

PrnAsk 優化整合報告

日期：2026-05-27 | 共 6 項優化 + 1 項網路慢診斷

一、本次優化概要

編號	優化項目	影響
1	GDI 預覽 ESC + 滑鼠右鍵離開	使用者操作便利性
2	FwGetPrnLocation 永久快取	網路印表機第二次起秒返
3	GDI 列印對話框延遲 FwGetPrnLocation	對話框立刻完整顯示
4	FwPrintAsk PreviewMode 分支優化	省 4.2 秒 PRN→PDF（特定設定）
5	log 改寫 DEBUG_PRNASK.TXT	與使用者批次結果檔分離
6	統一 WriteDebugLog 到 HBSTK/compat	全專案一份；修正 Stk32 端失效
7（待）	客戶網路 EXE 5 秒慢	待協調部署方案

二、GDI 預覽 ESC + 滑鼠右鍵離開

訴求：客戶反映 GDI 預覽視窗無法按 ESC 離開，必須點關閉鈕。

解法：FwPrnPreviewGDI 視窗加 bKeyChar + bRClicked：

```
oWnd:bKeyChar := {|nKey| IIF(nKey == 27, oWnd:End(), NIL) }  
oWnd:bRClicked := {|| oWnd:End() }
```

三、FwGetPrnLocation 永久快取

訴求：每次開列印對話框，FwGetPrnLocation 對網路印表機要等 1.13 秒（本機）／更久（客戶機器）。

根因：每次呼叫都跑 pythonw + Win API 查印表機 Location 屬性，單次查詢無法加速（pythonw 啟動成本固定）。

解法：內部加 STATIC hash 快取

```
STATIC s_hPrnLocCache := NIL  
...  
IF hb_HHasKey( s_hPrnLocCache, cPrinter )  
    RETURN s_hPrnLocCache[cPrinter] // ■■  
ENDIF  
... pythonw ■■ ...  
s_hPrnLocCache[cPrinter] := cLoc // ■■■■
```

效果：

次數	結果
第 1 次同台印表機	1~3 秒（pythonw 成本，無解）
第 2 次起同台印表機	瞬間返回
不同印表機	各自第一次仍慢，之後都快

四、GDI 列印對話框延遲 FwGetPrnLocation

訴求：FwGdiPrintDialog 第一次顯示時凍結 1~5 秒，因為 ACTIVATE DIALOG 前同步呼叫 FwGetPrnLocation。

踩坑（三次嘗試）：

嘗試	方法	結果
1	放在 oDlg:bStart codeblock	失敗：bStart 阻塞 WM_PAINT，外框顯示但內部空白
2	oDlg:SetTimer + bTimer	失敗：TDialog 沒有 bTimer message，跑出 error BASE/1005
3	oDlg:bPainted + 一次性旗標	成功：bPainted 在對話框完整繪製後才觸發

最終解法：

```
LOCAL lLocDone := .F.
...
oDlg:bPainted := {|hDC| IIF(!lLocDone, ( ;
    lLocDone := .T., ;
    IIF(oSayLoc != NIL, oSayLoc:SetText(FwGetPrnLocation(cPrinter)), NIL) ;
), NIL) }
```

效果：對話框立刻完整顯示，Location 文字稍後填上，搭配快取後第二次起秒填。

五、FwPrintAsk PreviewMode 分支優化 (4.2 秒)

訴求：PrnAsk 自動預覽模式 (PRN_SKIPASK=1 + PRN_AUTOFMT=4 + PreviewMode=1 GDI) 下，FwPrintAsk 浪費 4.2 秒做 PRN→PDF 轉換，但 GDI 預覽根本不用 PDF。

Timing log 證據：

```
T4 FwPrintAsk entry      ms=X
T5 FwPrnPreview entry    ms=X+4227 ← 4.2 ■
T6 FwPrnPreviewGDI entry ms=X+4227
T7 before GdiPreviewPrn ms=X+4227
T8 after GdiPreviewPrn  ms=X+4362 (PRN→BMP ■■ 135ms)
```

4.2 秒 = FwPrnToPdf 內部的 ADVANCELINE GDI 渲染。GDI 路徑根本不用該 PDF。

解法 (在 CASE PRN_OUT_PREVIEW 加 PreviewMode 分支)：

```
CASE nAutoFmt == PRN_OUT_PREVIEW
  IF Val(FwReadIniKey("PreviewMode")) == 1
    // GDI ■■, ■■ FwPrnToPdf
    FwPrnPreviewGDI( cPrnFile, "" )
  ELSE
    // WebView2 ■■ PDF
    cPdfFile := FwPrnToPdf( ... )
    IF !Empty(cPdfFile) ; FwPrnPreview(...) ; ENDIF
  ENDIF
```

