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

#include <Vcl.Dialogs.hpp>
#include <System.DateUtils.hpp>
#include <System.SysUtils.hpp>
#include <iomanip>
#include <fstream>
#pragma hdrstop

#include "UnitDMClients.h"
#include "UnitFormMain.h"
#include "UnitUtils.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma classgroup "Vcl.Controls.TControl"
#pragma resource "*.dfm"
TdmClients *dmClients;
//---------------------------------------------------------------------------
#include <mutex>
std::mutex mutex;
//---------------------------------------------------------------------------
__fastcall tsp::Client::Client(void)
{
    Init();
}
//---------------------------------------------------------------------------
void __fastcall tsp::Client::Init(void)
{
    intLocalPort = -1;

    pairREQ.first = 0; pairREQ.second = 0;
    pairCNC.first = 0; pairCNC.second = 0;
    pairDSC.first = 0; pairDSC.second = 0;
    pairRPL.first = 0; pairRPL.second = 0;
    pairERR.first = 0; pairERR.second = 0;
    rpl = 0;
}
//---------------------------------------------------------------------------
void __fastcall tsp::ClientsPack::Init(TClientSocket* csOut, tsp::Parameters& p)
{
    csOutBase = csOut;

    vClients.clear();

    Parameters = p;

    for(int i = 0; i < Parameters.intVolume; i++)
    {
        tsp::Client* tspClient = new tsp::Client;

        tspClient->csOut = new TClientSocket(csOut->Owner);
        tspClient->csOut->Port = 37;
        tspClient->csOut->Address = Parameters.strIPAddress;

        tspClient->csOut->OnConnect = csOut->OnConnect;
        tspClient->csOut->OnDisconnect = csOut->OnDisconnect;
        tspClient->csOut->OnError = csOut->OnError;
        tspClient->csOut->OnRead = csOut->OnRead;

        tspClient->State = tsp::Client::Closed;

        vClients.push_back(tspClient);
    }

    intPending  = 0;
    intCNC      = 0;
    intERR      = 0;
    boolSyncing = false;

    dtTsyn = 0;
    dtT0   = 0;
    dtTend = 0;
}
//---------------------------------------------------------------------------
```

```
String __fastcall tsp::ClientsPack::ToMilliSeconds(TTime t, bool boolFormat)
{
    String strResult;

    if(!boolFormat)
    {   // hh:mm:ss.zzz
        strResult = FormatDateTime("hh:mm:ss.zzz", t);
    }
    else
    {
        // ms
        if(!IsTimeZero(t))
        {
            strResult = MilliSecondsBetween(t, vClients[0]->pairREQ.first);
        }
    }

    return strResult;
}
//---------------------------------------------------------------------------
void __fastcall tsp::ClientsPack::Save(void)
{
    String strFileName = Parameters.strReportFileNamePrefix + "." +
                         Parameters.strReportFileVersion + "." +
                         FormatDateTime("yyyymmdd.hhmmss", dtTsyn.DateTimeString()) +
                         Parameters.strReportFileExt;

    // Resource Acquisition Is Initialization or RAII,
    // is a C++ programming technique[1][2] which binds
    // the life cycle of a resource that must be acquired before use
    // (allocated heap memory, thread of execution, open socket, open file,
    // locked mutex, disk space, database connection—anything
    // that exists in limited supply) to the lifetime of an object.
    std::wofstream file(strFileName.c_str());    // RAII

    if(file)
    {
        String strMode;
        if(Parameters.boolConsecutive)
        {
            strMode = "Consecutive";
        }
        else
        {
            strMode = "Simultaneous";
        }

        file << "ТЕСТОВЕ ПОД ТОВАР НА TIME PROTOCOL (RFC-868) СЪРВЪР" << std::endl << std::endl;

        file << "Източник на товар:;;;;" << formMain->Caption << std::endl << std::endl;

