

Cloud Printer MQTT Communication Protocol

Document V1.0 · Released 2026-10-07 · Models HS-M80 / HS-830 / HS-K39

MQTT v3.1.1

TLS 支持 / TLS supported

PDF · 图片 · 标签指令

PDF · Image · Label commands

文档信息 / Document info

Vendor: _____ · Technical support: _____ · Version: **V1.0** · Date: **2026-10-07**

0 · Models & Capabilities

This protocol applies to all cloud printer models. **Direct PDF/image printing is a Linux-platform extension** available only on selected models. Always check the table below before sending — a PDF/image job sent to a non-supporting model will not print.

机型 / Model	云打印指令 Command	图片 PNG / BMP Image	PDF PDF	Media width	Notes
HS-M80	支持 Yes	不支持 No	不支持 No	80 mm	Command printing only — \$2 packet format exclusively
HS-830	支持 Yes	支持 Yes	支持 Yes	80 mm	Full capability: command + image + PDF (\$2 / \$5)
HS-K39	支持 Yes	支持 Yes	Yes	4 inch (可调 25–110 mm)	Thermal label printer. Besides cloud commands it natively supports TSPL / JPL / ZPL label languages , plus image + PDF (\$2 / \$5)

Model specifications

Official specifications: **HS-M80 and HS-830 are 80 mm models; HS-K39 is a 4-inch thermal label printer with adjustable 25–110 mm media width.** Of the three, **only HS-M80 lacks PDF/image support** and must use the \$2 command stream.

Useful note on HS-K39

HS-K39 natively supports **TSPL / JPL / ZPL** label languages. Customers with existing label templates **do not need to rasterise labels into images or PDF** — send the original label commands directly, which skips rendering and prints faster. When using TSPL / JPL / ZPL, wrap them with the **\$2 command-stream framing** (Flag + ticket ID + command body) — **full examples in §2.4**.

1 · Deployment & Connection (self-hosted broker + topics)

The printer connects to the MQTT broker on power-up (MQTT v3.1.1). Your backend (the **print server**) acts as another broker client: it publishes jobs to the printer's **subscribe topic** and subscribes to the **publish topics** for status and job results.

1.1 Self-hosted broker · TLS supported

Deployment model

The MQTT broker (host, port, credentials) is deployed by the customer / integrator; the vendor does not provide a public broker. The broker must support **MQTT v3.1.1** (Eclipse Mosquitto recommended; EMQX, HiveMQ and VerneMQ also work). **TLS is supported on the printer side**.

Broker requirements

- **MQTT v3.1.1**; · downlink **QoS 2** must be permitted; · the printer holds a persistent connection from power-on — tune `keepalive` and reconnect behaviour; · broker address / port / credentials / TLS are written via the **configuration tool** and can be verified on the self-test page (§10.1).

1.2 Choosing a host (Linux recommended)

Print workloads mean **many long-lived connections, small messages (≤ 2 MB), 24×7 uptime** — a classic lightweight IoT load. You need **stability, low ops cost and proximity to the printers**, not database-class hardware.

Item	Recommendation	Why
OS	Linux (Ubuntu 22.04 / 24.04 LTS or Debian 12)	Mosquitto is a first-class Linux citizen: one-command install, systemd supervision, conventional log and certificate paths, better long-run memory and connection stability than the Windows build, and cheaper cloud images. Use Windows only for local debugging — not for production .
Cloud / region	Pick the region closest to the printers (Germany → Frankfurt, UK → London) and use a fixed public IP or a stable domain name	MQTT is a persistent, latency-sensitive link — intercontinental round-trips visibly slow down order printing; a changing IP disconnects the entire fleet.
Starting size	Start at 2 vCPU / 2–4 GB RAM / 40 GB SSD; load-test with the real fleet size before go-live	Mosquitto is extremely light — a few hundred persistent printer connections run comfortably on this. At several thousand connections you must raise <code>ulimit -n</code> (file descriptors) and consider a distributed broker such as EMQX.
Ports	<code>1883</code> (plaintext) / <code>8883</code> (TLS, recommended for production)	If the broker is reachable from the internet, always enable TLS plus username/password — otherwise anyone can push print jobs to your printers.
Time sync	Enable NTP on both the server and the printers	The printer uses its own clock to decide whether a job has expired (§2.2); clock drift causes jobs to be wrongly discarded or printed past their deadline.

Never use a public test broker in production

Public brokers such as `broker.emqx.io` and `test.mosquitto.org` are for local testing only. **Anyone can subscribe to your topics** (leaking receipt contents and store orders), and they may throttle or disappear at any time.

1.3 Setting up Mosquitto on Linux (with TLS)

Examples below use **Ubuntu 22.04 / 24.04**. On other distributions replace `apt` with your package manager (package names and config paths are largely the same).

① Install

```
sudo apt update
sudo apt install -y mosquitto mosquitto-clients
sudo systemctl enable --now mosquitto
```

② Create credentials (no anonymous access)

```
sudo mosquitto_passwd -c /etc/mosquitto/passwd printer
# 再追加业务系统用的账号 use a second account for your backend
sudo mosquitto_passwd -b /etc/mosquitto/passwd backend 'StrongPasswordHere'
sudo chmod 640 /etc/mosquitto/passwd
sudo chown root:mosquitto /etc/mosquitto/passwd
```

③ Configuration

```
# /etc/mosquitto/conf.d/cloudprinter.conf

listener 1883 0.0.0.0          # 内网明文端口，可按需删掉
allow_anonymous false         # 必须显式关闭匿名
password_file /etc/mosquitto/passwd

persistence true
persistence_location /var/lib/mosquitto/
log_dest file /var/log/mosquitto/mosquitto.log

max_queued_messages 1000      # 离线队列上限，避免打印机掉线后堆积
max_keepalive 3600            # 打印机长连，放宽 keepalive 上限
```

The classic mosquitto 2.x trap

Since **mosquitto 2.0** the default is to listen on localhost only, with anonymous access disabled. Without explicit `listener` and `password_file` directives the symptom is: "local `mosquitto_sub` works, but the printer never connects". **Write both explicitly.**

④ Enable TLS (mandatory for public deployments)

```
# 已有证书时（推荐用 Let's Encrypt / 企业内网 CA）
sudo mkdir -p /etc/mosquitto/certs
# 把 server.crt / server.key / ca.crt 放进去，注意私钥权限
sudo chmod 640 /etc/mosquitto/certs/server.key
sudo chown root:mosquitto /etc/mosquitto/certs/server.key
```

```
# 追加到 /etc/mosquitto/conf.d/cloudprinter.conf
listener 8883 0.0.0.0
certfile /etc/mosquitto/certs/server.crt
keyfile /etc/mosquitto/certs/server.key
cafile /etc/mosquitto/certs/ca.crt
require_certificate false      # 打印机用账号密码认证；如需双向 TLS 改 true
allow_anonymous false
password_file /etc/mosquitto/passwd
```

How to enable TLS on the printer (port, certificate format, the switch inside the configuration tool) is documented in §1.7.

⑤ Open the firewall and restart

```
sudo ufw allow 1883/tcp
sudo ufw allow 8883/tcp
sudo systemctl restart mosquitto
sudo systemctl status mosquitto
sudo tail -f /var/log/mosquitto/mosquitto.log
```

⑥ Verify from the CLI (before connecting a printer)

```
# 终端 A: 订阅两个上报主题
mosquitto_sub -h 127.0.0.1 -p 1883 -u backend -P 'StrongPasswordHere' \
  -t 'heartbeat' -t 'PrintSuccess' -v

# 终端 B: 模拟一次打印结果
mosquitto_pub -h 127.0.0.1 -p 1883 -u backend -P 'StrongPasswordHere' \
  -t 'PrintSuccess' -m '4;[PrnTEST];9800;TEST-0001' -q 2
```

⑦ Verify with real hardware

- 1 Print the self-test page (power off, hold FEED while powering on), scan the QR code for the **CPUUID** — the subscribe topic is `Prn{CPUUID}`.
- 2 Run `mosquitto_sub -t 'heartbeat' -v` on the server, then power on the printer.
- 3 Within seconds you should receive the **online message** (type 1, §3.1 B). That confirms the whole path printer → broker → your subscriber.
- 4 Publish `01 00` (hex) to `Prn{CPUUID}` and confirm a status reply; then send a real job and confirm paper comes out.

1.4 Subscribe topic (downlink)

Item	Value	
<code>Prn</code> + CPUUID	Format	<code>Prn</code> + CPUUID
<code>Prn60A44CAB3AA72020170420170648017920201704201706480179</code>	Example	<code>Prn60A44CAB3AA7...480179</code>

- **How to obtain the CPUUID:** power off, hold the FEED key while powering on — the printer prints a self-test page containing a QR code with the ID. In the configuration tool, when "Sub Topic" is set to "Generate From CPU UUID", the subscribe topic is `Prn` + CPUUID. Devices can also be configured to derive this ID from the **MAC address**.
- **Every printer has its own topic**, keyed by CPUUID. For fleet rollouts, keep a mapping table of "self-test QR → CPUUID → store / station".

