```
#pragma once

#ifndef SERVER_SOCKET_H
#define SERVER_SOCKET_H

// THIS_IS_SERVICE_APP is defined in the PAM project under C++ | Preprocessor Definitions

#ifdef THIS_IS_SERVICE_APP
//#define MAIN_DLG_NAME CTscanDlg
#else
#define MAIN_DLG_NAME   CTscanDlg
#endif


#ifndef SERVER_CONNECTION_MANAGEMENT_H
#include "afxsock.h"
#include "ServerConnectionManagement.h"
#include "CCmdFifo.h"
#endif
#include "HwTimer.h"


#define MAX_PAM_BYTES   1260*8
// expected msg size is 1260*8 = 10080

// Listener thread creates and holds the eListener socker
// Permanent Server connection thread are designated as eServerConnection
// The temporary socket used by OnAccept to build the server connetion socket is the eOnStack socket
enum { eListener, eServerConnection,eOnStack };

// CServerSocket command target

class CServerConnectionManagement;
class CServerSocketOwnerThread;

//*pSCM;
class CServerSocket : public CAsyncSocket
{
public:
    CServerSocket();
    CServerSocket(CServerConnectionManagement *pSCM, int nOwningThreadType);

    virtual ~CServerSocket();
    void Init(void);
    virtual void OnAccept(int nErrorCode);
    virtual void OnReceive(int nErrorCode);
    virtual void OnClose(int nErrorCode);

    int InitListeningSocket(CServerConnectionManagement * pSCM);
    void SetSCM(CServerConnectionManagement *pSCM)      {   m_pSCM = pSCM;  }
```

```
51      CServerConnectionManagement * GetSCM(void)      {   return m_pSCM;  }
52
53      void OnAcceptInitializeConnectionStats(ST_SERVERS_CLIENT_CONNECTION *pscc,
          int nMyServer, int nClientPortIndex);
54      //void KillpClientConnectionStruct(void);
55
56      //int SendPacket(BYTE *pB, int nBytes, int nDeleteFlag);
57
58      void LockRcvPktList(void)       { if (m_pSCC) { if (m_pSCC->pCSRcvPkt)
          EnterCriticalSection(m_pSCC->pCSRcvPkt);    }   }
59      void AddTailRcvPkt(void *pV)    { if (m_pSCC) { if (m_pSCC->pRcvPktList)
          m_pSCC->pRcvPktList->AddTail(pV);   }   }
60      int GetRcvListCount(void)       { if (m_pSCC) { if (m_pSCC->pRcvPktList)
          return m_pSCC->pRcvPktList->GetCount(); } return 0; }
61      void UnLockRcvPktList(void)     { if (m_pSCC) { if (m_pSCC->pCSRcvPkt)
          LeaveCriticalSection(m_pSCC->pCSRcvPkt);    }   }
62
63      void LockSendPktList(void)      { if (m_pSCC) { if (m_pSCC->pCSSendPkt)
          EnterCriticalSection(m_pSCC->pCSSendPkt);}  }
64      void AddTailSendPkt(void *pV)   { if (m_pSCC) { if (m_pSCC->pSendPktList)
          m_pSCC->pSendPktList->AddTail(pV);} }
65      void UnLockSendPktList(void)    { if (m_pSCC) { if (m_pSCC->pCSSendPkt)
          LeaveCriticalSection(m_pSCC->pCSSendPkt);}  }
66
67      //void KillMyThread(void)           { m_pSCC->bExitThread = 1;
            }
68
69      UINT GetPacketsSent(void)       { return (m_pSCC ?  m_pSCC->uPacketsSent :
          NULL ); }
70      void SetClientIp4(CString s)    { m_sClientIp4 = s;             }
71      CString GetClientIp4(void)      { return m_sClientIp4;          }
72
73      //void * GetWholePacket(int nPacketSize, int *pReceived);
74
75      BYTE GetConnectionStatus(void)      { return (m_pSCC ?  m_pSCC->bConnected :
          eInstrumentNotPresent); }
76      void SetConnectionStatus(BYTE s)    { if     (m_pSCC)   m_pSCC->bConnected =
          s;                          }
77
78      void SetClientPortIndex( int indx ) { m_nClientIndex = indx; }
79      // call these get/set function from ServerSocketOwnerThread ExitInstance to
          update
80      // the values
81      ST_SERVERS_CLIENT_CONNECTION * GetpSCC( void );
82      void SetpSCC( ST_SERVERS_CLIENT_CONNECTION* p ) { m_pSCM->m_pstSCM-
          >pClientConnection[m_nClientIndex] =  p; }
83      void NullpSCC(void)                             {m_pSCM->m_pstSCM-
          >pClientConnection[m_nClientIndex] =  0;}
84
85
86      // variables
87      CServerConnectionManagement *m_pSCM;        // ptr to the controlling class
```

```
88      ST_SERVERS_CLIENT_CONNECTION *m_pSCC;       // ptr to my connection info/
          statistics/objects
89      ST_SERVER_CONNECTION_MANAGEMENT *m_pstSCM;  // pointer to my global structure
           instance
90      int m_nMyServer;                            // which server are we connected
          stSCM[MAX_SERVERS]
91      int m_nClientIndex;                         // which instance of
          pClientConnection[MAX_CLIENTS_PER_SERVER];
92      CWinThread * m_pThread;                     // ptr to thread which created
          the socket
93      int m_nOwningThreadType;                    // 0=Listener, 1=ServerConnection
           thread owns this socket class
94      int m_nOwningThreadId;              // debugging
95      int m_nAsyncSocketCnt;              // debugging
96
97      CString m_sClientIp4;                       // IP4 address... 192.168.123.123
           etc of server connected socket
98
99      CCmdFifo *m_pFifo;      // In ClientConnectionManagement get PAG commands.
          Here gets instrument data
100                             // created in CServerSocket::CServerSocket
                        (CServerConnectionManagement *pSCM)
101                             // deleted in CServerSocket::~CServerSocket()
102     CString szName;
103     CHwTimer *m_pElapseTimer;   // created in CServerSocket::CServerSocket()
104                                 // deleted in CServerSocket::~CServerSocket()
105     int m_nElapseTime;
106     int m_nOnAcceptClientIndex;     // cheating to let OnAccept pass info to
          OnClose
107     int m_nSeqCntDbg[1024];
108     int m_nSeqIndx;
109     USHORT m_nLastSeqCnt;
110
111     // debugging
112     GenericPacketHeader m_HeaderDbg[8];
113     int m_dbg_cnt;      // counter to select pHeaderDbg variable
114     int m_nListCount;   // how deep is the linked list?
115     int m_nListCountChanged;
116     };
117
118 #endif
119
120
```