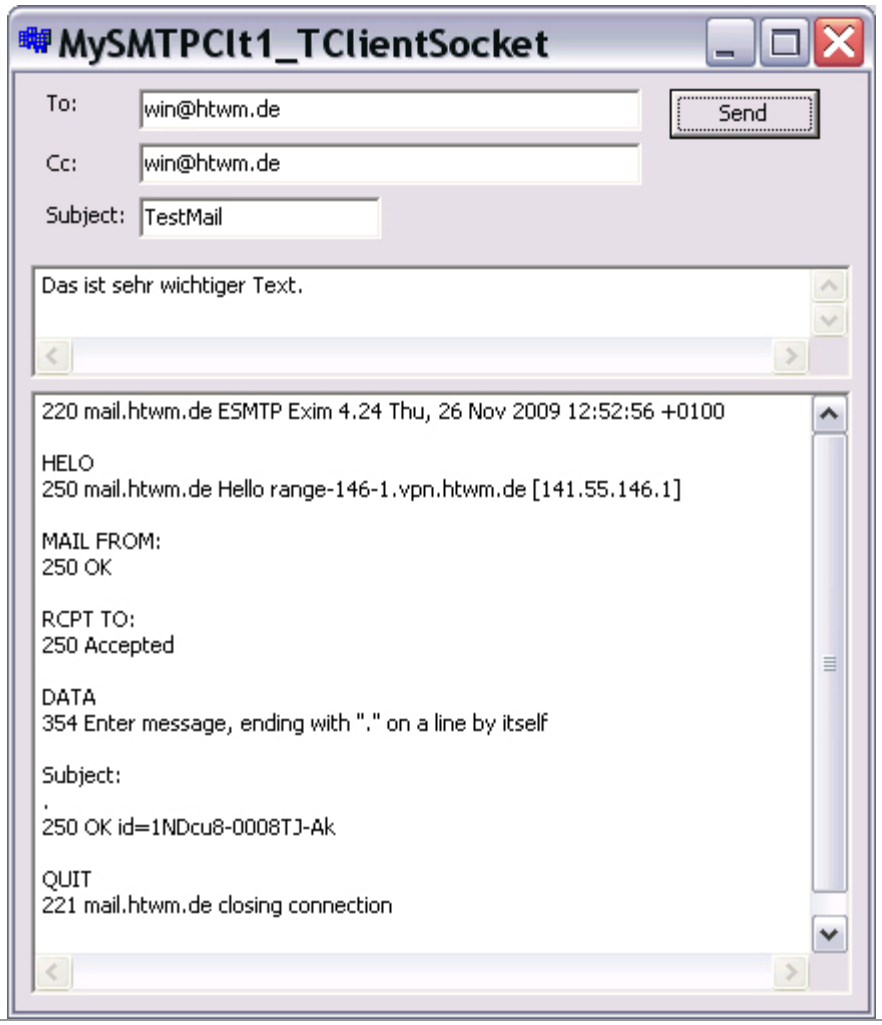
1.1 Ziel des Projektes

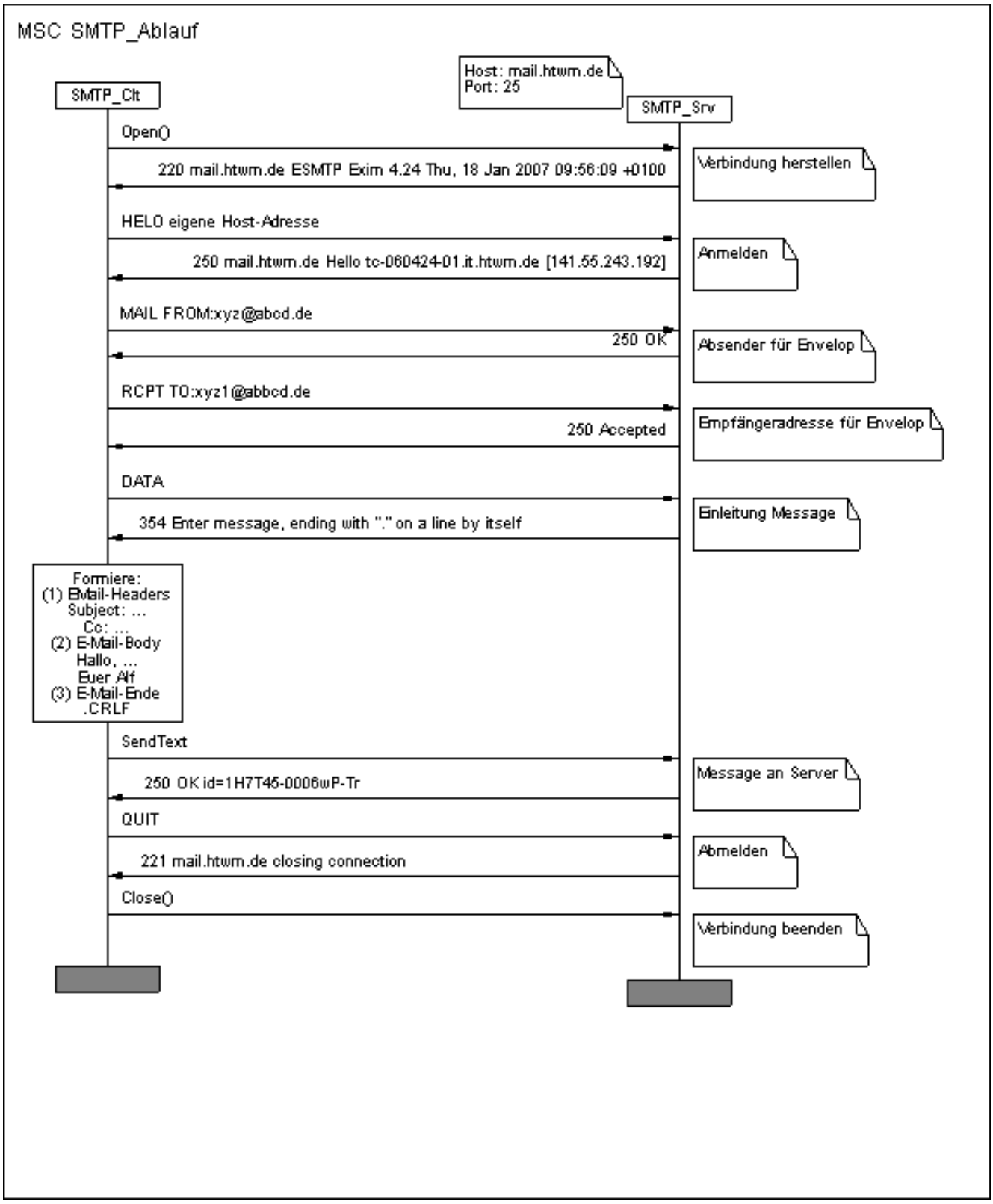
Es soll ein SMTP-Client nach RFC 2821 *in einfachster Form* programmiert werden.
Es wird die Borland-IDE-Komponente TClientSocket verwende.
Achten Sie darauf, dass der Socket_Eigenschaft ClientType = ctBlocking eingestellt ist.



2 Wichtige SMTP-Requests

HELO localhost*crlf*	Anmelden beim Server
MAIL FROM:mail-addr*crlf*	Umschlag: spezifiziert den Absender
RCPT TO: mail-addr*crlf*	Umschlag: spezifiziert Empfänger. Wiederholen, wenn an mehrere!
DATA*crlf*	Inhalt: leitet die Eingabe des E-Mail-Inhaltes ein
.*crlf*	Abbruch einer laufenden Übertragung und Reset der Verbindung.
RSET *crlf*	Abbruch einer laufenden Übertragung und Reset der Verbindung.
VRFY:mail-add*crlf*	Kontrolle, ob ein bestimmter Empfänger verfügbar ist.
HELP ? *crlf*	Fordert Hilfeinstruktionen an.
NOOP *crlf*	gibt einen positiven Wert zurück, falls der Server noch lebt
QUIT *crlf*	Abmelden beim SMTP-Server.

3 Der MSC (message sequence chart)



4 Realisierung des Projektes

Legen Sie ein neues Projekt “MySMTPClt1_TClientSocket“ an, speichern Sie dies in einem gleichnamigen Ordner.

Erzeugen Sie die gezeigte grafische Oberfläche wie folgt.

- für den E-Mail-Text ein TMemo (Standard)
- für die Ablaufanzeige ein TRichEdit (Win32)

- für den Socket die Komponente TClientSocket (Internet). Properties:
  - **host : mail.htwm.de**
  - **port : 25**
  - **active : false**
  - **ClientType: ctBlocking**

Bei SendClick auf Menü „Send“ verwenden Sie die Methode ClientSocket1->Open.
Es wird eine Verbindung zum Server hergestellt. Dieser schickt eine Antwort.
Dann werden die Request-PDUs zum Server gesendet und die Response-PDUs vom Server empfangen und ausgewertet.

