```
//---------------------------------------------------------------------------

#ifndef UnitDMClientsH
#define UnitDMClientsH
//---------------------------------------------------------------------------
#include <System.Classes.hpp>
#include <System.Win.ScktComp.hpp>
#include <IniFiles.hpp>
//---------------------------------------------------------------------------
#include <atomic>
#include <vector>
//---------------------------------------------------------------------------
#define __DYNAMIC__
#define __LAMBDA_THREAD__
#include "UnitThreadWorking.h"
//---------------------------------------------------------------------------
//  ПРОСТРАНСТВО НА ИМЕНАТА НА TSP (Time Server Protocol)
namespace tsp
{
    //  TSP КЛИЕНТ
    class Client
    {
    public:
        enum STATE {Closed = 0, Transient, Open} State; // състояние
        TClientSocket* csOut;           // изходен канал на клиента

        int intLocalPort;
        std::pair<TTime, int> pairREQ;
        std::pair<TTime, int> pairCNC;
        std::pair<TTime, int> pairDSC;
        std::pair<TTime, int> pairRPL;
        std::pair<TTime, int> pairERR;
        unsigned long rpl;

        __fastcall Client(void);
        void __fastcall Init(void);
    };

    //  КОНФИГУРАЦИОННИ ПАРАМЕТРИ НА ПАКЕТА TSP КЛИЕНТИ
    class Parameters
    {
    public:
        // Секция Form
        int intLeft;                    // координата на левия край главната форма
        int intTop;                     // координата на горния край на главната форма
        // Секция Log
        bool boolLogAuto;               // автоматична визуализация на протокола
        bool boolLogVisible;            // визуализация при отваряне на приложението
        int intLogHeight;               // височина на полето с протокола
        // Секция TimeServer
        String strIPAddress;            // IP адрес на TP сървъра
        // Секция Package
        int intVolume;                  // обем на пакета с множествени заявки
        bool boolConsecutive;           // последователен режим на подаване на заявките
        int intDelayToFree;             // период на обработка на отговора в клиента
        // Секция Report
        String strReportFileNamePrefix;
        String strReportFileVersion;
        const String strReportFileExt = ".csv";
        // Секция Debug
        bool boolL1;                    // тестване Ниво 1
        bool boolL2;                    // тестване Ниво 2
        bool boolL3;                    // тестване Ниво 3

        Parameters& operator=(Parameters& src)
        {
            intLeft         = src.intLeft;
            intTop          = src.intTop;

            boolLogAuto     = src.boolLogAuto;
            boolLogVisible  = src.boolLogVisible;
            intLogHeight    = src.intLogHeight;

            strIPAddress    = src.strIPAddress;
```

```
            intVolume       = src.intVolume;
            boolConsecutive = src.boolConsecutive;
            intDelayToFree  = src.intDelayToFree;

            strReportFileNamePrefix     = src.strReportFileNamePrefix;
            strReportFileVersion        = src.strReportFileVersion;

            boolL1          = src.boolL1;
            boolL2          = src.boolL2;
            boolL3          = src.boolL3;

            return *this;
        }
    };

    //  ПАКЕТ TSP КЛИЕНТИ
    class ClientsPack
    {
    public:
        std::vector<Client*> vClients;      // клиенти в пакета
        std::atomic<int> intPending;        // общ брой необслужени клиенти
        std::atomic<int> intCNC;            // клиенти на връзка
        std::atomic<int> intERR;            // клиенти с отхвърлена заявка
        std::atomic<bool> boolSyncing;      // обобщено състояние на пакета клиенти

        TClientSocket* csOutBase;
        TDateTime dtTsyn;                   // Tsyn
        TDateTime dtT0;                     // T0
        TDateTime dtTend;                   // Tend

        // Конфигурационни параметри
        tsp::Parameters Parameters;

        void __fastcall Init(TClientSocket* csOut, tsp::Parameters& p);
        String __fastcall ToMilliSeconds(TTime t, bool boolFormat = false);
        void __fastcall Save(void);
    };


    //  БУФЕРЕН LOG В ПАМЕТТА
    class Log
    {
    private:
        std::vector<String> Lines;

    public:
        __fastcall Log(void) {}
        void __fastcall Clear(void)     { Lines.clear(); }
        void __fastcall Add(String str, bool boolStopper);
        void __fastcall Add(TCustomWinSocket* sock, String str, bool boolStopper);
        void __fastcall Show(TMemo* dst, bool boolStopper);
    };
}
//---------------------------------------------------------------------------
class TdmClients : public TDataModule
{
__published:    // IDE-managed Components
    TClientSocket *csOut;
    void __fastcall csOutConnect(TObject *Sender, TCustomWinSocket *Socket);
    void __fastcall csOutDisconnect(TObject *Sender, TCustomWinSocket *Socket);
    void __fastcall csOutError(TObject *Sender, TCustomWinSocket *Socket, TErrorEvent ErrorEvent,
          int &ErrorCode);
    void __fastcall csOutRead(TObject *Sender, TCustomWinSocket *Socket);

private:    // User declarations
    // Променливи на състоянието  ////////////////////////////////////////
    int intSession;                 // номер на работна сесия

    tsp::ClientsPack clientsPackStat;   // статичен пакет TS клиенти
    tsp::Log logStat;                   // статичен системен протокол в паметта

    tsp::ClientsPack* clientsPack;      // динамичен пакет TS клиенти
    tsp::Log* log;                  // динамичен системен протокол в паметта
```

```
    // Конфигурационен файл ////////////////////////////////////////////////
    TIniFile* iniFile;
    String strIniFileName;

    // Конфигурационни параметри
    tsp::Parameters Parameters;

    void __fastcall __read_ini_file(tsp::Parameters& dst);
    void __fastcall __write_ini_file(void);
    int __fastcall GetChannelId(TObject *Sender);

public:     // User declarations
    __fastcall TdmClients(TComponent* Owner);
    __fastcall ~TdmClients(void);

    bool __fastcall IsSyncing(void)          { return clientsPack->boolSyncing; }
    int __fastcall GetPending(void)          { return clientsPack->intPending; }
    TDateTime& __fastcall GetTSyn(void)      { return clientsPack->dtTsyn; }


    void __fastcall SendBatchOfReq(void);    // формиране пакет от заявки
    void __fastcall Save(void)               { clientsPack->Save(); }

    const tsp::Parameters& GetParameters(void)    { return Parameters; }

    friend void DoWork(Item c, int id, int delay, TObject* o);
};
//---------------------------------------------------------------------------
extern PACKAGE TdmClients *dmClients;
//---------------------------------------------------------------------------
#endif
```