

Commander SCREEN 截圖改良 (DPI-aware 全螢幕 + 無閃爍背景執行)

日期：2026/06/03 模組：STK Agent (SOCKET/StkAgent)

檔案：source/stk_agent.prg、tools/screenshot.c、stk_agent.manifest/rc (新增)

一、訴求與現象

Commander 發 SCREEN 指令給遠端 Agent 截圖回傳。客戶反映：截到的常常只是畫面**左上一小塊** (不是整個畫面)，且遠端截圖當下會**閃過一個 PowerShell 視窗**。需求： 截到完整、清晰、跨所有螢幕的畫面； 背景默默執行、完全不閃爍。

二、踩坑與陷阱 (找錯地方改了三輪)

#	坑	原因 / 說明
1	誤判為 C 截圖的 DPI 問題	Agent 有 C 函式 CaptureScreenshot (GetSystemMetrics+BitBlt)，直覺認定是它 DPI-unaware → 改用 virtual screen，並在 Main() 開頭加 SetProcessDPIAware。部署後仍一小塊。
2	Main 早呼叫 SetProcessDPIAware 也沒用	Harbour 的 -gtwin 在 Main() 執行前就初始化 GT (terminal window)，此時 DPI awareness 已被系統鎖定；Main 第一行的 SetProcessDPIAware 太晚，回 FALSE。
3	manifest 嵌入也救不了 SCREEN	改用 application manifest (dpiAware/PerMonitorV2) 嵌進 exe，loader 層級最早設定。驗證 exe 確實含 dpiAware=True，但 SCREEN 截圖**還是一小塊**！
4	★真正根因：改錯地方了	CMD_SCREENSHOT 指令根本不是用 C CaptureScreenshot，而是另一段「PowerShell 腳本」(hb_MemoWrit 寫 _capture.ps1 再執行)。前面 C / manifest 三輪改動全部沒碰到真正被呼叫的程式碼。

#	坑	原因 / 說明
5	那段 PowerShell 的雙重缺陷	完全沒有 SetProcessDPIAware → 高 DPI 下回邏輯像素、CopyFromScreen 只抓物理畫面左上 1/4；用 PrimaryScreen → 只截主螢幕，多螢幕抓不到。
6	PS 閃爍：Hidden 不夠	腳本本來就帶 -WindowStyle Hidden，但 powershell.exe 仍會短暫建立 console host (conhost.exe) → 畫面閃一下。

三、解決方法

截到完整畫面 —— 修真正被呼叫的 PowerShell 腳本 (_capture.psl)：

參照 STK32 已驗證的 ERRORSYS 做法：腳本開頭加 SetProcessDPIAware() (DllImport user32)。關鍵：PowerShell 是獨立子 process，啟動時還沒做任何 screen 操作，此時呼叫 SetProcessDPIAware 一定生效（不像 Agent 主程式被 -gtwin 鎖定）。再把 PrimaryScreen 換成 SystemInformation::VirtualScreen，CopyFromScreen 從 virtual origin(\$v.X,\$v.Y) 抓，一次涵蓋所有螢幕的物理全解析度。

無閃爍、背景默默 —— 新增 C 函式 RUNHIDDEN：

用 CreateProcess + CREATE_NO_WINDOW 啟動 PowerShell —— 連 console 視窗都不建立，從根本消除閃爍（-WindowStyle Hidden 只是隱藏 PS 視窗，conhost 仍會閃）。Agent 改用 RunHidden() 取代 hb_processRun() 來跑 _capture.psl，等它完成後讀回 JPG。

本地驗證：實測腳本 VirtualScreen 1920x1080 → 輸出 JPG 1920x1080 完整畫面。

四、使用技術與用法

技術	用法 / 重點
DPI awareness 生效時機	程式碼 SetProcessDPIAware() (GT/window 初始化後 = 太晚) < application manifest (loader 最早，唯一能趕在 -gtwin 前)。但獨立子 process 自己呼叫 SetProcessDPIAware 不受父程式限制 → PowerShell 截圖最省事。

技術	用法 / 重點
VirtualScreen vs PrimaryScreen	SystemInformation::VirtualScreen 涵蓋所有螢幕 (X/Y 可為負) ; CopyFromScreen(\$v.X, \$v.Y, 0, 0, \$v.Size) 抓全部。PrimaryScreen 只主螢幕。
CREATE_NO_WINDOW	優於 -WindowStyle Hidden: 後者 powershell 仍建 conhost 短暫閃現; 前者連 console 都不給, 真正背景無痕。常數 0x08000000。
RUNHIDDEN 用法	RunHidden(cCmdLine [, nTimeoutMs]); 內部 CreateProcessA + STARTF_USESHOWWINDOW/SW_HIDE + CREATE_NO_WINDOW + WaitForSingleObject。在 tools/screenshot.c。
診斷教訓	改 bug 前先確認「真正被執行的是哪段程式碼」。本案 SCREEN 用 PowerShell 卻先去改 C, 白繞三輪。第一步應 grep 指令處理流程 (CASE cCmd==CMD_SCREENSHOT) 看它實際呼叫什麼。

五、最終成果

SCREEN 指令現在截到完整、物理解析度、跨所有螢幕的畫面, 且 PowerShell 完全在背景執行、無任何視窗閃爍。

保留的備援改動: application manifest (dpiAware/PerMonitorV2) 與 C CaptureScreenshot 的 virtual screen 改動仍保留 —— 它們讓 ERROR_REPORT 走的 C 截圖路徑同樣受益 (DPI-aware+多螢幕), 無害。

最大教訓: 同一支程式可能有「兩套截圖實作」(C 與 PowerShell), 不同指令走不同路徑。修 bug 必先確認目標指令實際呼叫哪一套, 否則改了沒被執行的程式碼, 再多輪也是徒勞。