EFT-Komponente

Version 1.0 / Juni 99

Komponente TEFTProtokoll

Die Komponente TEFTProtokoll stellte eine Implementation des genormten EuroFileTransfer-Protokolles dar.

Es können damit sowohl EFT-Clients als auch ein EFT-Server erstellt werden.

Die Komponente ist von der Basiskomponente TCapiProt abgeleitet.

Funktionsumfang

Die Komponente bietet alle notwendigen Funktionen um eine EFT-Sitzung auf- und abzubauen und um Dateien zwischen zwei Gegenstellen zu übertragen..
Dabei kann sie sowohl als EFT-Client als auch als Server eingesetzt werden.

Verbindungsaufbau

Für eine EFT-Verbindung müssen vor dem Aufbau der B-Kanal-Verbindung in der zugehörigen CAPI-Komponente folgende Parameter definiert werden.

B1Protokoll := HDLC_64K
B2Protokoll := X75
B3Protokoll := ISO8208

Bei abgehenden Anrufen muss das Dienstkennzeichen DATA_64k benutzt werden.

Beispiel

```
Capi20Handler1.Call ('123456','',DATA_64K);
```

Client-Funktionen

Nach dem eine ISDN-Verbindung initiiert und von einer Gegenstelle angenommen wurde, übernimmt die Komponente den Aufbau einer EFT-Sitzung.

Sie kann Verzeichnisse (FileStores) auszuwählen und die darin enthaltenen Dateien als Liste anfordern. Dateien können gesendet und empfangen werden. Zusätzlich kann die Gegenstelle Dateien anfordern oder ohne Aufforderung welche schicken.

Server-Funktionen

Nach dem eine ISDN-Verbindung angenommen wurde, übernimmt die Komponente den Aufbau einer EFT-Sitzung.

Sie kann Unterverzeichnisse (FileStores) verwalten, die die Gegenstelle auswählen kann.

Dateien können gesendet und empfangen werden. Zusätzlich kann die Gegenstelle Dateien anfordern oder ohne Aufforderung welche schicken.

Dateiinformationen

Alle Information über den Inhalt von Verzeichnissen werden in einer Liste vom Typ TDirList abgelegt.
Diese Komponente kann stellte Funktionen zum Verwalten und Sortieren von Filelisten zur Verfügung und kann eigenständig Listen von lokalen Verzeichnissen erzeugen.

Eine genau Beschreibung befindet sich im Abschnitt über die Komponente TDirList

Einsatz der Komponente

Server

Der Server wartet auf einen eingehenden Anruf. Dieser wird mit **OnRing** der Komponente TCAPI20Handler gemeldet.
Damit eine EFT-Verbindung zustande kommen kann, müssen vor dem Aufbauen der B-Kanal-Verbindung die Protokolle viel folgt gesetzt werden:

B1Protokoll := HDLC_64K
B2Protokoll := X75
B3Protokoll := ISO8208

Den erfolgreichen Aufbau einer EFT-Verbindung meldet die Komponente über den Event **Connect**.
Dieser ist nicht zu verwechseln mit dem Event **OnConnect** oder **OnBConnect** der Komponente TCAPI20Handler. Diese melden nur das Zustandekommen einer D- bzw. B-Kanal-Verbindung.

Für den Betrieb als EFT-Server ist keine weiteren Funktionen notwendig.

Client

Verbindung aufbauen

Als erstes wird eine Verbindung aufgebaut.

```
EFTProtokoll1.UserName := ‘gast’;
EFTProtokoll1.Password := ‘gast;

Capi20Handler1.Call (FormCallData.Edit1.Text,'',DATA_64K);
```

Wenn der Server die Anmeldung akzeptiert hat, teil dies der Event **Connect** mit, andernfalls wird über den Event **Disconnect**.
Dieser ist nicht zu verwechseln mit dem Event **OnConnect** oder **OnBConnect** der Komponente TCAPI20Handler. Diese melden nur das Zustandekommen einer D- bzw. B-Kanal-Verbindung.

Verzeichnisse empfangen

Wenn eine Verbindung zustande gekommen ist, wird als erstes der Inhalt des Ausgangsverzeichnisses abgefragt.

```
EFTProtokoll1.DirList.SortBy := s_Name;
EFTProtokoll1.RequestDirectory ('*');
```

Im Beispiel wird die Sortierreihenfolge für die Dateien auf den Dateinamen festgelegt.

