

STK 自動錯誤回報系統

截圖 + 元資料 + Agent 上傳 FTP + 剪貼簿
完整踩坑記錄與解決方案
2026-05-28

一、目標與訴求

原 Clipper 時代設計：客戶端 STK 出錯 → 寫 Error.TXT → 顯示「請傳真到 (02)8209-2893」→ 客戶自己手寫傳真。30 年前合理，現在客戶都不裝傳真機；即使裝了，畫面早被使用者按掉，看不出當時操作了什麼。

新目標：

#	需求	達成方式
1	出錯瞬間自動截圖	同步 PowerShell 截 _before.bmp (錯誤前) + 非同步截 _after.bmp (含 MessageBox)
2	錯誤檔自動上傳到 FTP	Agent 2 秒掃 trigger 檔 → 自動上傳 /H/stk-error/{KMSN}/{S00X}/
3	客戶端零設定	HBSTK 端用 hb_CurDrive()+CurDir() 推導路徑，不必改 stk.ini
4	手動觸發版 (畫面怪但無錯誤)	Ctrl + Keypad+ → 截圖 + 寫檔 + 上傳 + 內容塞剪貼簿
5	錯誤檔含完整環境資訊	KMSN/STATION/COMPUTER/USER/SUBSYS/CODE/DESC/OP/PRO C/STACK + 公司資訊

二、完整資料流向

兩個觸發源（自動 DefError / 手動 Ctrl+Keypad+）→ 寫入本地 error_pending\ → Agent 主 loop 2 秒掃描 → 上傳 FTP → 成功後刪除本地。

觸發 → 寫檔 → 上傳 全部步驟：

階段	動作	產生檔案	注意
[A]	錯誤拋出 → ErrorBlock 觸發 DefError	—	防遞迴 s_lInError
[B]	PowerShell 同步截圖 (hb_run 阻塞 ~300ms)	_before.bmp	搶在 MessageBox 跳出前
[C]	寫 WriteDebugLog (ERROR + STACK)	Debug_Stk32.txt	原本就有
[D]	特殊 case 過濾 (EG_ZERODIV/BOUND/DATAWIDTH ...)	—	可能提早 RETURN
[E]	組 cMessage + aOptions	—	—
[F]	非同步啟動 _after.bmp 截圖 (cmd /c start /B)	_after.bmp (背景)	500ms 後拍含 MessageBox 畫面
[G]	MessageBox 跳出 → 使用者按 Quit/Default/Retry	—	modal 阻塞
[H]	SET ALTERNATE → ErrorMsg.TXT (累積)	ErrorMsg.TXT	SET CONSOLE OFF 避免畫面 spam
[I]	SET ALTERNATE → cTxtFile (per-timestamp)	xxx.txt	原本是 Error.TXT，改寫到 error_pending
[J]	hb_MemoWrit → 寫 trigger (一次性)	xxx.trigger	含 FILES= 列出所有檔
[K]	QUIT 結束程式	—	—
	——— Agent 主 loop (每 2 秒 RelayRecv timeout) ———		
[L]	ProcessErrorPending() 掃 *.trigger	—	併入既有 timeout，零新增 timer

階段	動作	產生檔案	注意
[M]	ParseTriggerFile → 取出 FILES= 欄位	—	按 trigger 列舉，不寫死
[N]	FtpUploadFile → /H/stk-error/{KMSN}/{S00X}/	FTP 上	
[O]	全部成功 → FErase 本地，失敗保留下次重試	—	—

三、踩坑記錄（共 9 個雷區）

每個坑用「症狀 / 原因 / 解法」三段格式說明，給未來踩坑的人少走冤路。

坑 1: 相對路徑 `.\subdir\` 在 Harbour FCreate/hb_MemoWrit 假成功

症狀：FCreate 回傳 valid handle（如 1476）、FError() 回 0，但檔案就是不出現在磁碟。寫 7 個小時找不到原因。

原因：當 `cPendDir = ".\error_pending\"` 而 CWD 是網路磁碟（`W:\S001`）時，Harbour FCreate / hb_MemoWrit 處理「相對路徑+子目錄」的組合會 silent fail。PowerShell 寫同樣路徑卻 OK，因為 PowerShell 自己解析路徑。