1.5 Publish topics (uplink)

Purpose	Topic	Remark	
<code>heartbeat</code>	Printer status (heartbeat / state change)	<code>heartbeat</code>	Online, offline, heartbeat, status change — see §3 types 0 / 1 / 2 / 7
<code>PrintSuccess</code>	Job result	<code>PrintSuccess</code>	Received / printed / expired / duplicate ID / unparsable — see §3 types 3 / 4 / 5 / 8 / 9
<code>{型号}_UPDATE{版本号}</code>	Firmware update (printer subscribes)	<code>{model}_UPDATE{version}</code>	e.g. KP202 fw 1.06 → <code>KP202_UPDATE1.06</code>

Default uplink topics

There are exactly **two** default uplink topics: `heartbeat` carries printer status, `PrintSuccess` carries job results. **Subscribe to both.**

1.6 Printer identity in uplink messages

Every uplink message carries the printer identity wrapped in square brackets, by default identical to the subscribe topic:

`[Prn60A44CAB3AA7...480179]`. When the `Client ID` field in the configuration tool is left blank, the printer usually uses its own MAC address as the MQTT client ID (e.g. `14-7D-DA-22-9B-2C`). It can be configured to be MAC-derived, or omitted entirely. Referred to as `ID` below.

Fleet warning: MQTT client IDs must be unique

MQTT mandates that **duplicate client IDs kick each other off** the broker. In fleet deployments make sure each printer uses a unique identity (the CPUUID-based default satisfies this) — otherwise you get the notoriously hard-to-diagnose "two printers dropping in turns" fault.

1.7 Printer-side MQTT config (PrinterSetting tool)

Use the vendor-supplied **PrinterSetting** tool to write broker credentials into the printer (screenshot shows PrinterSetting-3.1.3.20230208). Connect the printer via USB, enter the parameter-setting page and switch to the **MQTT** tab.

- 1 Connect the printer to the PC with a USB cable, open `printersetting.exe`, select **USB** under "Select Communication Port", then click **Enter para setting**.
- 2 Once inside the parameter page, click the **MQTT** tab in the top tab bar.
- 3 Fill in the server address, port, username and password, choose a TLS option if needed, then click the **Set** button next to each field to write it to the printer.

The screenshot shows the 'PrinterSetting-3.1.3.20230208' application window. It has two main sections: '1. Select Communication Port' and '2. Check Printer'. In the '1. Select Communication Port' section, the 'FILE' option is selected with the path 'C:/Users/mac/Desktop/data.bin'. In the '2. Check Printer' section, the 'MQTT' tab is selected in the top navigation bar. Below the tabs, there are several configuration fields for MQTT: 'Server Address' (43.154.166.244), 'Port' (1883), 'User Name' (test), and 'Password' (test). There are 'Set' buttons next to each of these fields. Below these fields, there are more configuration options for various services like TEST WEB, NTP Server, Sub Topic, Pub Receipt Info, Pub Qos, HeartBeat Interval, HeartBeat Publish Topic, TicketReplay Publish Topic, and KeepAlive Interval, each with a 'Set' button. At the bottom, there is a 'TLS' section with a dropdown menu for 'TLS Option' (Don't use tls authentication) and a 'Set TLS Option' button. There is also an 'Others' section with a 'Clear Ticket' button.

Figure 1 · MQTT tab of the PrinterSetting tool

Field	Meaning / suggested value
Server Address	IP or domain of the MQTT broker
Port	1883 (plaintext) / 8883 (TLS)
User Name / Password	Account created on the broker
Client ID	Left blank, the printer usually uses its MAC as MQTT client ID; if custom, ensure global uniqueness
Sub Topic	Choose Generate From CPU UUID → subscribe topic is Prn{CPUUUID}
Pub Receipt Info / Pub Qos	Job-result publish settings; Pub QoS suggested 2
HeartBeat Interval / KeepAlive Interval	Suggested 30 seconds; matches §6.1
TLS Option	Don't use tls authentication or enable TLS; when enabled, port must be 8883
NTP Server / Date Zone	Point to your broker or a public NTP server (e.g. ntp1.aliyun.com) and set the local timezone

How settings take effect

Each field has its own **Set** button; click it after changing a value. After configuration, print a self-test page (§10.1) to verify that the MQTT address and port are stored.

1.8 Printer WiFi network configuration

The cloud printer joins the customer's LAN in **WiFi STA mode** and then connects to the MQTT broker described in §1.1. Before first use, write the target WiFi SSID and password into the printer. The vendor provides configuration tools for Windows, iOS and Android; the core flow is the same: establish a local connection between the phone/PC and the printer → enter parameter settings → WiFi tab → fill in SSID / password → tap Set.

Important restriction: 5 GHz WiFi is not supported

The iOS and Android tool screenshots confirm that this printer family **only supports 2.4 GHz WiFi**. Make sure a 2.4 GHz network is available on site. If the router merges 2.4 GHz and 5 GHz under one SSID, temporarily split them, configure the printer, then merge again — or lock the printer to 2.4 GHz in the router.

1.8.1 Windows: PrinterSettingTools.exe

The Windows configuration tool is **printersettingtools.exe** (screenshot shows PrinterSetting-3.1.3.20230208). Connect the printer via USB and follow these steps:

1. In **Connection port** at the top, select **USB** (or the actual serial / network port) and click **Check printer** to confirm the connection.
2. Click the **WiFi** tab in the tab bar below.
3. Enter the 2.4 GHz network name in **SSID** and the password in **Password**.
4. Click the **Set** button in the STA area and wait for the success message.
5. Restart the printer and print the self-test page to confirm it obtained a WiFi IP (§10.1).

The screenshot shows the 'PrinterSetting-3.1.3.20230208' application window. The 'WiFi' tab is selected in the top navigation bar. Under the '1. 选择通讯端口' (Select communication port) section, the '网络' (Network) option is selected with IP '192.168.1.166' and port '9100'. The '2. 检查打印机' (Check printer) section has buttons for '打印自检页' (Print self-test page) and '读取打印机参数' (Read printer parameters). The '显示专业设置' (Show professional settings) checkbox is checked. In the 'STA' section, 'SSID' is 'HSPOS-LOCAL' and '密码' (Password) is 'HSpos2015', with a '设置' (Set) button below. The 'P2P' section has empty fields for 'SSID后缀' (SSID suffix) and '密码' (Password), also with a '设置' (Set) button. At the bottom, the 'Airkiss' section has two buttons: '启用Airkiss' (Enable Airkiss) and '禁用Airkiss' (Disable Airkiss).

Figure 1-1: Windows config tool WiFi tab (STA mode)

What are P2P and Airkiss?

The same WiFi tab also has **P2P** and **Airkiss** areas. P2P turns the printer into an access point so a phone can connect directly to it for provisioning. Airkiss is an older WeChat one-click provisioning protocol. For typical restaurant / retail deployments, **use STA mode and enter the SSID + password directly** — it is the simplest and most stable. Consider P2P / Airkiss only when USB access is unavailable or the password cannot be entered directly.

1.8.2 iOS: Printer Settings Tool

On iPhone / iPad use the **Printer Settings Tool** app (Bluetooth connection). The flow is:

1. Open the app, choose **Bluetooth** under **Connection Type**, and connect to the target printer (the device name and BDA address are shown).
2. Tap **Parameter Settings** in Functions.
3. Select **WiFi Settings**.
4. Enter **WiFi Name** and **WiFi Password**, then tap **Set**.
5. Restart the printer and print the self-test page to confirm network access.