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


TForm1 *Form1;
char recv_buf[128];
int len;

//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Send_Click(TObject *Sender)
{
//--Verbindung zum Mail-Server herstellen
  ClientSocket1->Host="mail.htwm.de";
  ClientSocket1->Port=25;
  ClientSocket1->Open();

  len=ClientSocket1->Socket->ReceiveBuf(&recv_buf,128);
  Memo2->Lines->Add(((AnsiString)recv_buf).SubString(0,len));
  if (((AnsiString)recv_buf).SubString(0,3)!="220") goto ende;

//--am Server melden
  Memo2->Lines->Add("HELO");
  ClientSocket1->Socket->SendText("HELO localhost\r\n");
  len=ClientSocket1->Socket->ReceiveBuf(&recv_buf,128);
  Memo2->Lines->Add(((AnsiString)recv_buf).SubString(0,len));
  if (((AnsiString)recv_buf).SubString(0,3)!="250") goto ende;

//--Umschlag schreiben (envelop)
  //Absender
  Memo2->Lines->Add("MAIL FROM:");
  ClientSocket1->Socket->SendText("MAIL FROM:win@htwm.de\r\n");
  len=ClientSocket1->Socket->ReceiveBuf(&recv_buf,128);
  Memo2->Lines->Add(((AnsiString)recv_buf).SubString(0,len));
  if (((AnsiString)recv_buf).SubString(0,3)!="250") goto ende;

  //Empfängeradresse
  Memo2->Lines->Add("RCPT TO:");
  ClientSocket1->Socket->SendText("RCPT TO:win@htwm.de\r\n");
  len=ClientSocket1->Socket->ReceiveBuf(&recv_buf,128);
  Memo2->Lines->Add(((AnsiString)recv_buf).SubString(0,len));
  if (((AnsiString)recv_buf).SubString(0,3)!="250") goto ende;
```

```
//--Übergabe der Message
  //Einleitung
  Memo2->Lines->Add("DATA");
  ClientSocket1->Socket->SendText("DATA\r\n");
  len=ClientSocket1->Socket->ReceiveBuf(&recv_buf,128);
  Memo2->Lines->Add(((AnsiString)recv_buf).SubString(0,len));
  if (((AnsiString)recv_buf).SubString(0,3)!="354") goto ende;

  //Header-Infos
  Memo2->Lines->Add("Subject:");
  ClientSocket1->Socket->SendText("Subject:Eine Testmail\r\n");
  //Der eigentliche Text
  //ClientSocket1->Socket->SendText("Hallo Leute, dies ist eine Testmail\r\n");
  ClientSocket1->Socket->SendText(Memo1->Text    );

//Beendigung der Message
  Memo2->Lines->Add(".");
  ClientSocket1->Socket->SendText(".\r\n");
  len=ClientSocket1->Socket->ReceiveBuf(&recv_buf,128);
  Memo2->Lines->Add(((AnsiString)recv_buf).SubString(0,len));
  if (((AnsiString)recv_buf).SubString(0,3)!="250") goto ende;

//--Beenden der Sitzung
ende:
  Memo2->Lines->Add("QUIT");
  ClientSocket1->Socket->SendText("QUIT\r\n");
  len=ClientSocket1->Socket->ReceiveBuf(&recv_buf,128);
  Memo2->Lines->Add(((AnsiString)recv_buf).SubString(0,len));
  ClientSocket1->Close();
}
//---------------------------------------------------------------------------
```

5 Response-Codes

```
CONNECTION ESTABLISHMENT
      S: 220
      E: 554
   EHLO or HELO
      S: 250
      E: 504, 550
   MAIL
      S: 250
      E: 552, 451, 452, 550, 553, 503
   RCPT
      S: 250, 251
      E: 550, 551, 552, 553, 450, 451, 452, 503, 550
   DATA
      I: 354 -> data -> S: 250
                        E: 552, 554, 451, 452
      E: 451, 554, 503
   RSET
      S: 250
   VRFY
      S: 250, 251, 252
      E: 550, 551, 553, 502, 504
   EXPN
      S: 250, 252
      E: 550, 500, 502, 504
   HELP
      S: 211, 214
      E: 502, 504
   NOOP
      S: 250
   QUIT
      S: 221
```