Immer wenn der Server ein Inhaltsverzeichnis übertragen hat, wird dies durch den Event **DirReceived** signalisiert. Der Inhalt kann dann der Eigenschaft **DirList** entnommen werden. Die Eigenschaft **RemoteDirectory** gibt den Name des Verzeichnisses auf dem Server an.

```
procedure TForm1.EFTProtokoll1DirReceived(Sender: TObject);
var
  i       : Integer;
  dirname : TDirName;
begin
  dirname := EFTProtokoll1.RemoteDirectory;
  LocalDirName.Caption := dirname;

  if (StringGrid1.FixedRows > 0) then begin
    StringGrid1.Cells [0, 0] := 'Dateiname';
    StringGrid1.Cells [1, 0] := 'Grösse';
  end;

  with EFTProtokoll1 do begin
    StringGrid1.RowCount := StringGrid1.FixedRows +  DirList.Count;

    if (DirList.Count > 0) then begin
      for i:= 0 to DirList.Count - 1 do begin
        with DirList.Items [i] do begin
          StringGrid1.Cells [0, StringGrid1.FixedRows + i] := Name;

          if ((Attr and faDirectory) <> 0) then
            StringGrid1.Cells [1, StringGrid1.FixedRows + i] := 'DIR'
          else
            StringGrid1.Cells [1, StringGrid1.FixedRows + i] := IntToStr (Size);
        end;
      end;
    end;
  end;
end;
```

Dateien empfangen

Mit der Funktion **RequestFile** kann eine Datei vom Server geladen werden.
Die Funktion kehrt erst zurück, wenn der Server die Anforderung quittiert hat.
Wenn die eigentliche Übertragung beginnt wird einmalig der Event **GetFile** aufgerufen. Innerhalb der Eventbehandlung kann man der zu empfangenden Datei noch einen eigenen Namen zuteilen.
Sobald alle Daten übertragen wurden ruft die Komponente den Event **FileReceived** auf.

```
function TForm1.EFTProtokoll1GetFile(Sender: TObject; const FileName:
TFileName): TFileName;
begin
  FormShowTransfer.Caption := 'Empfangen ...';
  FormShowTransfer.Gauge1.Progress := 0;
  FormShowTransfer.NameLabel.Caption := FileName;
  FormShowTransfer.SizeLabel.Caption := '';
  FormShowTransfer.Visible := true;

  EFTProtokoll1GetFile := FileName;
end;
```

Dateien senden

Um eine Datei zum Server zu übertragen wird die Funktion **SendingFile** aufgerufen.
Auch diese Funktion kehrt erst zurück, wenn der Server die Anforderung quittiert hat.
Wenn die eigentliche Übertragung beginnt wird einmalig der Event **PutFile** aufgerufen.
Sobald alle Daten übertragen wurden ruft die Komponente den Event **FileSend** auf.

```
function TForm1.EFTProtokoll1PutFile(Sender: TObject; const FileName:
TFileName): TFileName;
begin
  FormShowTransfer.Caption := 'Senden ...';
  FormShowTransfer.Gauge1.Progress := 0;
  FormShowTransfer.NameLabel.Caption := FileName;
  FormShowTransfer.SizeLabel.Caption := '';
  FormShowTransfer.Visible := true;

  EFTProtokoll1PutFile := FileName;
end;
```

Übertragungsverlauf

Im Verlauf der Übertragung wird bei jedem Datenblock der Event **BlockTransfer** aufgerufen. Über die Eigenschaften **RecvFileSize** und **RecvFileCount** bzw. **SendFileSize** und **SendFileCount** kann der aktuelle Stand der Übertragung angezeigt werden.

```
procedure TForm1.EFTProtokoll1BlockReceived(Sender: TObject);
begin
  with (EFTProtokoll1 as TEFTProtokoll) do begin
    if (TransferMode = Receive) then begin
      if (Length (FormShowTransfer.SizeLabel.Caption) = 0) then begin
        FormShowTransfer.SizeLabel.Caption := IntToStr (RecvFileSize);
        FormShowTransfer.Gauge1.MaxValue := RecvFileSize;
      end;

      FormShowTransfer.Gauge1.Progress := RecvFileCount;
    end else begin
      if (Length (FormShowTransfer.SizeLabel.Caption) = 0) then begin
        FormShowTransfer.SizeLabel.Caption := IntToStr (SendFileSize);
        FormShowTransfer.Gauge1.MaxValue := SendFileSize;
      end;

      FormShowTransfer.Gauge1.Progress := SendFileCount;
    end;
  end;
end;
```

Übertragung abbrechen

Durch Aufrufen der Funktion **AbortTransfer** kann eine Dateiübertragung vorzeitig abgebrochen werden.
Der erfolgte Abbruch wird dann über den Event **TransferAbort** mitgeteilt. Dies geschieht auch dann, wenn der Server die Übertragung abgebrochen hat.

