

STK Commander 隱形 modal 死當自癒

日期：2026-07-23 | 檔案：stk_commander_qt.py (PyQt5)

一、訴求與現象

逐筆檢視客戶傳來的 FTP 錯誤回報時，看完某一筆後，Commander 主視窗與錯誤清單視窗整個死當，點什麼都沒反應。這個問題先前已修過多次（截圖背景化、送出佇列非阻塞、凍結 watchdog），但都沒治好，因為每次修的都是「主執行緒卡死」這個型態——而這次的死當根本不是那個型態。

二、踩坑與陷阱

採證推翻了舊假設。用 py-spy 對執行中的 Commander 抓執行緒堆疊、用 EnumWindows 列出所有頂層視窗，得到三個鐵證：process 從開機起從未重啟、freeze_stacks.log 根本不存在（代表 GUI 心跳從未停過超過 10 秒）、主執行緒停在正常的事件迴圈、CPU 幾乎為零。

視窗	狀態
錯誤清單 / 主視窗	enabled = 0（被鎖住，不可互動）
某 QDialog（詳情視窗）	visible = 0（看不見）但 enabled = 1（是 modal，霸佔全部輸入）

真相：一個「看不見的 modal 對話框」鎖住了整個 App。事件迴圈還活著（所以凍結偵測器抓不到、Responding 仍是 True），但所有輸入都被這個隱形 modal 攔截，看起來就是死當。後來被 ESC／切換焦點意外解開。

修完第一版又踩退化坑。第一版讓詳情視窗「先 show() 再 exec_()」，但 Qt 對「已顯示的視窗」設定 modality 是無效的——詳情視窗因此變成非 modal，activeModalWidget() 反而回傳底下的清單視窗，守護程式每秒把清單視窗提頂一次、瘋狂搶焦點。症狀變成：點進摘要框選字，滑鼠一放開焦點就被搶走。modal_guard.log 第一次上場就當場記下這個兇手。

三、解決方法

利用「這個死當型態下唯一還活著的鉤子」——每秒 GUI 心跳 QTimer（它在 modal 巢狀事件迴圈中照樣會跳）——加上 modal guard 自癒檢查，並收斂搶焦點的邊界條件。

招式	動作
隱形 modal	第 1 秒只觀察（避免誤救正在正常關閉的視窗）；第 2~4 秒 show/raise/activateWindow 救回；仍失敗則 reject() 強制解鎖（最壞 5 秒內自解）
被壓在自家視窗後	前景視窗屬本 process、且無任何自家視窗作用中時才提頂（使用者操作中絕不出手）
popup 抓取殘留	activePopupWidget 不可見時 close() 釋放
源頭防護	_open_detail 改用 QTimer.singleShot(0, raise+activate)，在 exec_ 以 modal 正常顯示後才提頂，不破壞 modality

每次介入都寫 modal_guard.log（EXE 同目錄），留下第一手證據。

四、使用技術與用法

技術	用途
py-spy dump --pid	對執行中 exe 抓 Python 執行緒堆疊（裝在 Python312-32，dump 子 process 非 bootloader）
EnumWindows + IsWindowVisible/Enabled	列出程序所有頂層視窗的可見性與啟用狀態，揪出隱形 modal
QApplication.activeModalWidget / activePopupWidget / activeWindow	判斷當前是否有 modal / popup / 作用中視窗
QTimer.singleShot(0, ...)	延到 exec_ 進入事件迴圈後才提頂，避免破壞 modality
modal_guard.log	自癒介入留證，將來再發生可直接定位

診斷標準流程（更新）：死當時先看

freeze_stacks.log——若無新紀錄＝非執行緒卡死→改查

modal_guard.log；兩者都空，再用 py-spy dump + EnumWindows

抓隱形視窗。鐵律：guard 類自癒機制，使用者有作用中視窗時絕不出手。