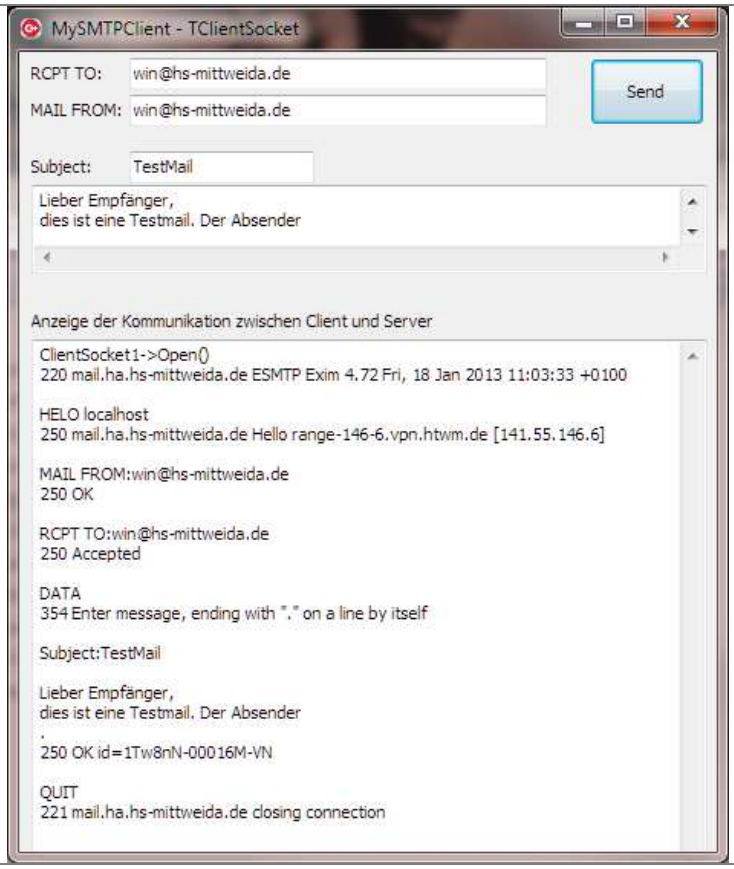
MySMTPClient		Prof. Dr.-Ing. habil. L. Winkler	HOCHSCHULE MITTWEIDA UNIVERSITY OF APPLIED SCIENCES
C++-Project mit TClientSocket	2013-12	Fakultät Informations- und Elektrotechnik	

1 Ziel des Projektes

Es soll ein SMTP-Client nach RFC 2821 *in einfachster Form* programmiert werden.