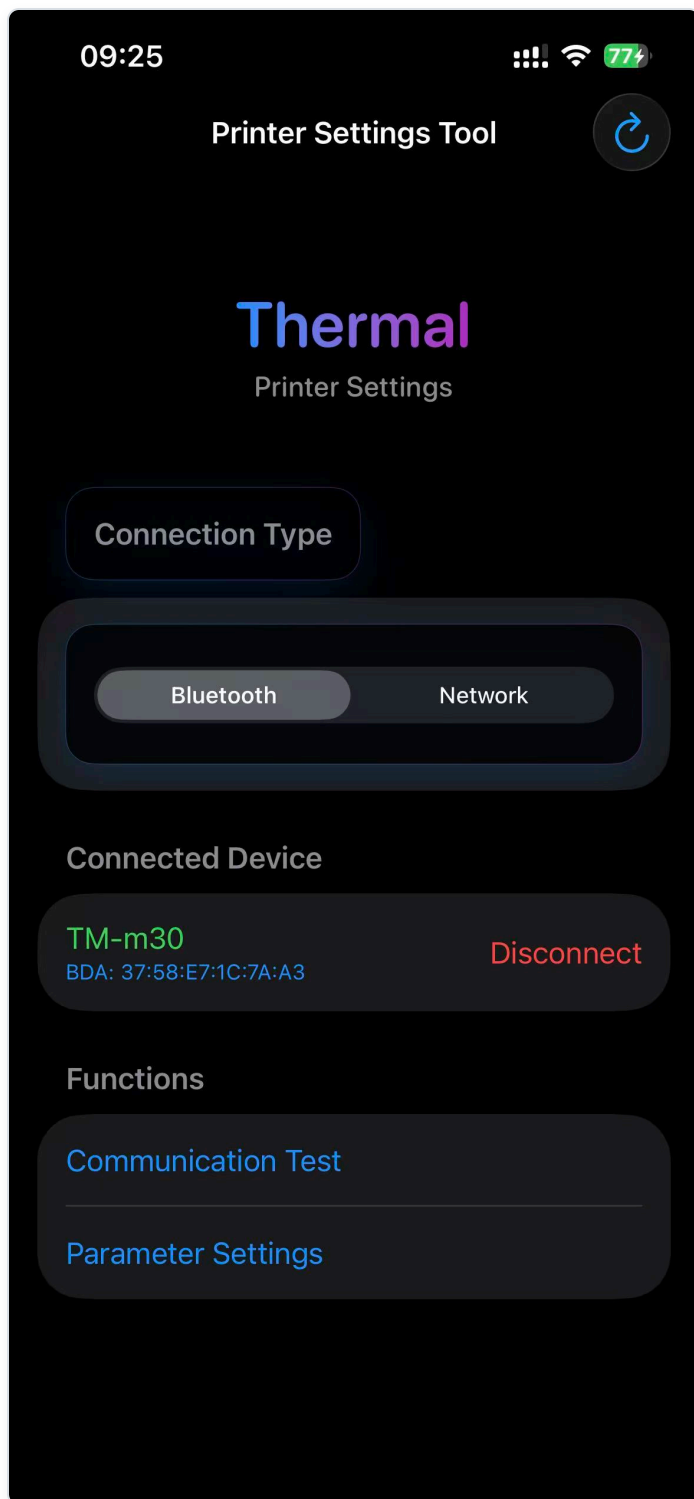


Figure 1-2: iOS app connecting via Bluetooth

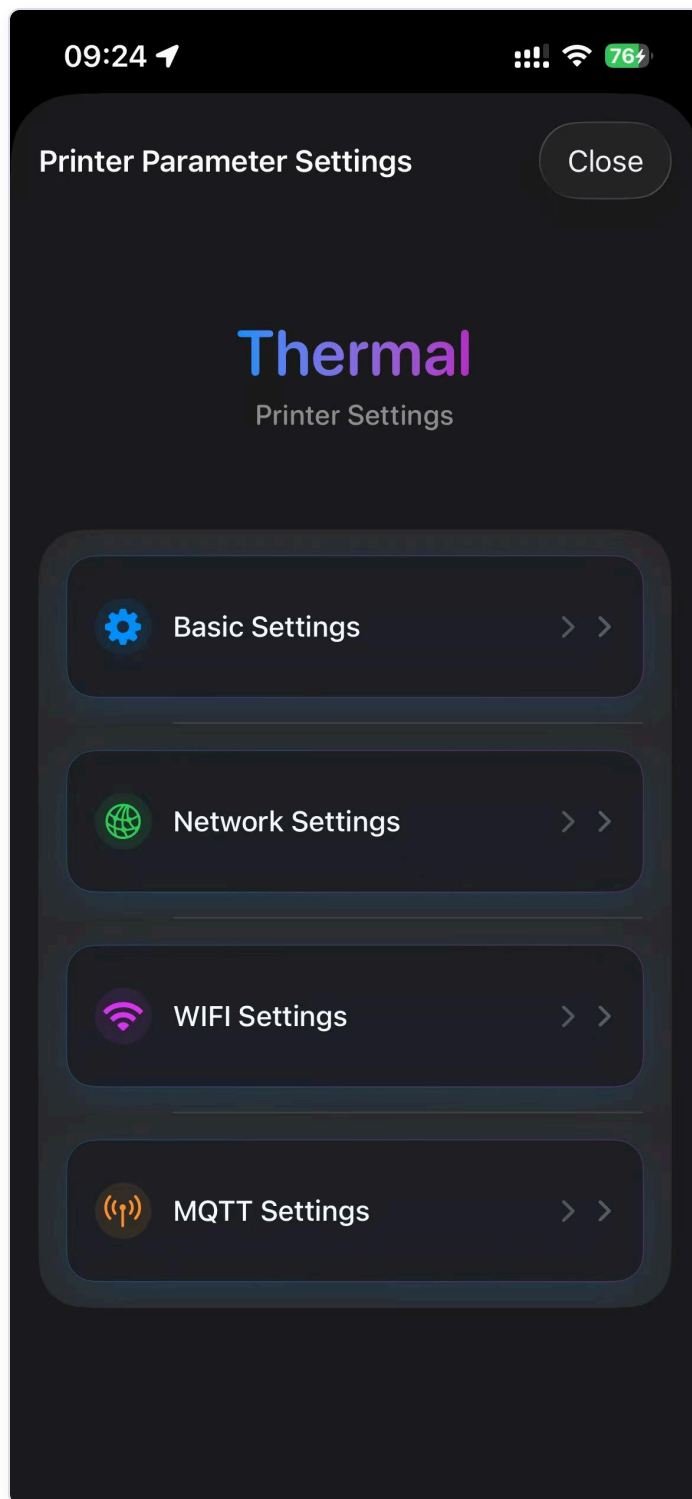


Figure 1-3: Parameter Settings → WiFi Settings

09:26



WIFI Settings

WIFI Name

HSPOS-LOCAL

WIFI Password

Note: 5G WIFI not supported

Set

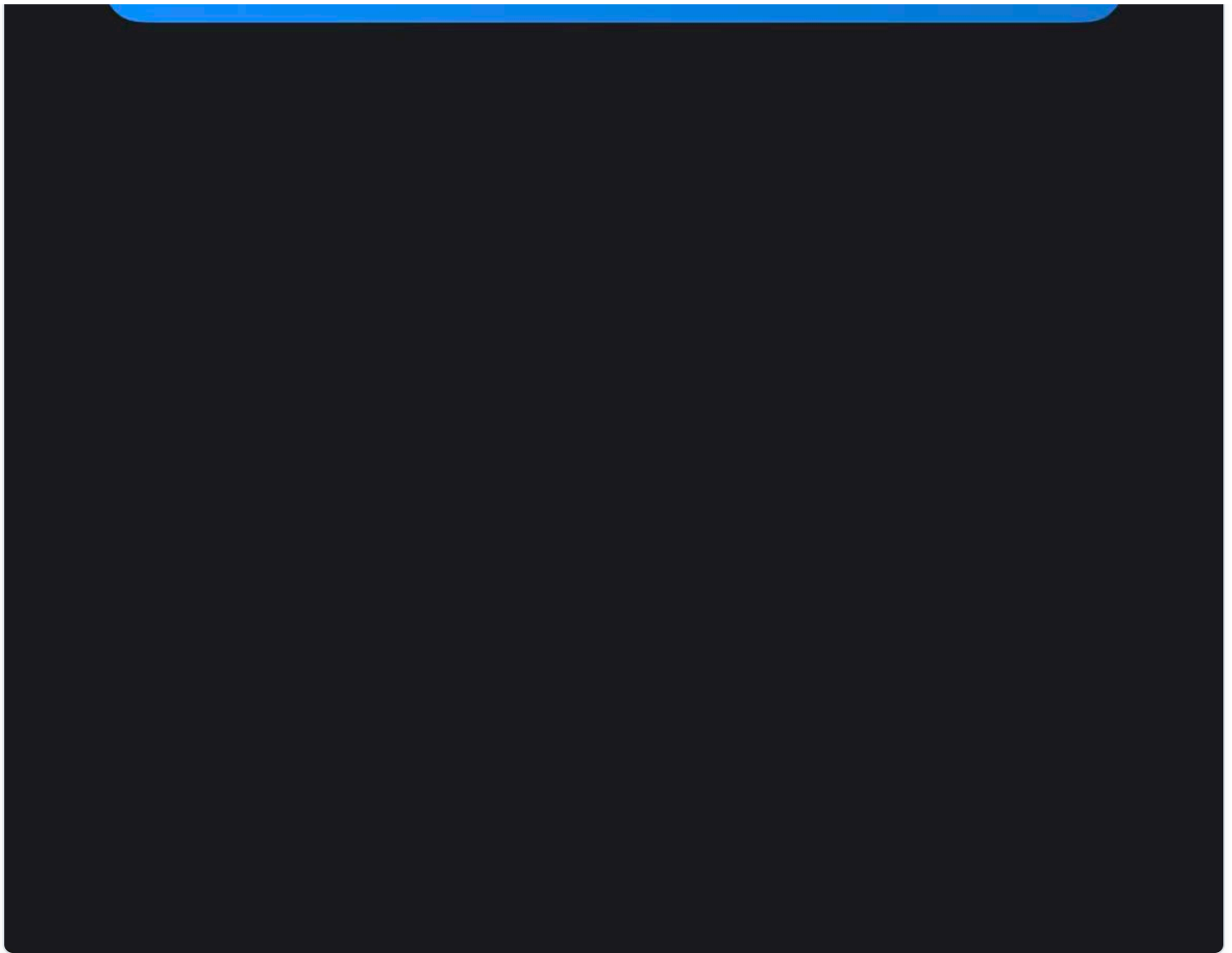


Figure 1-4: Fill WiFi Name / Password; note the "5G WiFi not supported" warning

1.8.3 Android: HSPOS app

On Android use the **HSPOS** app and connect via Bluetooth:

1. Open the HSPOS app and tap **网络连接** (Network connection).
2. Pair with the target printer over Bluetooth.
3. Enter **参数设置** (Parameter settings) and choose the **WiFi** tab.
4. Enter **WiFi名称** (WiFi name) and **WiFi密码** (WiFi password), then tap **设置** (Set).
5. Restart the printer and print the self-test page to confirm.



Figure 1-5: HSPOS app home → Network connection

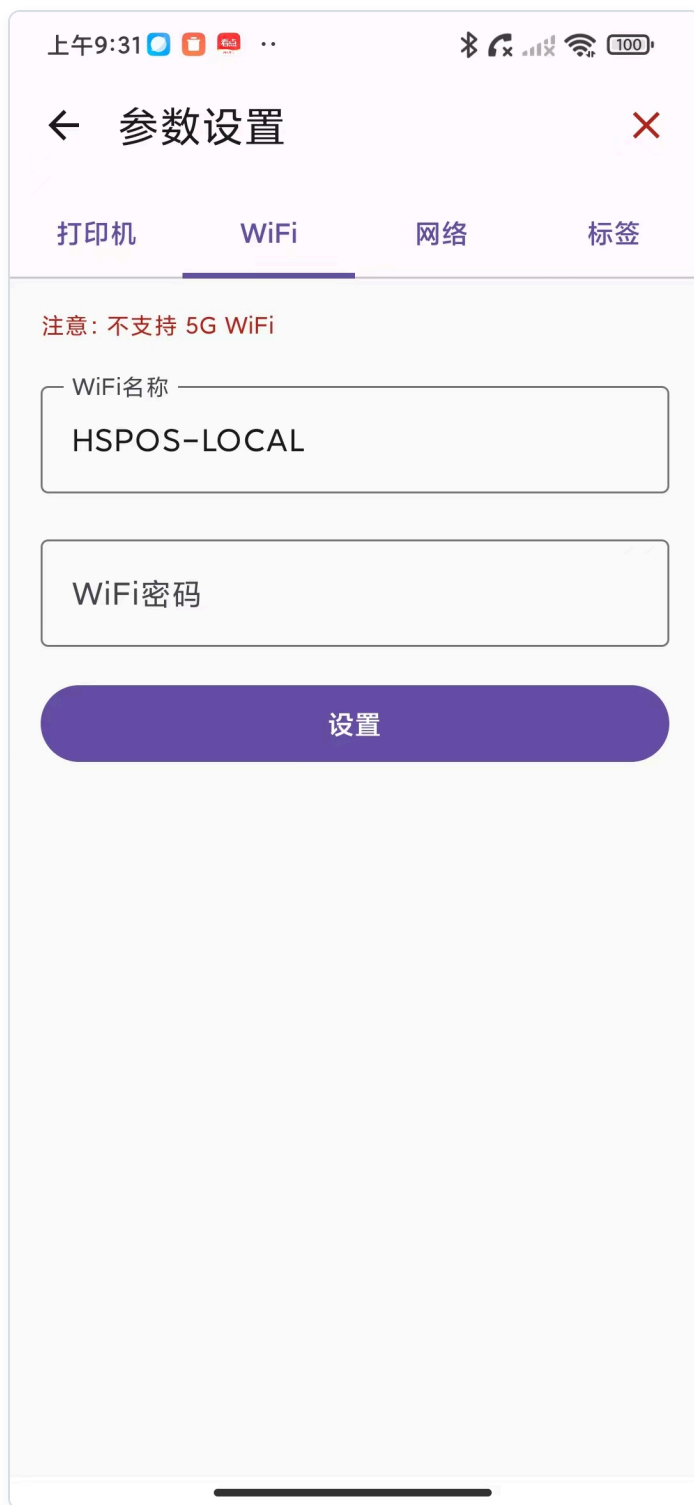


Figure 1-6: Parameter settings → WiFi tab; note the "5G WiFi not supported" warning

WiFi configuration checklist

① Confirm a 2.4 GHz network is available. ② Avoid special characters or Chinese characters in the SSID / password (some firmware may not support them). ③ Always restart the printer after writing WiFi settings. ④ Print the self-test page and verify IP, subnet mask, gateway and MQTT broker address. ⑤ If the printer cannot get online, first verify from a phone on the same WiFi that the broker address and port are reachable.

1.9 MQTT printer test tool (HSPOS MQTT PrinterTest)

The vendor also supplies the **HSPOS MQTT PrinterTest** desktop tool for quickly verifying the full path "publish → printer → report" before formal integration. The screenshot below shows a typical debugging session.

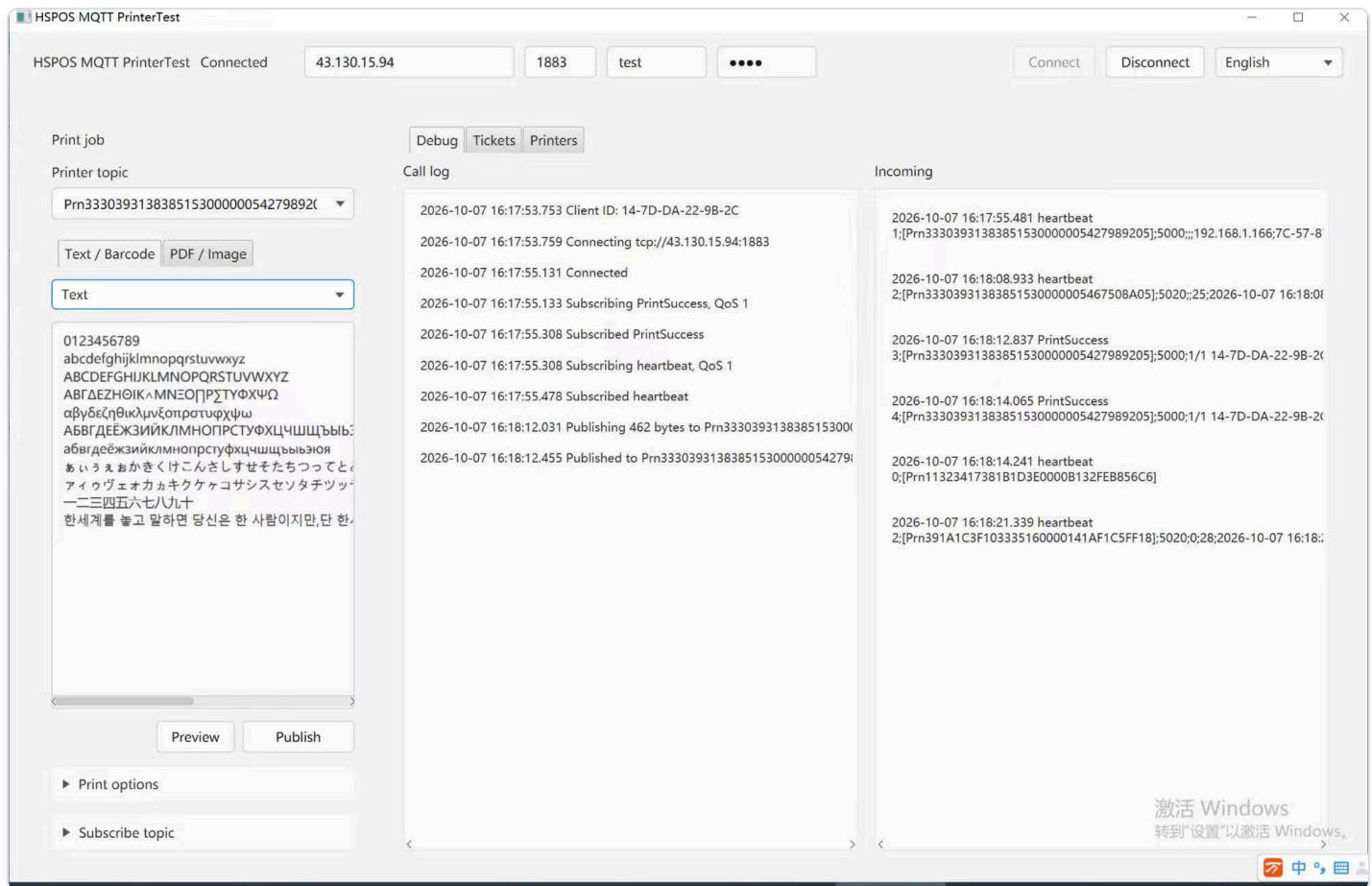


Figure 2 · HSPOS MQTT PrinterTest debug screen

- 1 Enter the broker address, port, username and password at the top, then click **Connect**.
- 2 In the **Printer topic** dropdown, select the target printer (`Prn{CPUUID}`).
- 3 Switch to the **Text / Barcode** or **PDF / Image** tab, enter content and click **Publish**.
- 4 The **Incoming** panel on the right shows `heartbeat` and `PrintSuccess` replies.