        file << "Дата:;;;;" << FormatDateTime("yyyy.mm.dd", dtTsyn) << std::endl;
        file << "IP адрес сървър:;;;;" << Parameters.strIPAddress << std::endl;
        file << "IP адрес клиент:;;;;" << csOutBase->Socket->LocalAddress << std::endl;
        file << "Обем на пакета:;;;;" << Parameters.intVolume << std::endl;
        file << "Режим на формиране на заявките:;;;;" << strMode << std::endl;
        file << "Задръжка до разпадане:;;;;" << Parameters.intDelayToFree << std::endl << std::endl

        file << "Tsyn:;;;;" << FormatDateTime("hh:mm:ss.zzz", dtTsyn) << std::endl;
        file << "T0:;;;;" << FormatDateTime("hh:mm:ss.zzz", dtT0) << std::endl;
        file << "Tend:;;;;" << FormatDateTime("hh:mm:ss.zzz", dtTend) << std::endl;
        file << "T0 - Tsyn:;;;;" << FormatDateTime("s,zzz", dtT0 - dtTsyn) << std::endl;
        file << "Tend - T0:;;;;" << FormatDateTime("s,zzz", dtTend - dtT0) << std::endl;
        file << "Tend - Tsyn:;;;;" << FormatDateTime("s,zzz", dtTend - dtTsyn) << std::endl << std:

        double t = MilliSecondsBetween(dtTend, dtTsyn)/1000.;
        double p = (Parameters.intVolume - intERR)/t;
        FormatSettings.DecimalSeparator = ',';
        String strP = FloatToStrF(p, ffFixed, 2, 2);
        file << "P:;;;;" << strP << std::endl;
        //file << "P:;;;;" << std::setprecision(2) << (Parameters.intVolume - intERR)/t << std::end
        file << "ERR:;;;;" << intERR << std::endl << std::endl;

        file << ";REQ;;CNC;;RPL;;DSC;;ERR;;;" << std::endl;
```

```
        file << "Client;T REQ;N cnc;T CNC;N cnc;T RPL;N cnc;T DSC;N cnc;T ERR;N cnc;Reply;Port" << 

        for(int i = 0; i < vClients.size(); i++)
        {
            file << i << ";"
                 << ToMilliSeconds(vClients[i]->pairREQ.first, true) << ";" << vClients[i]->pairREQ
                 << ToMilliSeconds(vClients[i]->pairCNC.first, true) << ";" << vClients[i]->pairCNC
                 << ToMilliSeconds(vClients[i]->pairRPL.first, true) << ";" << vClients[i]->pairRPL
                 << ToMilliSeconds(vClients[i]->pairDSC.first, true) << ";" << vClients[i]->pairDSC
                 << ToMilliSeconds(vClients[i]->pairERR.first, true) << ";" << vClients[i]->pairERR
                 << std::hex << std::uppercase << vClients[i]->rpl << ";"
                 << std::dec << vClients[i]->intLocalPort
                 << std::endl;
        }

        MessageDlg(strFileName + " saved", mtInformation, TMsgDlgButtons() << mbOK, 0);
    }
    else
    {
        throw Exception("Cannot open " + strFileName);
    }
}
//---------------------------------------------------------------------------
void __fastcall tsp::Log::Add(String str, bool boolStopper)
{
    if(!boolStopper)
    {
        String ws = FormatDateTime("hh:mm:ss.zzz",Time()) + " " + str;

        // формиране на критична секция до изхода на функцията
        // секцията сериализира достъпа на нишките до GUI
        std::lock_guard<std::mutex> guard(mutex);   // RAII
        //
        Lines.push_back(ws);
    }
}
//---------------------------------------------------------------------------
void __fastcall tsp::Log::Add(TCustomWinSocket* sock, String str, bool boolStopper)
{
    if(!boolStopper)
    {
        String ws = FormatDateTime("hh:mm:ss.zzz",Time()) +
                    " [" + sock->RemoteHost + "::" + sock->RemoteAddress + "] " +
                    str;