解法：改用絕對路徑。HBSTK 用 `hb_CurDrive() + ":\\" + CurDir() + "\\error_pending\\"` 推導出 `W:\\S001\\error_pending\\`，所有檔案 IO 操作都用絕對路徑。

```
cPendDir := hb_CurDrive() + ":\\" + CurDir() + "\error_pending\"
```

★ 教訓：碰到網路磁碟上的子目錄，永遠用絕對路徑。

坑 2: 多個 FWrite 連續呼叫產生 0-byte 檔案

症狀：FCreate 後接 8 次 FWrite + FClose，FCreate 成功，但檔案結果是 0 bytes 上傳到 FTP。

原因：DefError 內 async PowerShell 仍在背景跑（搶磁碟），FWrite 多次連續呼叫時某次失敗、後續全部 silent fail，但 FClose 仍然完成留下 0-byte 檔。FWrite 不會 throw，回傳 -1 也沒人檢查。

解法：整個 trigger 內容組成一個大字串，用單次 hb_MemoWrit 寫入。一次性 IO 操作，沒有中段失敗的機會。

```
hb_MemoWrit( cTrgFile, "TIME=" + cTS + CRLF + "KMSN=" + ... + CRLF + ... )
```

★ 教訓：對於小檔案（< 64KB），永遠優先用 hb_MemoWrit，避免 FCreate+FWrite 連續呼叫。

坑 3: FtpUploadFile 無條件回 .T. (假成功)

症狀：Agent log 顯示 upload done，但 FTP 上沒檔案。

原因：shared/ftp_helper.prg 的 FtpUploadFile 函式在 STOR 之後 read reply 但沒檢查，無條件 RETURN .T.：

```
cReply := FtpRecv( hSock ) ← 讀完丟掉  
HB_SYMBOL_UNUSED( cReply )  
RETURN .T. ← 永遠成功
```

解法：檢查 reply 第一個字元是 "2" (226 Transfer complete) 才回 .T.：

```
cReply := FtpRecv( hSock )  
1OK := ( Left( cReply, 1 ) == "2" )  
RETURN 1OK
```

★ 教訓：第三方函式庫永遠要驗證它真的做了該做的事。

坑 4: FtpUploadFile 沒吃 150 preliminary reply

症狀：修了坑 3 後，FtpUploadFile 仍回 FAIL，但 FTP 上檔案是 OK 的。

原因：FTP 協議規定 STOR 後 server 先發 "150 Opening data connection" (preliminary)，data 傳完才發 "226 Transfer complete"。原 code 在 STOR 後不 recv，只在 close data 後 recv 一次，拿到的是 "150"。Left("150...", 1) == "1" 不等於 "2" → 回 FAIL。

解法：STOR 後立刻 recv 一次吃掉 150；data 傳完 close 後再 recv 一次拿 226：

```
FtpSend( hSock, "STOR " + cRemoteName )  
cReply := FtpRecv( hSock ) ← 吃 150  
IF Left(cReply,1) == "4" .OR. Left(cReply,1) == "5" RETURN .F. ENDIF  
... send data, close ...  
cReply := FtpRecv( hSock ) ← 真正的 226
```

★ 教訓：實作 protocol 一定要照 RFC 走，不能省 round-trip。

坑 5: Agent ProcessErrorPending 寫死檔名清單

症狀：4 個檔（_before.bmp / _after.bmp / .txt / .trigger）寫入本地，但 Agent 只上傳 trigger + .txt 兩個，BMP 留在本地永遠沒人上傳。