Sample uplink messages seen in the tool

```
2026-10-07 16:17:55.481 heartbeat 1:[Prn333039313838515300000054279892f];5000;;192.168.1.166
2026-10-07 16:18:08.933 heartbeat 2:[Prn33303931383851530000005467508A05];5020;25;2026-10-07
2026-10-07 16:18:12.837 PrintSuccess 3:[Prn333039313838515300000054279892f];5000;1/1 14-7D-D
2026-10-07 16:18:14.065 PrintSuccess 4:[Prn333039313838515300000054279892f];5000;1/1 14-7D-D
```

What these samples verify

- Type **1** is the online message — the printer has joined the broker;
- Type **2** is a heartbeat showing STATE (**5020**), signal and temperature;
- Type **3** is the "received" ack; **1/1** means the first page of a 1-page job was received;
- Type **4** is "printed"; the same **1/1** means the job finished.

The test tool is not a production client

It is good for single-printer debugging and demos. For production, build your own print server as described in §8 / §9 to guarantee unique ticket IDs, reconnect, retry and logging.

2 · Downlink Packet Format (command stream & settings)

Applies to

This framing applies to **command printing and setting commands only**. PDF/image jobs use JSON with no Flag byte — see §5.

Every downlink packet has the following structure (hex):

Flag	指定回复主题(0-64 B)	单据号(0-64 B)	打印内容(0-16000 B)
1 byte	N bytes + 0x00 结尾	N bytes + 0x00 结尾	

Total length limit

The whole packet must not exceed $16384 - 6 = 16378$ bytes.

2.1 Flag byte

Bit	值 Value (HEX)	Meaning
0	0x01	Publish print-success status to the specified topic
1	0x02	Packet carries a ticket ID
2	0x04	Packet is a setting command (not a print job)
3	0x08	Packet carries an expiry timestamp
4–7	0x10/0x20/0x40/0x80	Fixed to 0

JSON payloads carry no Flag byte

PDF/image JSON payloads carry **no Flag byte** — the entire message body is the JSON document, published straight to the subscribe topic (see §5). The Flag table above applies only to the **command-stream / setting-command** packets of §2.

2.2 Field details

Reply topic (0–64 bytes, 0x00 -terminated)

A zero-length field (a single 0x00) means "use the default job topic". Otherwise up to 64 bytes of custom topic.

Ticket ID (0–64 bytes, 0x00 -terminated)

Generated by your server. **The printer performs duplicate checking: if two different jobs share the same ticket ID, the later one is discarded** and a duplicate-ID message is published (type 8). Since QoS 2 is unavailable on standard IoT platforms, the ticket ID is the only de-duplication mechanism — make it unique.

Setting payload (when bit2 = 1)

- The payload is treated as settings, not a print job, and **must be smaller than 4 KB**.
- A ticket ID may be included to detect whether the setting was applied — the reply appends OK or ERROR to the end of the ticket ID.

Expiry timestamp (when bit3 = 1)

- Placed at the **head of the print content**, wrapped by 0x06 on both sides, **little-endian**: 06 FF 6A 77 59 06 = 0x59776AFF.
- If printer time > expiry, the job is discarded and a timeout message (type 5) is published. **The "received" acknowledgement (type 3) is still sent regardless of expiry.**
- Gotcha:** if the timestamp is malformed, the printer **will not** return a timeout packet — it prints the job regardless of expiry.

2.3 Downlink examples (all hex)

A · Simple print job (with ticket ID + receipt)

03 00 53 69 6D 70 6C 65 50 72 69 6E 74 00 48 65 6C 6C 6F 2C 20 57 6F 72 6C 64 21 0D 0A

03 Flag = 0x01 | 0x02 → ticket ID present + receipt requested

00 Empty reply topic → default job topic

53 69 6D 70 6C 65 50 72 Ticket ID "SimplePrint" + 0x00

69 6E 74 00

48 65 6C 6C 6F ... 21 0D Content "Hello, World!"\r\n

0A

B · Setting command (heartbeat = 30 s)

07 00 73 65 74 20 70 61 72 61 00 53 45 54 20 48 45 41 52 54 42 45 41 54 20 33 30 0D 0A

07 Flag = 0x01 | 0x02 | 0x04 → ticket ID + receipt + setting command

00 Default reply topic

73 65 74 20 70 61 72 61 Ticket ID "set para" + 0x00

00

53 45 54 ... 33 30 0D 0A Payload "SET HEARTBEAT 30"\r\n

C · Query with no ticket ID (printer status)

01 00

01 = reply status to the specified topic; 00 = default topic.

D · Job with expiry timestamp

0B 00 53 69 6D 70 6C 65 50 72 69 6E 74 00 06 FF 6A 77 59 06 48 65 6C 6C 6F 2C 20 57 6F 72 6C

0B Flag = 0x01 | 0x02 | 0x08 → ticket ID + receipt + expiry

06 FF 6A 77 59 06 Unix timestamp 0x59776AFF (little-endian) = 2017-07-25 23:59:59 Beijing time

E · Firmware update packet (address 0x5000)

00 00 50 00 B0 F8 CC 00 10 B9 9D 42 00 D1 25 46 88 42 02 D0 ...

Update packets are sent in **4 KB (4096-byte) units**, prefixed with 4 bytes of address, **big-endian**. Here 00 00 50 00 = address 0x00005000; total packet length = 4096 + 4 bytes.

2.4 Label-command examples (TSPL / ZPL)

Applicable model

Label command languages (TSPL / JPL / ZPL) are **confirmed on HS-K39 only**. HS-M80 and HS-830 are not listed as supporting label languages — **do not assume support**; confirm with the vendor.

Label commands are framed exactly as in §2: **Flag + reply topic + ticket ID + command body**. The body is a plain TSPL or ZPL script — no base64, no JSON.

Flag	回复主题	单据号	TSPL / ZPL 指令文本
0x03	00 (默认)	"LBL-0001" + 00	SIZE 100 mm,150 mm\r\nCLS\r\n...

A · TSPL example (100 × 150 mm label: text + Code128 + QR)

```
SIZE 100 mm,150 mm
GAP 3 mm,0 mm
DIRECTION 1
CLS
TEXT 50,50,"3",0,1,1,"Cloud Printer"
TEXT 50,110,"2",0,1,1,"HS-K39 / TSPL"
BARCODE 50,170,"128",100,1,0,2,4,"CP20261007001"
QRCODE 50,300,"H",5,"A",0,"https://example.com/o/CP20261007001"
PRINT 1
```

- SIZE w,h

Label size. Unit depends on firmware setting (mm or dots) — check the unit first if sizes look wrong
- GAP m,n

Gap stock: gap height, gap offset. For **continuous media** omit this line or use `GAP 0,0`; for **black-mark media** use `BLINE`
- CLS

Clear the buffer — **required at the start of every label**
- TEXT

Text. font "1"–"5" (larger = bigger), rot = rotation 0–3
- x,y,"font",rot,xmul,ymul,"content"
- BARCODE

1D barcode — Code128 here
- x,y,"type",height,readable,rot,narrow,wide,"content"
- QRCODE

QR code. ECC is L / M / Q / H
- x,y,ECC,cellwidth,mode,rot,"content"
- PRINT n

Print and finish — **must be last**; n = number of copies

B · ZPL example (the same label)

```

^XA
^PW800
^LL1200
^FO50,50^A0N,45,45^FDCloud Printer^FS
^FO50,120^A0N,30,30^FDHS-K39 / ZPL^FS
^FO50,180^BY2^BCN,90,Y,N,N^FDCP20261007001^FS
^FO50,320^BQN,2,5^FDMM,Ahttps://example.com/o/CP20261007001^FS
^XZ