        // формиране на критична секция до изхода на функцията
        // секцията сериализира достъпа на нишките до GUI
        std::lock_guard<std::mutex> guard(mutex);   // RAII
        //
        Lines.push_back(ws);
    }
}
//---------------------------------------------------------------------------
void __fastcall tsp::Log::Show(TMemo* dst, bool boolStopper)
{
    if(!boolStopper)
    {
        for(int i = 0; i < Lines.size(); i++)
        {
            dst->Lines->Add(Lines[i]);
        }
    }
    else
    {
        dst->Lines->Add("Debug.L1");
    }
}
//---------------------------------------------------------------------------
__fastcall TdmClients::TdmClients(TComponent* Owner)
    : TDataModule(Owner)
{
    strIniFileName = Application->ExeName;
    strIniFileName = strIniFileName.SubString(1, strIniFileName.Length() - 3) + "ini";
```

```
    iniFile = new TIniFile(strIniFileName);

    intSession = 0;

    __read_ini_file(Parameters);    // първоначално четене на конфигурационните параметри

//  СЪЗДАВАНЕТО НА КЛИЕНТИТЕ В ПАКЕТА
//  извършва се също в началото на всяка синхронизация

#if defined(__DYNAMIC__)
    log = new tsp::Log;
    clientsPack = new tsp::ClientsPack;
#else
    log = &logStat;
    clientsPack = &clientsPackStat;
#endif
    clientsPack->Init(csOut, Parameters);
}
//---------------------------------------------------------------------------
__fastcall TdmClients::~TdmClients(void)
{
    __write_ini_file();
}
//---------------------------------------------------------------------------
void __fastcall TdmClients::__read_ini_file(tsp::Parameters& dst)
{
    if(!FileExists(strIniFileName))
    {
        MessageDlg("Missing configuration file!", mtError, TMsgDlgButtons() << mbOK, 0);
    }

    // Секция Form
    dst.intTop    = iniFile->ReadInteger(L"Form", L"Top", 0);
    dst.intLeft   = iniFile->ReadInteger(L"Form", L"Left", 0);

    // Секция Log
    dst.boolLogAuto     = iniFile->ReadInteger(L"Log", L"Auto", 0);
    dst.boolLogVisible  = iniFile->ReadInteger(L"Log", L"Visible", 0);
    dst.intLogHeight    = iniFile->ReadInteger(L"Log", L"Height", 300);

    // Секция TimeServer
    dst.strIPAddress    = iniFile->ReadString(L"TimeServer", L"IPAddress", L"127.0.0.1");

    // Секция Package
    dst.intVolume       = iniFile->ReadInteger(L"Package", L"Volume", 1);
    dst.intDelayToFree  = iniFile->ReadInteger(L"Package", L"DelayToFree", 0);
    dst.boolConsecutive = iniFile->ReadInteger(L"Package", L"Consecutive", 1);

    // Секция Report
    dst.strReportFileNamePrefix     = iniFile->ReadString(L"Report", L"Prefix", "TBL");
    dst.strReportFileVersion        = iniFile->ReadString(L"Report", L"Version", "0.0.000");

    // Секция Debug
    dst.boolL1  = iniFile->ReadInteger(L"Debug", L"L1", 0);
    dst.boolL2  = iniFile->ReadInteger(L"Debug", L"L2", 0);
    dst.boolL3  = iniFile->ReadInteger(L"Debug", L"L3", 0);
}
//---------------------------------------------------------------------------
void __fastcall TdmClients::__write_ini_file(void)
{
    try
    {
        iniFile->WriteInteger("Form", "Top", formMain->Top);
        iniFile->WriteInteger("Form", "Left", formMain->Left);

        iniFile->WriteInteger(L"Log", L"Visible", formMain->memoLog->Visible);
    }
    catch(Exception& e)
    {
        MessageDlg(e.Message, mtError, TMsgDlgButtons() << mbOK, 0);
    }