原因：Agent 端寫死的清單長這樣：

```
aFiles := { cTrgFile }  
AAdd( aFiles, cDir + cBase + ".bmp" ) ← 單張 .bmp  
AAdd( aFiles, cDir + cBase + ".txt" )
```

HBSTK 後來改寫 _before.bmp / _after.bmp，但 Agent 還在找單張 BASE.bmp（不存在）→ 跳過 → 不會上傳 BMP。

解法：讓 Agent 讀 trigger 內 FILES= 欄位，按那裡列舉的檔名上傳：

```
IF HB_HHasKey( hMeta, "FILES" )  
aFilesList := hb_ATokens( hMeta["FILES"], "," )  
FOR each → AAdd( aFiles, cDir + filename )  
ENDIF
```

★ 教訓：跨組件協議讓「資料」自己描述自己，不要寫死 schema。

坑 6: ErrorMessage.TXT 段 QOUT 沒 SET CONSOLE OFF → 畫面 spam

症狀：錯誤發生時螢幕被一堆公司資訊+錯誤訊息+stack trace 訊息蓋滿。

原因：原 Clipper 設計：ErrorMessage.TXT 段 QOUT 同時輸出到檔案和螢幕（給支援人員「現場看」），但現代客戶不需要看，視覺干擾大。Error.TXT 段反而有 SET CONSOLE OFF 沒這問題。

解法：在 ErrorMessage.TXT 段 QOUT 區塊前加 SET CONSOLE OFF、結尾加 SET CONSOLE ON：

```
SET ALTERNATE TO ErrorMessage.TXT ADDITIVE  
SET ALTERNATE ON  
SET CONSOLE OFF ← 新加  
QOUT( ... )  
SET CONSOLE ON ← 新加  
SET ALTERNATE OFF
```

坑 7: SET ALTERNATE 跨進程 race condition

症狀：原本想用 `cmd /c copy Error.TXT xxx.txt` 把 Error.TXT 複製成 per-timestamp，但 copy 經常 fail 或拿到 0 bytes。

原因：SET ALTERNATE → CLOSE ALTERNATE 在 Harbour 端是「呼叫 `fclose()` flush stdio buffer」，但這只保證資料推給 OS。OS page cache → SMB redirector buffer 到網路落地的延遲（特別是 W:\ 網路磁碟）可能達數百 ms。下一行的 `cmd copy` 是獨立進程透過獨立 SMB session 讀 Error.TXT，看到的可能是舊版或空檔。

解法：不要跨進程依賴剛寫的檔案。改用同進程直接寫到目標檔案：把原 Error.TXT 的 SET ALTERNATE 目標改成 (cTxtFile)，一次到位。

```
SET ALTERNATE TO (cTxtFile) ← 直接寫到 error_pending\xxx.txt
```

★ 教訓：跨進程的「先 A 寫、後 B 讀」就是 race。改成「A 直接寫到 B 要的位置」。

坑 8: KeyAltC 殭屍綁定（被下一行覆蓋）

症狀：K_ALT_6 設了 KeyAltC()，但實際按 Alt+6 觸發 PrePricelBox() 而非 KeyAltC()。KeyAltC 多年來其實沒人觸發過。

原因：APPSETUP.PRG 連續兩行 SetKey 都用 K_ALT_6：

```
SetKey( K_ALT_6, {|| KeyAltC()} ) ← 立刻被覆蓋  
SetKey( K_ALT_6, {|| PrePricelBox(...)} ) ← 實際生效
```

解法：改綁 Ctrl + Keypad+ (KP_CTRL_PLUS = 400)，跟其他按鍵不衝突，記憶上跟「增加/上傳」概念也對。

```
SetKey( KP_CTRL_PLUS, {|| KeyAltC()} )
```

★ 教訓：SetKey 重複綁定不會 warning，定期 audit。

坑 9: aExeInfo 抓 Stk.exe 拿不到時間（已不存在）

症狀：程式日期顯示 1980/01/01 或空白。

原因：STK.PRG:269 寫 `aExeInfo := Directory(cLocExe+"\Stk.exe", "D")`。但 HBSTK 編譯出來是 Stk32.exe，不是 Stk.exe。Directory() 找不到 Stk.exe 回空陣列。