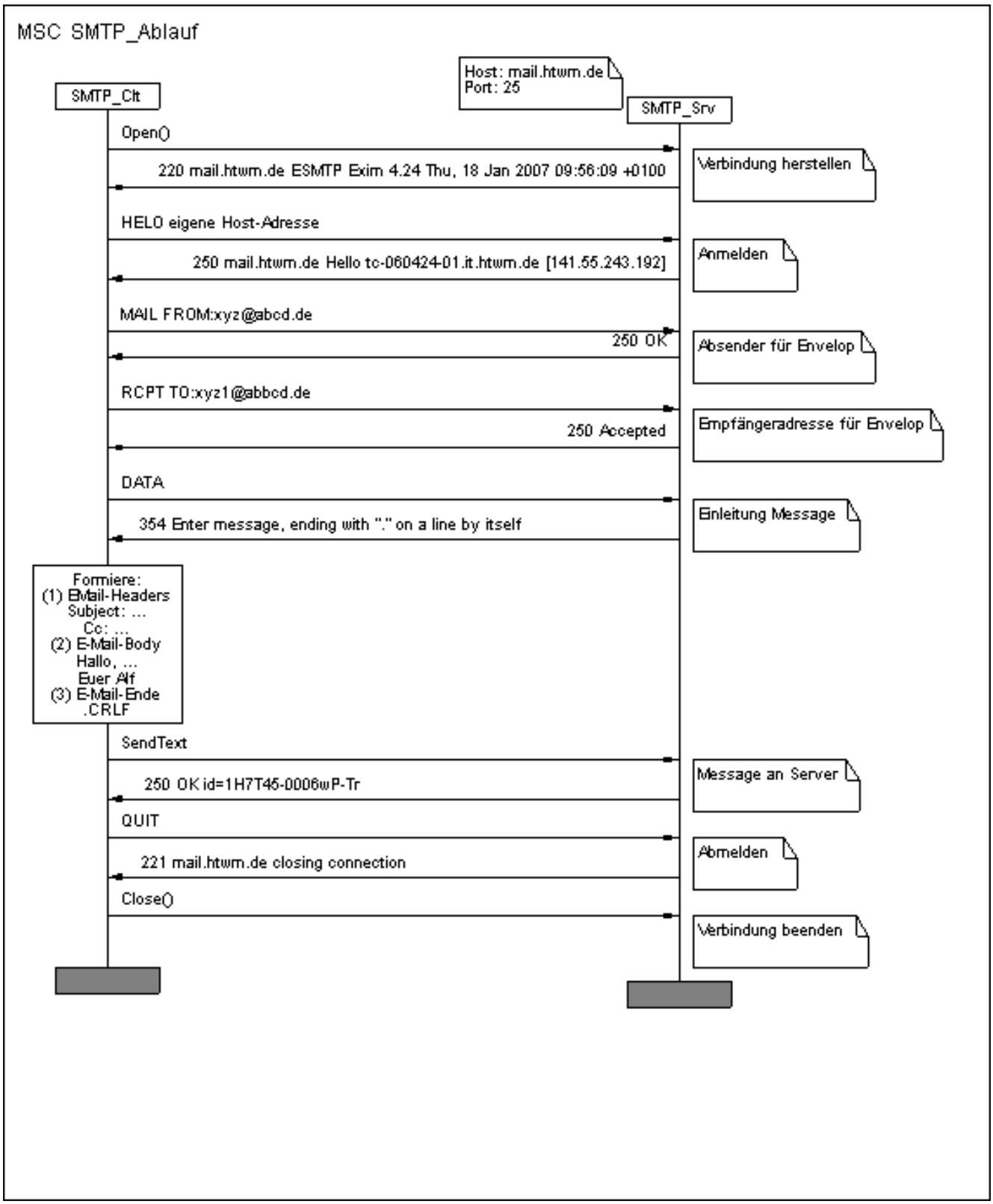
- Entwicklungsumgebung Embarcadero® RAD Studio XE2-C++.
- Bei der Realisierung der nebenstehenden Benutzeroberfläche nutzen Sie bitte folgende Klassen und Objektbezeichner:
  - TEdit: Edit1, Edit2, Edit3
  - TButton: Button1
  - TMemo: Memo1, Memo2
  - TLabel: Label1 .. Label4
  - TClientSocket: ClientSocket1



2 Command-Reply-Sequences

Connection-Functions SMTP-Commands	Funktion	Inter-mediate	Success	Error
CONNECTION ESTABLISHMENT	Socketverbindung zum SMTP-Server herstellen		220	554
HELO localhost*crlf*	Errichtung einer Session		250	504, 550
MAIL FROM:mail-addr*crlf*	Übergabe des Absenders für den Umschlag		250	552, 451, 452, 550, 553, 503
RCPT TO:mail-addr*crlf*	Übergabe des Empfängers für den Umschlag. Wiederholen, wenn Mail an mehrere gesendet werden soll!		250, 251	550, 551, 552, 553, 450, 451, 452, 503, 550
DATA*crlf*	Leitet Eingabe des E-Mail-Inhaltes ein	354	250	552, 554, 451, 452, 451, 554, 503
.crlf	Beendet die E-Mail-Inhaltseingabe		250	
QUIT*crlf*	Beenden der Session beim SMTP-Server.		221	
CONNECTION CLEARING	Socketverbindung zum SMTP-Server beenden.			
Weitere SMTP-Commands				
VRFY:mail-add*crlf*	Kontrolle, ob ein bestimmter Empfänger verfügbar ist.		250, 251, 252	550, 551, 553, 502, 504
HELP*crlf*	HELP leer Anzeige aller Commands HELP Cmd Hilfe zu Cmd		211, 214	502, 504
NOOP*crlf*	zur Aufrechterhaltung der Session		250	

3 **Der MSC** (message sequence chart)



4 Realisierung des Projektes

Legen Sie ein neues Projekt “MySMTPClt“ an, speichern Sie dies in einem gleichnamigen Ordner.