    delete iniFile;
}
```

```
//---------------------------------------------------------------------------
int __fastcall TdmClients::GetChannelId(TObject *Sender)
{
    int intId = -1;

    for(int i = 0; i < clientsPack->vClients.size(); i++)
    {
        if(clientsPack->vClients[i]->csOut == Sender)
        {
            intId = i;
            break;
        }
    }

    return intId;
}
//---------------------------------------------------------------------------
void __fastcall TdmClients::SendBatchOfReq(void)
{
    if(clientsPack->boolSyncing)
    {
        String str = "Cannot start nested sync [Pending clients #" + IntToStr(GetPending()) + "]";
        MessageDlg(str, mtError, TMsgDlgButtons() << mbOK, 0);
        return;
    }

    if(formMain->memoLog->Visible)
    {
        formMain->labelShowHideLogClick(formMain);
    }
    formMain->memoLog->Clear();

    formMain->buttonSync->Enabled = false;
    Screen->Cursor = crHourGlass;

    intSession++;

#if defined(__DYNAMIC__)
    delete log;
    delete clientsPack;

    log = new tsp::Log;
    clientsPack = new tsp::ClientsPack;
#endif

    log->Clear();
    formMain->stBarReport->Panels->Items[0]->Text = "#" + IntToStr(intSession);
    formMain->stBarReport->Panels->Items[1]->Text = "";
    formMain->stBarReport->Panels->Items[2]->Text = "";
    formMain->stBarReport->Panels->Items[3]->Text = "In progress...";

    __read_ini_file(Parameters);    // четене на параметрите от конфигурационния файл

    clientsPack->Init(csOut, Parameters);

    clientsPack->intPending = clientsPack->Parameters.intVolume;
    clientsPack->boolSyncing = true;

    if(clientsPack->Parameters.boolConsecutive)
    {
        if(clientsPack->vClients.size() > 0)
        {
            clientsPack->vClients[0]->csOut->Open();
            clientsPack->vClients[0]->State = tsp::Client::Transient;
            clientsPack->vClients[0]->intLocalPort = clientsPack->vClients[0]->csOut->Socket->Local
            clientsPack->vClients[0]->pairREQ.first = Time();
            clientsPack->vClients[0]->pairREQ.second = clientsPack->intCNC;

            clientsPack->dtTsyn = Date() + clientsPack->vClients[0]->pairREQ.first;
            log->Add("SYN", Parameters.boolL1);
        }
    }
    else
    {
```

```
        for(int i = 0; i < clientsPack->vClients.size(); i++)
        {
            clientsPack->vClients[i]->csOut->Open();
            clientsPack->vClients[i]->State = tsp::Client::Transient;
            clientsPack->vClients[i]->intLocalPort = clientsPack->vClients[i]->csOut->Socket->Local
            clientsPack->vClients[i]->pairREQ.first = Time();
            clientsPack->vClients[i]->pairREQ.second = clientsPack->intCNC;

            if(i == 0)
            {
                clientsPack->dtTsyn = Date() + clientsPack->vClients[0]->pairREQ.first;
                log->Add("SYN", Parameters.boolL1);
            }
        }
    }
}
//---------------------------------------------------------------------------
void __fastcall TdmClients::csOutConnect(TObject *Sender, TCustomWinSocket *Socket)
{
    int intReqId = GetChannelId(Sender);
    log->Add(Socket, "CNC[" + IntToStr(intReqId) + "]", Parameters.boolL1);

    if(intReqId >= 0)
    {
        clientsPack->intCNC++;
        clientsPack->vClients[intReqId]->State = tsp::Client::Open;
        clientsPack->vClients[intReqId]->pairCNC.first = Time();
        clientsPack->vClients[intReqId]->pairCNC.second = clientsPack->intCNC;