解法：改抓 Stk32.exe：

```
aExeInfo := Directory(cLocExe+"\Stk32.exe", "D")
```

四、最終架構與檔案

修改的檔案：

檔案	修改內容
HBSTK/source/ERRORSYS. PRG	DefError 截圖+trigger; KeyAltC 升級截圖+trigger+clipboard; SET CONSOLE OFF; GetStackStr() helper
HBSTK/source/STK. PRG	aExeInfo 改抓 Stk32.exe
HBSTK/source/APPSETUP. PRG	KeyAltC 綁 KP_CTRL_PLUS (取代殭屍 K_ALT_6)
SOCKET/StkAgent/source/stk_agent. prg	main loop 加 ProcessErrorPending; 新增 ProcessErrorPending() + ParseTriggerFile(); FTP 路徑加 /S00X/
SOCKET/shared/ftp_helper.prg	FtpUploadFile 真檢查 STOR 回應; 吃掉 150 preliminary reply

trigger 檔內容範例：

```
TIME=20260528_164927
KMSN=9999-9999-99-99
STATION=S001
ERR_SUBSYS=DBFCDX
ERR_CODE=1021
ERR_DESC=Data width error
ERR_OP=NSUBAMOUNT
COMPUTER=S001
USER=user
PROC=QUOT(1241)
STACK=(b)ERRORSYS(31)>QUOT(1241)>(b)APPSETUP(112)>...
FILES=20260528_164927_before.bmp, 20260528_164927_after.bmp, 20260528_164927.txt
```

FTP 目錄結構：

```
/H/stk-error/{KMSN}/{S00X}/
├ 20260528_164927_before.bmp (~200 KB)
├ 20260528_164927_after.bmp (~200 KB)
├ 20260528_164927.txt (含 boxed 公司資訊+錯誤+stack)
└ 20260528_164927.trigger (元資料 ASCII)
```

五、使用方式

自動觸發（程式錯誤）：

無需操作。當 STK 拋出錯誤時：

1. 截圖（前後兩張）
2. 寫 .txt + .trigger
3. 跳 MessageBox 給使用者按 Quit
4. QUIT 結束程式
5. Agent 2 秒內自動上傳 FTP

手動觸發（畫面怪但無錯誤）：

按 Ctrl + Keypad +（數字鍵盤的 +）：

1. 截當下螢幕
2. 寫 .txt + .trigger (TYPE=MANUAL)
3. .txt 全內容已塞入剪貼簿 → 可 Ctrl+V 貼到 Email/LINE
4. 顯示資訊，按任意鍵恢復
5. Agent 自動上傳 FTP

支援人員處理流程：

1. 客戶說「STK 跳錯誤」或「畫面怪怪的」
2. 開 FTP 看 /H/stk-error/{客戶 KMSN}/{站別}/
3. 找最新 timestamp 的 4 個檔
4. _before.bmp 看出錯前操作，_after.bmp 看跳錯框內容
5. .txt 看完整錯誤訊息與 stack trace（中文版）
6. .trigger 機器可讀，含元資料（KMSN/SUBSYS/CODE/...）
7. 或直接請客戶按 Ctrl+V，Email 過來最快

後續維護要點：

- error_pending 目錄不存在會自動 hb_DirBuild 建立
- 上傳失敗的檔留在本地，下次 Agent loop 自動重試
- BMP 不會清理（讓客戶本地也有備份）；trigger+.txt 上傳成功才刪
- 注意 FTP_HOST / FTP_USER_ENC / FTP_PASS_ENC / FTP_ERROR_DIR 都在 stk_agent.ini
- KMSN 從 stk.ini 自動載入（MEMVAR->cKMSerialNo）