

PrnAsk 列印字型讀取懸案

System.ini 的兩大 hb_iniRead 陷阱:0x1A 截斷 · 值含 [...] 誤判區段
2026-07-22 HBSTK/compat/prn_to_pdf.prg

一、訴求與現象

客戶端工作站列印/預覽, 字型設定「調了沒反應」: 設列印字型黑體仍是細明體; 'GDI_FONT_ADJ' / 'GDI_ASCII_SCALE' (字高/ASCII 縮放) 怎麼調都無效。

最弔詭的三點, 讓人完全摸不著頭緒:

- 主機 S001 正常, 所有工作站(S002~S012)都怪。
- 同一支 exe(各站 PrnAsk.exe 時間戳、大小、內含字串完全相同)。
- 讀的是同一個檔——S001 的 W:\S001 == C:\STK\S001

是同一實體, 清乾淨後仍失敗。

二、偵錯歷程(一路被誤導的懸疑)

這案最難的是「看似合理的假設一個個成立又被推翻」。完整心路:

#	當時的假設	怎麼被推翻
1	exe 目錄有 prnask.ini 蓋掉 System.ini	全站掃描, 根本沒有 prnask.ini
2	key 放在錯的 [區段](非 MAIN)	解析確認 GDI 鍵在 [FontMap] 之前=MAIN, 位置對
3	行尾 LF 讓 hb_iniRead 出錯	開發機也是 LF 卻正常, 排除
4	★ 0x1A(Ctrl-Z/DOS EOF)截斷其後區段	真的有! 清掉並加 IniReadSafe。但 S001 清完仍失敗→非唯一因
5	PrnAsk 讀本機 C:\STK\S00X 舊檔, 非網路 w:\s00X	加 path log 後證實: 讀的就是 w:\s004(對的檔)
6	PreviewMode(1=GDI/2=WebView 2 pdfjs)不同	兩站都 PreviewMode=1, 排除
7	FontDirect(2 vs 3)擋掉套用	FontDirect=3 只載 USRFONT 給 PUA 造字, 不碰主字型, 排除
8	W→C 同步清單缺字型鍵→本機永遠舊	app 其實讀網路 w:(靠 /LOCSYS), 非主因

三、破案的兩個關鍵手法

手法 A:加 path log 印出「實際讀的檔全路徑」。在 CJKDefaultFont/LoadGdiAsciiAdj 加 'WriteDebugLog(cIni)', log 印出 'INIPATH: [W:\S004\System.ini]' —— 證實「讀對檔、卻讀到 GDI_FONT_ADJ=0」,把問題從「讀哪個檔」鎖進「hb_iniRead 怎麼解析這個檔」。

手法 B:用最小測試程式(test_ini.prg)把該檔餵給 hb_iniRead, dump 出所有 section。這一步直接看到真相:

```
Top-level section ■■■■■■
[MAIN] (19 keys)
[FILE.PRN] /CALLER=HDOS4 /P /ZN='[PRINTER] ← ■■■■■■ section!
[FILE.PRN] /CALLER=HDOS4 /ZN='[PRINTER] (13 keys) ← ■■■■
[FONTMAP] / [OCR] / [FONT]
NOT FOUND: GDI_FONT_ADJ
```

四、真兇(root cause)

System.ini 裡 'PNTGO_CMD_PRINT' / 'PNTGO_CMD_PREVIEW' 的「值」含 '[file.prn]'、'[printer]' (命令列佔位符), 被 hb_iniRead 誤判成 '[區段]' 標頭。

這兩行在 88-89 行, 而 'GDI_FONT_ADJ' 在 96 行(它們後面)→ GDI_FONT_ADJ 掉進假 section '[FILE.PRN]...'、不在 MAIN → 程式 'hIni["MAIN"]["GDI_FONT_ADJ"]' 找不到 → 用預設 0。CJK_DEFAULT 因在明確的 '[Font]' 區段所以還讀得到(黑體會出), 但字高調整就失效。

五、為什麼主機 S001 正常

S001/開發機 build 的 System.ini, 'PNTGO_CMD' 那兩行剛好排在 'GDI_FONT_ADJ' 後面, 所以 GDI 鍵還在假 section 之前、仍落在 MAIN → 讀得到。純粹是 key 排列順序的運氣——這也是為什麼「把 S001 的 System.ini 複製到工作站就好了」(順序被換了), 但那只是繞過、不是根治。

六、解決方法

根治在讀取端 'IniReadSafe()' (prn_to_pdf.prg), hb_iniRead 前逐行淨化, 一次免疫兩個陷阱:

1. 'StrTran(cBuf, Chr(26), "")' — 剝掉 0x1A(陷阱 A)。
2. 逐行: 'LTrim(cLine)' 以 '[' 開頭才是真 section 原樣保留; 其餘行(key=value/註解)移除 '[' 和 ']' (陷阱 B)。
3. 寫 TEMP 暫存檔再 'hb_iniRead'。IniReadSafe 只供讀字型鍵(值不含 '['), 移除方括號無副作用。

取代 CJKDefaultFont / LoadFontMap / LoadGdiAsciiAdj 的 3 處
hb_iniRead;prn_to_pdf 編進 stk32/prnask/prnpdf,PrnAsk+Stk32
都重編。另 INI_Editor 載入時也剝
0x1A(阻止再散播;它原本逐行原封寫回=永久保留 0x1A)。

驗證：用原始問題檔(PNTGO_CMD 在前),新 PrnAsk 的 log 由
‘GDI_FONT_ADJ=0’ 變 ‘GDI_FONT_ADJ=5 / GDI_ASCII_SCALE=1.50’ →
不用調順序就正確。

七、使用技術與工具

技術/工具	用途
WriteDebugLog(cIni) path log	印出實際讀的 System.ini 全路徑,把「讀哪個檔」與「怎麼解析」分離
test_ini.prg + hb_HKeys dump	最小測試把檔餵 hb_iniRead,列出所有 section→假 section 現形
Python bytes 逐 key diff / section 模擬	比對好壞站、模擬淨化後 GDI 鍵是否回 MAIN(改碼前先驗邏輯)
xxd / cat -A	看 raw bytes 抓 0x1A、值裡的 []
IniReadSafe(TEMP 暫存 + 逐行淨化)	讀取端免疫,不改原檔、不動 PNTGO_CMD 實際運作

八、心法與部署

心法：「同一支 exe、讀同一個檔、卻讀不到某 key」→ 別再猜設定,直接
(1) 加 path log 確認讀哪個檔、(2) 用最小程式把該檔餵給同一個解析函式
dump 結構。INI 問題先看 raw bytes 與值裡的 ‘[’
‘]’,不要只看文字。A站正常B站不正常且同版 exe→逐 key diff 並注意 key
順序。

部署：新 PrnAsk.exe + Stk32.exe(含 IniReadSafe)走 uploader EXE
群組派送各站 → 各站原始 System.ini 直接正常,不用逐站改順序。新
INI_Editor.exe(剝 0x1A)派送 tools。