```

<code>^XA</code> / <code>^XZ</code>	Start / end of label — always paired
<code>^PW</code> / <code>^LL</code>	Print width / label length in dots : 4 inch at 203 DPI $\approx$ 812 dots
<code>^FO x,y</code>	Field origin (coordinates)
<code>^A0N,h,w</code>	Font and size
<code>^FD...^FS</code>	Field data / field separator — always paired
<code>^BY2^BCN,90,Y,N,N</code>	Code128: module width 2, height 90, human-readable line shown
<code>^BQN,2,5</code>	QR code: model 2, magnification 5

C · Publish code (Python, TSPL)

```

TOPIC = "Prn60A44CAB3AA72020170420170648017920201704201706480179"

tspl = (
    'SIZE 100 mm,150 mm\r\n'
    'GAP 3 mm,0 mm\r\n'
    'DIRECTION 1\r\n'
    'CLS\r\n'
    'TEXT 50,50,"3",0,1,1,"Cloud Printer"\r\n'
    'TEXT 50,110,"2",0,1,1,"HS-K39 / TSPL"\r\n'
    'BARCODE 50,170,"128",100,1,0,2,4,"CP20261007001"\r\n'
    'QRCODE 50,300,"H",5,"A",0,"https://example.com/o/CP20261007001"\r\n'
    'PRINT 1\r\n'
).encode()          # 注意：按打印机要求用 CRLF 结尾

ticket = "LBL-20261007-0001"
pkt = bytes([0x03]) + b"\x00" + ticket.encode() + b"\x00" + tspl
assert len(pkt) <= 16378          # §2 总长上限
c.publish(TOPIC, pkt, qos=2)

```

Integration tip: print a minimal label first

The TSPL / ZPL syntax above follows the **public specifications** of those languages. Firmware may differ on parameter values (units, font sizes, DPI), so **start integration with the shortest possible script** and add content once paper comes out:

Send this minimal label first; add content once paper comes out:

TSPL: `SIZE 100 mm,100 mm\r\nCLS\r\nTEXT 50,50,"3",0,1,1,"OK"\r\nPRINT 1\r\n`

ZPL: `^XA^F050,50^A0N,40,40^FDOK^FS^XZ`

D • Common label-printing pitfalls

1. **Skipping media calibration skews the feed:** after changing label stock, run a **gap / black-mark calibration** via the self-test page or config tool, otherwise every label drifts.
2. **Confirm units and DPI:** TSPL `SIZE` and coordinates may be mm or dots; ZPL is always dots (203 DPI $\approx$ 8 dots/mm, 300 DPI $\approx$ 11.8 dots/mm). Check both if positioning is off.
3. **Do not omit the terminator:** TSPL needs `PRINT`, ZPL needs `^XZ`. Without it nothing prints and **no error is reported** — the most common "sent but nothing happened".
4. **One label = one MQTT message:** to print several labels, send several messages with distinct ticket IDs; do not batch multiple labels into one message (the 16378-byte cap from §2 applies).
5. **Label commands are optional:** if you already have label artwork, render it to PNG / PDF and send it via the \$5 JSON path — best compatibility and easiest debugging.

3 • Uplink Message Formats

Uplink messages are **plain text**, fields separated by semicolons `;`, with the message type number first.

Type	Description	
0	offline;ID	Printer went offline
1	online;ID;STATE;IMEI;IMSI;IP;MAC;TIME;VER;MachineType	Login / online
2	heartbeat;ID;STATE;GPRSdBm;TEMP;TIME	Heartbeat
3	receivedID;ID;STATE;receiptID-Received	Job received (ack)
4	printDone;ID;STATE;receiptID	Job printed
5	timeout;ID;STATE;receiptID	Job expired and discarded
6	—	Unused
7	stChange;ID;STATE	Status changed (out of paper, cover open...)
8	discard;ID;STATE;receiptID	Duplicate ticket ID — job discarded
9	9;ID;STATE;receiptID	Job abandoned. Currently triggered only when a PDF/image cannot be parsed (e.g. oversized)

Field meanings

ID printer unique code (MAC-based); STATE status word (\$4); IMEI / IMSI cellular module IDs; IP uplink address; MAC Ethernet MAC; TIME current network time (varies with timezone setting); GPRSdBm signal strength; TEMP head temperature; receiptID the ticket ID you supplied; VER firmware version; MachineType model.

3.1 Uplink examples (plain text)

A • Offline

```
0;[Prn60A44CAB3AA720201704201706480179]
```

B • Online

```
1;[Prn60A44CAB3AA720201704201706480179];9800;860832038705287;460013990009623;10.45.0.244;60-
```

9800 = no error + currently on GPRS; 1.07 firmware; KP202 model.

C · Heartbeat

```
2;[Prn60A44CAB3AA720201704201706480179];9820;-58;25;2017-06-22 13:55:28
```

-58 dBm signal; 25 head temperature.

D · Job received

```
3;[Prn60A44CAB3AA720201704201706480179];9800;SimplePrint-Received
```

-Received is a fixed suffix.

E · Job printed

```
4;[Prn60A44CAB3AA720201704201706480179];9800;SimplePrint
```

F · Job timeout

```
5;[Prn60A44CAB3AA720201704201706480179];9800;SimplePrint
```

G · Status change

```
7;[Prn60A44CAB3AA720201704201706480179];9801;-58;25;2017-06-22 13:55:28
```

9801 = out of paper + GPRS link.

H · Duplicate ticket ID

```
8;[Prn60A44CAB3AA720201704201706480179];9800;SimplePrint
```

4 · Status Word (STATE)

The status word is 16-bit and reported in **decimal** (e.g. 9800). Bit definitions:

Bit(s)	值 Value (HEX)	Meaning
0	0x0001	Out of paper
1	0x0002	Cover open
2	0x0004	Cutter error
3	0x0008	Overheat
4	0x0010	Other error
5	0x0020	This reply is a heartbeat
6–7	—	Reserved
11–12	0x0000	Reserved
	0x0800	Active link: Ethernet
	0x1000	Active link: Wi-Fi
	0x1800	Active link: GPRS
13	0x2000	Ethernet connected
14	0x4000	Wi-Fi connected
15	0x8000	GPRS connected

Worked example: reading 9820

9820 = 0x8000 (bit15 GPRS up) + 0x1800 (bits 11–12: active link GPRS, i.e. bits 13/14 clear → Ethernet & Wi-Fi down) + 0x20 (bit5 heartbeat) + bits 0–4 clear (no error).

→ **GPRS connected, Ethernet and Wi-Fi not connected, traffic over GPRS, this is a heartbeat, printer healthy.**

5 · Direct PDF / Image Printing (JSON)

Applicable models

This section covers the **Linux-based cloud printers** (supported on HS-830 and HS-K39; **not** on HS-M80). **The downlink topic and the result-reply topic are unchanged from §1.**

Framing: no Flag byte — the whole payload is JSON

PDF/image jobs carry **no Flag byte, no reply-topic field and no separate ticket-ID field** from §2. **Publish the JSON text directly to the printer's subscribe topic**; the ticket ID lives in the JSON `ticket_id` field. Results still come back on `PrintSuccess`.

Encode the PDF/image as base64 and send it as JSON:

字段 Field	Type	Description
<code>ticket_id</code>	string required	Ticket ID. Must be unique — duplicates are not printed.
<code>data_type</code>	string required	One of <code>pdf</code> / <code>png</code> / <code>bmp</code>
<code>data_base64</code>	string required	Base64 of the file
<code>paper_type</code>	int optional (default 0)	<code>0</code> printer decides / <code>1</code> continuous / <code>2</code> label. With label paper, gap is located automatically before and after printing.
<code>paper_width_mm</code>	int optional (default 0)	Media width; <code>0</code> = printable width. E.g. an 80 mm model has 80 mm max and 72 mm default printable width. <code>0</code> /omitted → PDF scaled to 72 mm; <code>54</code> → scaled to 54 mm; values above the maximum are clamped.
<code>paper_height_mm</code>	int optional (default 0)	Media height; <code>0</code> = no page-height limit; non-zero guarantees content is scaled to fit. Use it for label printing; set 0 for receipts.
<code>cut_paper</code>	int optional (default 1)	<code>0</code> no cut / <code>1</code> cut after each page / <code>2</code> cut at end of document
<code>serial_paper_img_print_no_tail_feed</code>	int optional (default 0)	Suppress trailing feed for continuous-paper image printing; only effective when cutting is off. <code>0</code> auto-feed to tear position / <code>1</code> no auto-feed.

5.1 JSON example

```

{
  "ticket_id": "ORDER-20261007-0001",
  "data_type": "pdf",
  "data_base64": "JVBERi0xLjcKJeLjz9MKNiAwIG9iago8PCAvVHlwZSAvUGFnZSAvUGFyZW50IDEgMCBSIC9NZWl
  "paper_type": 0,
  "paper_width_mm": 0,
  "paper_height_mm": 0,
  "cut_paper": 1,
  "serial_paper_img_print_no_tail_feed": 0
}

```

5.2 Hard limits (exceeding them = no print)

1. **Total MQTT message ≤ 2 MB.** Printer resources are limited — compress or split the job yourself.
2. **Decoded image pixel data ≤ 8 MB.** Exceeding it returns an error and nothing prints (uplink type 9). Decoded size = bit depth $\times$ width $\times$ height.
3. **Ticket IDs must be unique** — duplicates are silently discarded (uplink type 8). Remember to change the ID when testing.