Erzeugen Sie die oben gezeigte grafische Oberfläche. Halten Sie sich bitte an die empfohlenen Objektnamen. Dies erleichtert eine eventuelle Fehlersuche.

Für den Socket verwenden Sie folgende Properties:

- **host : mail.hs-mittweida.de**
- **port : 25**
- **active : false**
- **ClientType: ctBlocking**

Erzeugen Sie für das Button1-Ereignis „OnClick“ eine Methode Button1Click. In diesem Methodenrumpf wird der wesentliche Programmiertext untergebracht.

Bei der Programmierung halten Sie sich an den MSC.
Zuerst wird eine Socket-Verbindung zum Server hergestellt. Falls die Antwort positiv war (220) folgt HELO, wenn Antwort 250, dann MAIL FROM, wenn Antwort 250 MAIL TO, wenn Antwort 250 DATA usw.
Gibt der Server keine Success-Message sondern eine Error-Message zurück, wird die Sitzung mit QUIT beendet und die Socket-Verbindung beendet.

_____
Grün: Header und Kommentare
Rot: Socket-Verbindung auf- und abbauen
Lila: Socketfunktionen nutzen
Blau: SMTP-Protokoll

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include <stdio.h>
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;
char recv_buf[128];
int len;
AnsiString txt, reply;

//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{

}
//---------------------------------------------------------------------------
void __fastcall TForm1::CltSockRead(TObject *Sender)
{
  len=ClientSocket1->Socket->ReceiveBuf(&recv_buf,128);  //socket lesen
  reply=((AnsiString)recv_buf).SubString(0,3);  //reply-nummer extrahieren
  Memo2->Lines->Add(((AnsiString)recv_buf).SubString(0,len)); //anzeigen
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
//--Memo2 löschen
  Memo2->Clear();
```

```
//--Verbindung zum Mail-Server herstellen
  Memo2->Lines->Add("ClientSocket1->Open()");
  ClientSocket1->Open();

  CltSockRead(Form1);
  if (reply!="220") goto ende;

//--am Server melden
  Memo2->Lines->Add("HELO localhost");
  ClientSocket1->Socket->SendText("HELO localhost\r\n");
  CltSockRead(Form1);
  if (reply!="250") goto ende;

//--Absenderadresse für Umschlag (envelop) übergeben
  Memo2->Lines->Add("MAIL FROM:"+Edit1->Text);
  ClientSocket1->Socket->SendText("MAIL FROM:"+Edit1->Text+"\r\n");
  CltSockRead(Form1);
  if (reply!="250") goto ende;

//--Empfängeradresse für Umschlag (envelop) übergeben
  Memo2->Lines->Add("RCPT TO:"+Edit2->Text);
  ClientSocket1->Socket->SendText("RCPT TO:"+Edit2->Text+"\r\n");
  CltSockRead(Form1);
  if (reply!="250") goto ende;

//--Übergabe der Message
  //-Einleitung
  Memo2->Lines->Add("DATA");
  ClientSocket1->Socket->SendText("DATA\r\n");
  CltSockRead(Form1);
  if (reply!="354") goto ende;

  //-Header-Infos
  Memo2->Lines->Add("Subject:"+Edit3->Text+"\r\n");
  ClientSocket1->Socket->SendText("Subject:"+Edit3->Text+"\r\n");

  //Der eigentliche Text
  Memo2->Text=Memo2->Text+Memo1->Text;
  ClientSocket1->Socket->SendText(Memo1->Text);

  //-Beendigung der Message
  Memo2->Lines->Add(".");
  ClientSocket1->Socket->SendText("\r\n.\r\n");
  CltSockRead(Form1);
  if (reply!="250") goto ende;

//--Beenden der Sitzung
ende:
  Memo2->Lines->Add("QUIT");
  ClientSocket1->Socket->SendText("QUIT\r\n");
  CltSockRead(Form1);
  ClientSocket1->Close();
}
//---------------------------------------------------------------------------
```