        if(IsTimeZero(clientsPack->dtT0))
        {
            clientsPack->dtT0 = Date() + clientsPack->vClients[intReqId]->pairCNC.first;
        }

        if(clientsPack->Parameters.boolConsecutive)
        {
            while(++intReqId < clientsPack->vClients.size())
            {
                if(clientsPack->vClients[intReqId]->State != tsp::Client::Closed)
                {
                    continue;
                }

                clientsPack->vClients[intReqId]->csOut->Open();
                clientsPack->vClients[intReqId]->State = tsp::Client::Transient;
                clientsPack->vClients[intReqId]->intLocalPort = clientsPack->vClients[intReqId]->cs
                clientsPack->vClients[intReqId]->pairREQ.first = Time();
                clientsPack->vClients[intReqId]->pairREQ.second = clientsPack->intCNC;

                break;
            }
        }
    }
}
//---------------------------------------------------------------------------
void __fastcall TdmClients::csOutDisconnect(TObject *Sender, TCustomWinSocket *Socket)
{
    int intReqId = GetChannelId(Sender);
    log->Add(Socket, "DSC[" + IntToStr(intReqId) + "]", Parameters.boolL1);

    if(intReqId >= 0)
    {
        clientsPack->intPending--;
        clientsPack->intCNC--;
        clientsPack->vClients[intReqId]->State = tsp::Client::Closed;
        clientsPack->vClients[intReqId]->pairDSC.first = Time();
        clientsPack->vClients[intReqId]->pairDSC.second = clientsPack->intCNC;
    }

    if(clientsPack->intPending == 0)
    {
        clientsPack->boolSyncing = false;
        clientsPack->dtTend = Date() + clientsPack->vClients[intReqId]->pairDSC.first;
```

```
        String strDuration = FormatDateTime("s.zzz", clientsPack->dtT0 - clientsPack->dtTsyn) + " |
                             FormatDateTime("s.zzz", clientsPack->dtTend - clientsPack->dtT0) + " |
                             FormatDateTime("s.zzz", clientsPack->dtTend - clientsPack->dtTsyn) + "
        formMain->stBarReport->Panels->Items[1]->Text = strDuration;
        formMain->stBarReport->Panels->Items[2]->Text = IntToStr(clientsPack->intERR);
        formMain->stBarReport->Panels->Items[3]->Text = "READY";

        log->Show(formMain->memoLog, Parameters.boolL1);
        if(!formMain->memoLog->Visible && Parameters.boolLogAuto)
        {
            formMain->labelShowHideLogClick(formMain);
        }

        Screen->Cursor = crDefault;
        formMain->buttonSync->Enabled = true;
    }
}
//---------------------------------------------------------------------------
void __fastcall TdmClients::csOutError(TObject *Sender, TCustomWinSocket *Socket,
          TErrorEvent ErrorEvent, int &ErrorCode)
{
    int intReqId = GetChannelId(Sender);
    log->Add(Socket, "ERR[" + IntToStr(intReqId) + "][" + IntToStr(ErrorCode) + "]", Parameters.bool

    if(intReqId >= 0)
    {
        clientsPack->intPending--;
        clientsPack->intERR++;
        if(!IsTimeZero(clientsPack->vClients[intReqId]->pairCNC.first))
        {
            clientsPack->intCNC--;
        }
        clientsPack->vClients[intReqId]->State = tsp::Client::Closed;
        clientsPack->vClients[intReqId]->pairERR.first = Time();
        clientsPack->vClients[intReqId]->pairERR.second = clientsPack->intCNC;
    }

    if(clientsPack->intPending == 0)
    {
        clientsPack->boolSyncing = false;
        clientsPack->dtTend = Date() + clientsPack->vClients[intReqId]->pairERR.first;