For a 640-dot-wide image, the 8 MB budget translates to these maximum heights:

Bit depth	Bytes per row (4-byte aligned)	Max image height	$\approx$ at 8 dots/mm			
$(640+7)/8 = 80$	$8388608/80 = 104857$	≈ 13.1 m	Mono, 1 bpp	$(640+7)/8 = 80$	$8388608/80 = 104857$	≈ 13.1 m
$(640+3)/4*4 = 640$	$8388608/640 = 13107$	≈ 1.6 m	Grayscale, 8 bpp	$(640+3)/4*4 = 640$	$8388608/640 = 13107$	≈ 1.6 m
$640*4 = 2560$	$8388608/2560 = 3276$	≈ 409 mm	RGB, 32 bpp	$640*4 = 2560$	$8388608/2560 = 3276$	≈ 409 mm

5.3 Seamless split of a long image

To split a long image across multiple jobs with no gaps: **all middle pages** use `cut_paper = 0` and `serial_paper_img_print_no_tail_feed = 1`; the **last page** uses `cut_paper = 1`.

6 · Setting Commands

Format constraints

Setting commands are **plain text, one per line**, max **128 characters** per line. A malformed line is ignored, **and subsequent print processing is dropped** — always send one command at a time.

6.1 Heartbeat · SET HEARTBEAT N

- `N` is the interval in seconds: `10-3600`, or `0`. `N = 0` disables heartbeats; **heartbeats are off by default**.
- Note:** publishing any job result resets the heartbeat timer — next heartbeat = now + N seconds.

Full packet: see §2.3 example B.

6.2 Remote reboot

```
07 00 + 唯一单据号 + 00 + REBOOT PRINTER
```

With ticket ID `001`, the full hex is:

```
07 00 30 30 31 00 52 45 42 4F 4F 54 20 50 52 49 4E 54 45 52
```

Use a fresh ticket ID every time

The printer **ignores** repeated ticket IDs. In production, generate the ID from a timestamp or UUID.

6.3 FTP update · UPDATE FTP

```
UPDATE FTP "serverName",port,"userName","pwd","fileName"
```

<code>serverName</code>	FTP host IP or domain
<code>port</code>	FTP port
<code>userName</code>	Login user; use <code>anonymous</code> for anonymous
<code>pwd</code>	Password; empty allowed
<code>fileName</code>	Firmware filename

Example: FTP host `27.159.89.242`, port `21`, anonymous login, firmware

`KP202_2.4(750,150,128)_GD_MQTT_GPRS_1.07.bin` — full packet hex:

07003230313731303037313130370055504441544520465450202232372E3135392E38392E323432222C32312C220

7 · QoS & Reliability

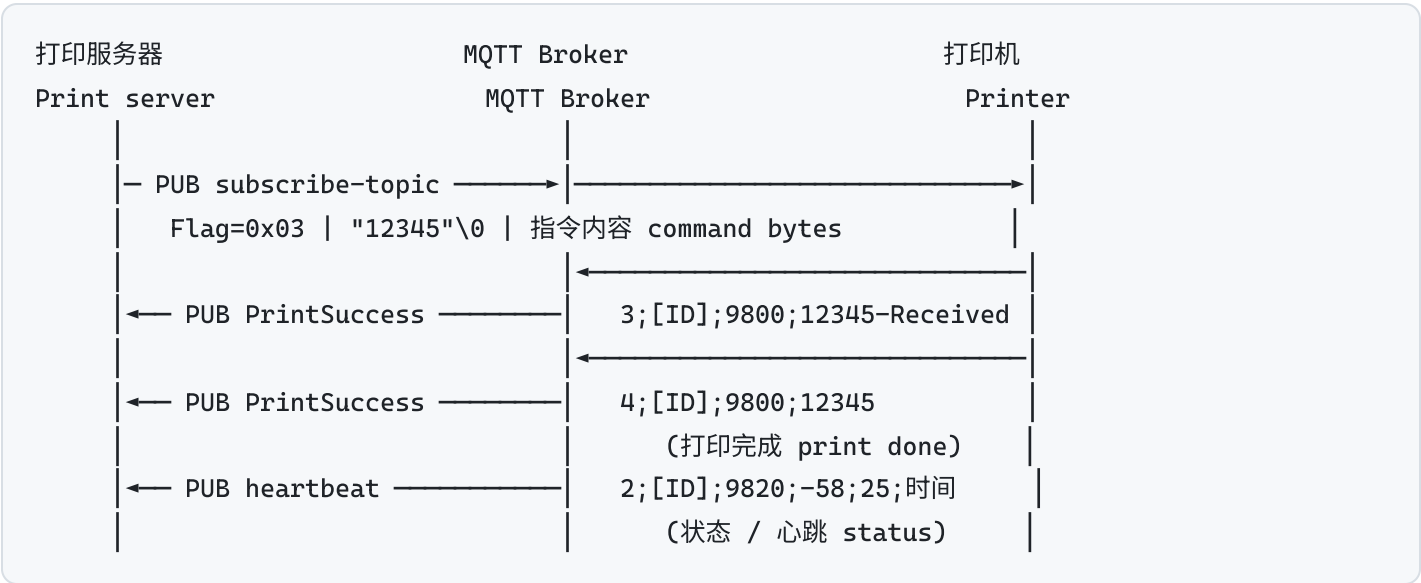
Direction	QoS requirement
To printer (downlink)	Must be 2 ; may be 1 when using a standard cloud-IoT platform. Any other level is not processed by the printer.
From printer (uplink)	1 or 2, default 1; fixed at 1 on standard cloud-IoT platforms.

Why the ticket ID is mandatory in practice

Even with QoS 2, reconnects, broker restarts and application-level retries can deliver the same job twice. The printer de-duplicates by ticket ID: when two different payloads share an ID, the later one is dropped and type 8 is published. So **ticket IDs must be globally unique**, and **when retrying a job reuse the same ticket ID** (the duplicate is discarded automatically instead of printing twice).

8 · Integration Flow

8.1 Job flow



Which topic carries what

Status messages (type 0 offline / 1 online / 2 heartbeat / 7 status change) go to `heartbeat`; **job-result** messages (type 3 received / 4 printed / 5 expired / 8 duplicate ID / 9 unparsable) go to `PrintSuccess`.

8.2 Checking printer status

- 1 First subscribe to both `heartbeat` and `PrintSuccess`.
- 2 Publish `01 00` (hex) to the printer's subscribe topic.
- 3 The printer reports its status, e.g. `4;[Prn...];9800` — receiving it confirms the printer is online.
- 4 If nothing arrives for a long period, treat the printer as offline.

8.3 Submitting print jobs

- 1 Publish the content to the subscribe topic; include a ticket ID whenever you need confirmation.
- 2 After printing, the printer decides whether to report, based on the Flag byte.
- 3 If reporting is requested, the printer publishes the ticket ID to the specified topic.

You may publish several jobs back-to-back; the printer queues them and reports results **in order**.

9 · Code Samples

Reference implementation

Reference implementations built strictly from this document. They **omit production concerns such as retries, reconnect handling and logging** — always validate against a real device. On first integration, check byte order and field order against the sample packets in §2.3 / §3.1.

