic. The heartbeat is also a type of notification, published in the **G2KT/HB** topic.

For the **notification** message, **type** will be set to **"notification":**

{

"**version**":<*message version*>,

"**type**": **"notification"**,

"**payload**": {

"**version**": <*payload version*>,

"**code**": <*notification code*>,

"**text**": <*notification text*>

}

}

For the message structure:

- <*message version*> and <*payload version*> are currently unused
- <notification code> and <*notification text*> will be from the table below: Table 4 Notification codes

Table 4: Notification codes

Notification code	Notification text	Notes
A0	CONNECTING	Attempting to connect to the BO. Sent when terminal receives a payment transaction request from the EPOS.
A1	CONNECTION MADE	Connected to BO. Sent at power up, or when reconnecting after C1.
A2	APPROVED	
A3	DECLINED	
A4	INSERT CARD	
A6	PROCESSING ERROR	
A7	REMOVE CARD	
A9	PRESENT CARD	
AA	RE-PRESENT CARD	
AB	CARD NOT SUPPORTED	
AC	PRESENT ONLY ONE CARD	
AD	PLEASE WAIT	
B1	PIN ENTRY	
C0	READER INTERFACE ERROR	Reader interface error. This happens when the Reader is disconnected from the Telemetry
C1	DISCONNECTED	Disconnected from BO.
C2	CONNECT FAIL	
C4	RDRIFACEOK	Reader interface OK. This happens when the Reader is reconnected to the Telemetry
C5	FWUPDATE	Firmware update (internal use)
C6	CFGUPD	Configuration updated (internal use)
C7	SWUPD	Software updated (internal use)
D1	CARD READ OK	
D2	CLEAR DISPLAY	
D3	SEE PHONE FOR INSTRUCTOINS	
FA	HB	Heartbeat message sent at a set interval (default 10 seconds)

10.6 Example JSON of Notification messages

(10.6.1) An A0 'CONNECTING' message

{"version": [1, 0],
"type": "notification",
"payload": {
"version": [1, 0],
"code": "A0",
"text": "CONNECTING"}}

(10.6.2) A heartbeat message

{"version": [1, 0],
"type": "notification",
"payload": {
"version": [1, 0],
"code": "FA",
"text": "HB"}}

11. Handling loss of BO communication

11.1 Overview

The terminal's connection to the BO is established at power up and remains open so that it can send and receive data at any time i.e. it does not initiate connect/disconnect around each transaction. It sends a notification to the EPOS when it connects/reconnects, or detects loss of connection.

All payment transactions initiated by the EPOS are temporarily stored on the terminal before being sent to the BO. If BO comms is unavailable then these transactions will accumulate on the terminal and be automatically sent to the BO once comms resume.

This section describes how the EPOS can detect when comms has been lost, cancel any accumulated transactions if necessary, and prevent further transactions from taking place until comms resume.

11.2 The message codes from the terminal

1. **A1** (Connection made to BO)
    - Issued only at terminal power up, or on reconnecting after C1
2. **C1** (Disconnected from BO)
    - Issued when the terminal has not heard from the BO after a set time has elapsed (currently 180 seconds). This means there is a delay between actual loss of comms and detection.
3. **EMVRESULT** (transaction result)
    - Issued when the terminal receives the payment transaction result from the BO.
4. **FA** (heartbeat)
    - Issued at a set interval (currently 10 seconds)

11.3 How the EPOS can use the message codes

1. Whenever A1 and C1 is received, set a connection state flag (connected/disconnected).
2. When it starts, the EPOS won't know the current state of BO connectivity. It can get it by carrying out the following at power up:
    2.1. Expect to receive A1 within 45 seconds (the terminal may also be powering up at this point, so we give it time to start)
    2.2. If it doesn't arrive after that, send a RESET command to reboot the terminal, then wait 45 seconds for an A1 (causes reconnection to the BO)
    2.3. Assume that there is no BO comms until you get an A1
3. Expect to periodically receive FA (heartbeat)
4. Only accept card transactions when in connected state, to avoid accumulating them on the terminal. Note that in reality it's possible to receive A1 and then immediately

lose comms, but because the terminal will only be aware of lost comms every 180s, it won't issue a C1 until then. The next step will double check it.

5. After sending a payment transaction, expect to receive EMVRESULT response before a reasonable timeout eg. 30s (typical response from the BO is <5s)
6. If EMVRESULT is received, having status = 2 and tr_ref = same one used in payment transaction, then card transaction was successful.
7. If no EMVRESULT is received by the end of the timeout (from step 5):

    7.1. You may assume lost comms, so set connection state = disconnected

    7.2. You may attempt cancellation by sending EMVCONFIRM with amount=0 and tr_ref = same one used in payment transaction

        7.2.1. If the payment transaction was EMVAUTH, cancellation will always succeed as this is the usual method to cancel pre-authorisations.

        7.2.2. If the payment transaction was EMVSALE, cancellation might not succeed, depending on the situation as described below. Either (a) or (b) will apply:

            a) The payment transaction has not yet made it to the BO, as comms were lost before it could be sent:
                - Cancellation will succeed
                - On comms restoration, expect to receive 2 EMVRESULT messages (one for payment transaction and one for cancellation)
            b) The payment transaction has made it to the BO, but comms were lost before a response could be received (extremely unlikely):
                - Cancellation will not be successful as the transaction will have been processed by the BO. In this case, any refund will need to be done manually.
                - On comms restoration, expect to receive 1 EMVRESULT message (for payment transaction)

End of document