        String strDuration = FormatDateTime("s.zzz", clientsPack->dtT0 - clientsPack->dtTsyn) + " |
                             FormatDateTime("s.zzz", clientsPack->dtTend - clientsPack->dtT0) + " |
                             FormatDateTime("s.zzz", clientsPack->dtTend - clientsPack->dtTsyn) + "
        formMain->stBarReport->Panels->Items[1]->Text = strDuration;
        formMain->stBarReport->Panels->Items[2]->Text = IntToStr(clientsPack->intERR);
        formMain->stBarReport->Panels->Items[3]->Text = "READY";

        log->Show(formMain->memoLog, Parameters.boolL1);
        if(!formMain->memoLog->Visible && Parameters.boolLogAuto)
        {
            formMain->labelShowHideLogClick(formMain);
        }

        Screen->Cursor = crDefault;
        formMain->buttonSync->Enabled = true;
    }
    else if(clientsPack->Parameters.boolConsecutive)
    {
        while(++intReqId < clientsPack->vClients.size())
        {
            if(clientsPack->vClients[intReqId]->State != tsp::Client::Closed)
            {
                continue;
            }

            clientsPack->vClients[intReqId]->csOut->Open();
            clientsPack->vClients[intReqId]->State = tsp::Client::Transient;
            clientsPack->vClients[intReqId]->intLocalPort = clientsPack->vClients[intReqId]->csOut-
            clientsPack->vClients[intReqId]->pairREQ.first = Time();
            clientsPack->vClients[intReqId]->pairREQ.second = clientsPack->intCNC;

            break;
```

```
        }
    }

    ErrorCode = 0;
}
//---------------------------------------------------------------------------
void __fastcall TdmClients::csOutRead(TObject *Sender, TCustomWinSocket *Socket)
{
    int intReqId = GetChannelId(Sender);

////////////////////////////////////////////////////////////////////////////
//  УЧАСТЪКЪТ Е ПРЕМЕСТЕН В КОДА НА ФОНОВАТА НИШКА
//
//  unsigned long ulTime;
//
//  Socket->ReceiveBuf(&ulTime, 4);
//  ulTime = ntohl(ulTime);
//
//  if(intReqId >= 0)
//  {
//      clientsPack->vClients[intReqId]->pairRPL.first = Time();
//      clientsPack->vClients[intReqId]->pairRPL.second = clientsPack->intCNC;
//      clientsPack->vClients[intReqId]->rpl = ulTime;
//  }
//
//  log->Add(Socket, "RPL[" + IntToStr(intReqId) + "]::[" + IntToHex((int)ulTime, 8) + "]", Paramet
////////////////////////////////////////////////////////////////////////////

    // СИМУЛАЦИЯ НА ОБРАБОТКА
    // многозадачно обслужване за избягване на сериализацията
    // на паралелните клонове
    //
#if !defined(__LAMBDA_THREAD__)
    std::thread threadWorking(DoWork, Socket, intReqId, clientsPack->intDelayToFree);
#else
    int intDelay = clientsPack->Parameters.intDelayToFree;
    std::thread threadWorking = std::thread([Socket, intReqId, intDelay, this]()
        {
            unsigned long ulTime;

            Socket->ReceiveBuf(&ulTime, 4);
            ulTime = ntohl(ulTime);

            if(intReqId >= 0)
            {
                this->clientsPack->vClients[intReqId]->pairRPL.first = Time();
                this->clientsPack->vClients[intReqId]->pairRPL.second = this->clientsPack->intCNC;
                this->clientsPack->vClients[intReqId]->rpl = ulTime;
            }
            this->log->Add(Socket, "RPL[" + IntToStr(intReqId) + "]::[" + IntToHex((int)ulTime, 8)

            std::chrono::milliseconds sleep_dur(intDelay);
            std::this_thread::sleep_for(sleep_dur);

            Socket->Close();
        });
#endif
    threadWorking.detach();      // развързване на дъщерната нишка от основната
    ////////////////////////////////////////////////////////////////////////////
}
//---------------------------------------------------------------------------
```