注意：此優化僅在 PRN_SKIPASK=1 路徑生效。若客戶設 PRN_SKIPASK=0 (顯示對話框路徑)，會走 FwPrintOutput，不會經過此 CASE。

六、RESULT.TXT → DEBUG_PRNASK.TXT

訴求：PrnAsk 的 debug log 寫到 RESULT.TXT，但 RESULT.TXT 也是使用者批次處理結果檔，兩者混淆。

解法：fw_prnout.prg 內 WriteDebugLog 與 FwOpenResult 統一改寫 DEBUG_PRNASK.TXT。

後續：WriteDebugLog 整段在 15:30 統一到 HBSTK/compat/writedebuglog.prg（見下節）。

七、統一 WriteDebugLog（全專案唯一一份）

訴求：ConvertFW 專案內 WriteDebugLog 定義散落 3 處：

位置	用途	狀態
HBSTK/compat/clipper_compat.prg	Stk32 用	失效（讀錯 INI）
PRNASK/source/fw_prnout.prg	PrnAsk 用	正常但只在 PrnAsk
PRNASK/source/debug_log.prg	無人呼叫	dead code

Stk32 端 WriteDebugLog 失效真兇：

```
clipper_compat.prg::WriteDebugLog ■ GetIniValue("DEBUG_LOG") ■■■■■■  
■ GetIniValue ■ System.ini■  
■ DEBUG_LOG=2 ■■ stk.ini■  
GetIniValue ■■■■ "" → Val("") = 0 → return NIL → 79 ■ WriteDebugLog ■■■■
```

解法：

1. 新檔 HBSTK/compat/writedebuglog.prg（唯一一份）
2. 內部用 WdlReadDebugLevel() 直接讀 stk.ini，繞過 GetIniValue
3. 依 hb_ProgName() 自動選輸出檔（Stk32 / PrnAsk）
4. 移除 clipper_compat.prg / fw_prnout.prg 的舊定義
5. PRNASK/debug_log.prg 改名為 .bak_unified_20260527
6. stk32.hbp + prnask.hbp 都加入新檔

踩坑：

1. 第一次正規表達式 `.*?RETURN NIL` 吃到函式內部的 `RETURN NIL`，把後半留下 → `ENDIF` 不對稱
2. `bPainted codeblock` 移除 `WriteDebugLog` 後留下尾逗號 → `syntax error`
3. `.hbp` 檔案 `LF/CRLF` 混用，要分別處理

八、客戶網路 EXE 5 秒慢診斷（待實作）

訴求：客戶反映 Stk32 按列印鈕到 PrnAsk UI 出現要 5 秒，本機只要 0.2 秒。

完整時間軸分析：

階段	本機（SSD）	客戶（W:\S007 網路）	比例
Stk32 SpoolClose → hb_processOpen	9 ms	~9 ms	小
OS 載入 PrnAsk + DLL + Harbour init	174 ms	~5 秒	主要
PrnAsk Main → 對話框顯示	29 ms	~29 ms	小
合計	212 ms	~5 秒	

根因：客戶 Stk32.exe + PrnAsk.exe 都在 W:\S007\ 網路磁碟，Windows 從網路載入 EXE + DLL + BCACHE + Resource 累積延遲。本機 SSD 載入快 25 倍。

待選優化方案：

方案	描述	工程量	效果
A 預熱	Stk32 啟動時偷跑一次 PrnAsk（隱藏視窗）讓 Windows file cache 命中	小	第一次後快
B 常駐	PrnAsk 變 server mode，IPC 通訊；不重啟	大	永遠 0 秒
C 部署	EXE 改放客戶本機 C:\STK\，DBF/設定仍在網路	0 code， 需協調	根本解

狀態：待跟客戶討論決定方向後實作。

九、使用技術與方法

技術	用途
Python bytes-only 修改 BIG5 PRG	註解直接 .encode("big5")，繞過 Edit tool 的中文亂碼

Harbour timing log + hb_MilliSeconds()	量測各階段耗時，精準鎖定瓶頸
對話框 bPainted 一次性旗標	對話框完整繪製後才執行延遲操作
STATIC hash 快取 (hb_HHasKey)	同一鍵值只查一次，後續秒返
INI 解析 At() + SubStr()	繞過共用 GetIniValue 讀錯檔案的問題
hb_ProgName() 自動偵測	WriteDebugLog 依 EXE 自動選輸出檔