9.1 Command printing (Python)

```

import paho.mqtt.client as mqtt, uuid, struct, time

TOPIC = "Prn60A44CAB3AA72020170420170648017920201704201706480179"

def build(flag: int, ticket: str, content: bytes, reply_topic: str = "") -> bytes:
    out = bytes([flag])
    out += reply_topic.encode() + b"\x00"
    out += ticket.encode() + b"\x00"
    out += content
    assert len(out) <= 16378, "packet too large"
    return out

# 普通打印：带单据号 + 要回执
pkt = build(0x01 | 0x02, str(uuid.uuid4()), b"Hello, World!\r\n")

# 带超时：时间戳放内容首部，前后 0x06 包裹，小端
expiry = int(time.time()) + 300
pkt = build(0x01 | 0x02 | 0x08, str(uuid.uuid4()),
            b"\x06" + struct.pack("<I", expiry) + b"\x06" + b"Hello, World!\r\n")

c = mqtt.Client(protocol=mqtt.MQTTv311)
c.username_pw_set("user", "pass")
c.connect("broker.example.com", 1883, 60)
c.publish(TOPIC, pkt, qos=2)          # 下发 QoS 必须为 2

```

9.2 PDF / image printing (Python)

```

import base64, json

payload = {
    "ticket_id":    str(uuid.uuid4()),          # 必须唯一 must be unique
    "data_type":    "pdf",                      # pdf | png | bmp
    "data_base64":  base64.b64encode(open("receipt.pdf", "rb").read()).decode(),
    "paper_type":   0,
    "paper_width_mm": 0,
    "paper_height_mm": 0,
    "cut_paper":    1,
    "serial_paper_img_print_no_tail_feed": 0,
}
body = json.dumps(payload, separators=(",", ":")).encode()
assert len(body) <= 2 * 1024 * 1024           # 整包 ≤ 2 MB

# 注意: JSON 下发不加 Flag 字节, 整包就是 JSON (厂家 2026-10-07 确认)
# Note: no Flag byte for JSON payloads – the body IS the JSON (vendor-confirmed 2026-10-07)
c.publish(TOPIC, body, qos=2)
c.subscribe("heartbeat"); c.subscribe("PrintSuccess")  # 两个默认上报主题都要订阅

```

9.3 Building the JSON (Java, from vendor source)

```

JSONObject json = new JSONObject();
json.put("ticket_id", TicketID);
json.put("data_type", final_data_type);
json.put("data_base64", final_data_base64);
json.put("paper_type", paper_type);
json.put("paper_width_mm", paper_width_mm);
json.put("paper_height_mm", paper_height_mm);
json.put("cut_paper", cut_paper);
String jsonString = json.toString();

```

10 · Troubleshooting

10.1 Printer cannot reach the broker

1. **Print a self-test page:** power off, hold FEED while powering on. Verify the MQTT broker address and port printed on it.
2. With Ethernet or Wi-Fi attached, the printer prints its IP within 2 minutes. If nothing prints, check cabling / Wi-Fi config and involve the network admin. GPRS-capable models also print GPRS status.

3. **Uplink priority: Ethernet > Wi-Fi > GPRS.** If the local LAN has no internet, unplug the cable to force GPRS.
4. The power LED blinks while connecting and stays solid once connected.

10.2 Error LED blink codes (GPRS not yet connected)

Blinks	Module state	
1	1	Network up, connecting to server
2	2	Attempting to dial up
3	3	Fetching network time
4	4	Reading signal strength
5	5	Reading SIM IMSI
6	6	Reading module IMEI
7	7	Initialising module

Each burst is followed by a 2-second pause — count the blinks. GPRS auto-redials after a drop; if dialing fails, the module is restarted and re-initialised.

10.3 Common "nothing printed" causes

Symptom	Cause
Nothing at all	Downlink QoS is not 2 (1 on standard IoT platforms)
Type 8 received	Duplicate ticket ID
Type 5 received	Job expired (if the timestamp is malformed you get no type 5 — it just prints)
Type 9 received	PDF/image cannot be parsed, usually the 8 MB pixel budget
PDF silently ignored	MQTT message over 2 MB, or the model does not support PDF (e.g. HS-M80)

11 · Appendix: FAQ & Support

11.1 FAQ

Question	Answer
Does the vendor provide an MQTT broker?	No. The broker is deployed by the customer / integrator and must support MQTT v3.1.1. See §1.3 for setup.
Is TLS supported?	Yes. Broker-side configuration is in §1.3 step ④; enabling it on the printer is in §1.7.
Which models print PDF / images directly?	HS-830 and HS-K39; HS-M80 is command-printing only. See §0.
Can it print labels?	HS-K39 supports TSPL / JPL / ZPL label commands (examples in §2.4). You can also render labels to PNG / PDF and use the JSON path in §5.
How do I configure the broker on the printer?	Connect the printer by USB, open printersetting.exe, enter parameter setting → MQTT tab, and fill in address, port, credentials, TLS etc. See §1.7.
Is there a ready-made debugging tool?	The vendor supplies the HSPOS MQTT PrinterTest desktop tool; usage is in §1.9. WiFi configuration is in §1.8.
How do I configure WiFi?	Use printersettingtools.exe on Windows, Printer Settings Tool on iOS and the HSPOS app on Android; see §1.8.
How do I turn off voice prompts?	Use printersetting.exe to enter parameter setting and delete the unwanted voice files from the list. See §12.
The printer never connects?	Follow §10.1: print the self-test page and verify the broker address and port. On mosquitto 2.x make sure <code>listener</code> and <code>password_file</code> are set explicitly (§1.3).
Nothing prints?	Check §10.3: QoS must be 2, the ticket ID must be unique, and the 2 MB / 8 MB limits must not be exceeded. With label commands also check the terminator (<code>PRINT</code> / <code>^XZ</code>).

11.2 Integration checklist

1. Deploy the MQTT broker (Linux + mosquitto, §1.3); confirm MQTT v3.1.1 and downlink QoS 2.
2. Write the broker address / port / credentials (and TLS) into the printer with the config tool and verify on the self-test page (§1.7).
3. Scan each printer's CPUUID and build a "CPUUID → store / station" mapping; confirm every client ID is unique.
4. Subscribe to `heartbeat` and `PrintSuccess`; power on and confirm the online message (type 1) arrives.
5. Send `01 00` for a status query, then a real job to confirm paper is printed. On first integration, use the §1.9 test tool to verify the path.
6. Implement the §7 QoS and ticket-ID de-duplication policy (reuse the original ticket ID when retrying).

11.3 Support

Contact

Support e-mail: _____ · Phone: _____ · Hours: _____

When reporting an issue, include: model, firmware version (**VER** on the self-test page or in the online message), CPUUID, broker type and version, and a broker-side capture of the topics.

11.4 Revision history

版本 / Version	日期 / Date	Notes
V1.0	2026-10-07	First release. Topics, downlink framing, uplink messages, status word, PDF/image printing, label commands, setting commands, self-hosted broker deployment, MQTT config tool / WiFi config / test tool / voice settings.

12 · Voice settings

Models such as HS-M80 / HS-830 that support voice prompts can have their built-in voice files downloaded or removed via the **printersetting.exe** tool. The steps below come from the vendor document "How to setting the voice for printer".

- 1 Connect the printer to the PC with a USB cable and open the **printersetting.exe** tool.
- 2 Select the USB port and click **Enter para setting** to enter the parameter page.
- 3 In the voice list, delete the files you do not need; check the ones you want to keep and click **downloadcheckedvoice**. The tool will close automatically and the changes take effect.

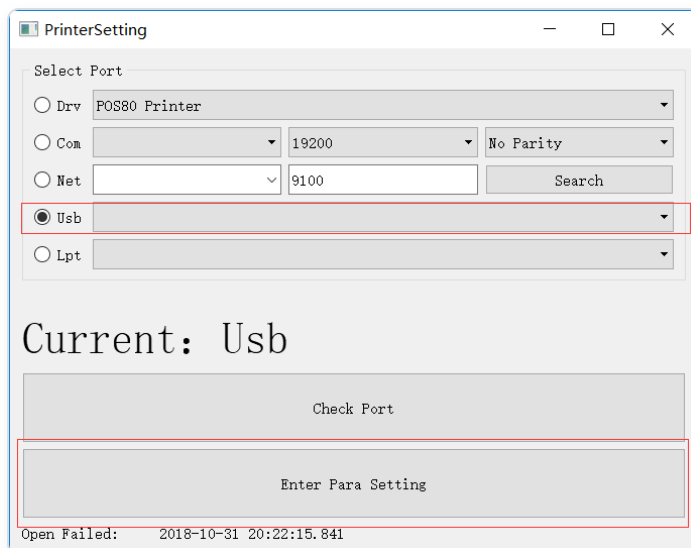


Figure 3 · Open printersetting and enter parameter setting

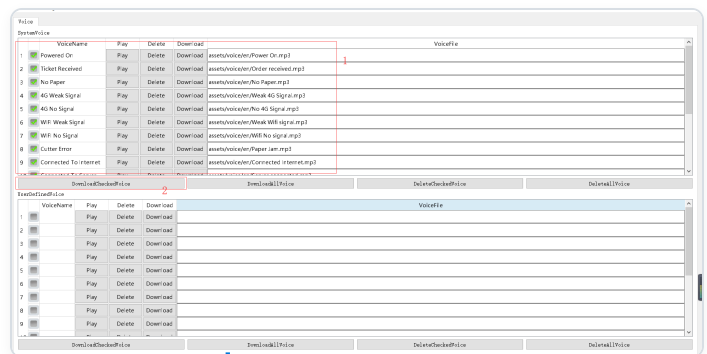


Figure 4 · Check required voices and download

Voice settings do not affect MQTT printing

Removing voice files only disables beeps or spoken prompts. It does not change the printer's MQTT topic, broker connection or printing capability.

Cloud Printer MQTT Communication Protocol V1.0 · 2026-10-07 · Models HS-M80 / HS-830 / HS-K39. Contents and samples are subject to verification against real hardware; contact technical support with any question.