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

#include <Vcl.Dialogs.hpp>
#pragma hdrstop

#include "UnitDMClients.h"
#include "UnitFormMain.h"
#include "UnitUtils.h"
#include "UnitThreadWorking.h"
//---------------------------------------------------------------------------
#include <mutex>
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma classgroup "Vcl.Controls.TControl"
#pragma resource "*.dfm"
TdmClients *dmClients;
//---------------------------------------------------------------------------
__fastcall TdmClients::TdmClients(TComponent* Owner)
    : TDataModule(Owner)
{
    __read_ini_file();  // четене на параметрите от конфигурационния файл

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

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

        clientsPack.vClients.push_back(tspClient);
    }
    clientsPack.intPending = 0;
    clientsPack.boolSyncing = false;
}
//---------------------------------------------------------------------------
__fastcall TdmClients::~TdmClients(void)
{
    __write_ini_file();
}
//---------------------------------------------------------------------------
void __fastcall TdmClients::__read_ini_file(void)
{
    strIniFileName = Application->ExeName;
    strIniFileName = strIniFileName.SubString(1, strIniFileName.Length() - 3) + "ini";

    iniFile = new TIniFile(strIniFileName);
    if(!FileExists(strIniFileName))
    {
        MessageDlg("Missing configuration file!", mtError, TMsgDlgButtons() << mbOK, 0);
    }

    formMain->Top    = iniFile->ReadInteger(L"FormPos", L"Top", 0);
    formMain->Left   = iniFile->ReadInteger(L"FormPos", L"Left", 0);

    formMain->memoLog->Visible = !iniFile->ReadInteger(L"Log", L"Visible", 0);

    strIPAddress     = iniFile->ReadString(L"TimeServer", L"IPAddress", L"127.0.0.1");

    intVolume        = iniFile->ReadInteger(L"Package", L"Volume", 1);
    intDelayToFree   = iniFile->ReadInteger(L"Package", L"DelayToFree", 0);
    boolConsecutive  = iniFile->ReadInteger(L"Package", L"Consecutive", 1);
}
//---------------------------------------------------------------------------
void __fastcall TdmClients::__write_ini_file(void)
{
    try
    {
        iniFile->WriteInteger("FormPos", "Top", formMain->Top);
        iniFile->WriteInteger("FormPos", "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)
    {
        MessageDlg("Cannot start nested sync", mtError, TMsgDlgButtons() << mbOK, 0);
        return;
    }

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

    clientsPack.intPending = 0;
    clientsPack.boolSyncing = true;

    if(boolConsecutive)
    {
        if(clientsPack.vClients.size() > 0)
        {
            clientsPack.vClients[0]->csOut->Open();
            clientsPack.vClients[0]->State = tsp::Client::Transient;
        }
    }
    else
    {
        for(int i = 0; i < clientsPack.vClients.size(); i++)
        {
            clientsPack.vClients[i]->csOut->Open();
            clientsPack.vClients[i]->State = tsp::Client::Transient;
        }
    }
}
//---------------------------------------------------------------------------
void __fastcall TdmClients::csOutConnect(TObject *Sender, TCustomWinSocket *Socket)
{
    int intReqId = GetChannelId(Sender);
    AddToLog(Socket, "CNC[" + IntToStr(intReqId) + "]");

    if(intReqId >= 0)
    {
        clientsPack.intPending++;
        clientsPack.vClients[intReqId]->State = tsp::Client::Open;

        if(boolConsecutive)
        {
            if(++intReqId < clientsPack.vClients.size())
            {
                clientsPack.vClients[intReqId]->csOut->Open();
                clientsPack.vClients[intReqId]->State = tsp::Client::Transient;
            }
        }
    }
}
//---------------------------------------------------------------------------
void __fastcall TdmClients::csOutDisconnect(TObject *Sender, TCustomWinSocket *Socket)
{
    int intReqId = GetChannelId(Sender);
    AddToLog(Socket, "DSC[" + IntToStr(intReqId) + "]");

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

    if(clientsPack.intPending == 0)
    {
        clientsPack.boolSyncing = false;

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

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

    if(clientsPack.intPending == 0)
    {
        clientsPack.boolSyncing = false;

        formMain->buttonSync->Enabled = true;
        Screen->Cursor = crDefault;
    }

    ErrorCode = 0;
}
//---------------------------------------------------------------------------
void __fastcall TdmClients::csOutRead(TObject *Sender, TCustomWinSocket *Socket)
{
    unsigned long ulTime;

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

    int intReqId = GetChannelId(Sender);
    AddToLog(Socket, "RPL[" + IntToStr(intReqId) + "]::[" + IntToHex((int)ulTime, 8) + "]");

    // СИМУЛАЦИЯ НА ОБРАБОТКА
    // многозадачно обслужване за избягване на сериализацията
    // на паралелните клонове
    //
    std::thread threadWorking(DoWork, Socket, intReqId, intDelayToFree);
    threadWorking.detach();      // развързване на дъщерната нишка от основната
    ////////////////////////////////////////////////////////////////////////////
}
//---------------------------------------------------------------------------
```