Verbindung abbauen

Die EFT-Verbindung wird mit der Funktion **SessionEnd** beendet. Falls zu diesem Zeitpunkt eine Datei übertragen wird, erfolgt eine sofortiger Abbruch der Übertragung.

Public Methoden

constructor Create (AOwner : TComponent);
Der Konstruktor der Komponete.

destructor Destroy;
Der Destruktor der Komponente.

procedure SessionEnd;
Die aktuelle EFT-Sitzung wird beendet.
Die Funktion initiiert nur den Verbindungsabbau, wartet aber nicht darauf.
Der erfolgte Abbau der Verbindung wird dann durch den Event **Disconnect** signalisiert.

function SelectDirectory (const DirName : TDirName) : Integer;
Diese Funktion selektiert ein Unterverzeichnis (FileStor) in der EFT-Gegenstelle.
Wenn die Gegenstelle dies annimmt, kann die Liste der enthaltenen Dateien mit der Funktion **RequestDirectory** abgerufen werden.

Zu Beginn einer jeden EFT-Sitzung muss das Verzeichnis **.** ausgewählt werden. Dies ist das Verzeichnis, welches der Server dem User aus Ausgangsverzeichnis - Home-Verzeichnis - zugewiesen hat.

Folgende Rückgabewerte liefert die Funktion

0 : Alles ok, das neue Verzeichnis wurde ausgewählt.
-2 : Die Gegenstelle verweigert den Zugriff
-8 : Zur Zeit besteht keine Verbindung
-9 : Timeout, die Gegenstelle reagiert nicht.

function RequestDirectory (const DirMaske : TFileName) : Integer;
Den Inhalt des Verzeichnisses mit dem Namen <*DirName*> wird der EFT-Gegenstelle angefordert.
Es gibt keine Überprüfung, ob es sich um ein korrektes Verzeichnis handelt. Der Rückgabewert der Folgende Rückgabewerte liefert die Funktion

0 : Alles ok, die Liste aller Files wird erstellt.
-2 : Die Gegenstelle verweigert den Zugriff
-8 : Zur Zeit besteht keine Verbindung
-9 : Timeout, die Gegenstelle reagiert nicht.

In einem ersten Schritt werden alle Unterverzeichnisse (FileStores) abgefragt und danach, alle Dateien in dem ausgewählten Verzeichnis.

Viele EFT-Server unterstützen keine Unterverzeichnisse (FileStores). In diesem Falle wird die entsprechende Anfrage negativ quittiert.

Der Empfange aller Verzeichnisinformationen wird über den Event ***DirReceived*** signalisierte.
Die empfangene Daten stehen dann in der Eigenschaftte ***DirList***.

function RequestFile (const FileName : TFileName) : Integer;
Die Datei mit dem Namen <*FileName*> wird von der EFT-Gegenstelle angefordert.
Die Funktion kehrt erst zurück, wenn die Gegenstelle die Anforderung quittiert hat.
Folgende Rückgabewerte sind dann möglich:

0 : Alles ok, die Übertragung läuft
-2 : Die Gegenstelle verweigert den Zugriff
-8 : Es besteht keine Verbindung
-9 : Timeout, die Gegenstelle antwortet nicht

Wenn die Gegenstelle die Datei schicken will, signalisiert dies die Komponente über den Event ***GetFile***. Dabei ist es nicht von Bedeutung, ob die Gegenstelle oder die eigene Applikation diese Übertragung initiiert hat.
Wenn der Rückgabewert dieses Eventes nicht ein Leerstring ist, beginnt die eigentliche Übertragung. Dabei wird in ***TransferMode*** der Typ *Receive* hintelegt.

Die Übertragung eines Datenblockes wird über den Event ***BlockTransfer*** signalisiert.

Nachdem alle Daten übertragen wurden, löst die Komponente den Event ***FileReceived*** aus.

function SendingFile (const FileName : TFileName) : Integer;

Die Datei mit dem Namen <*FileName*> wird zur EFT-Gegenstelle übertragen.
Die Funktion kehrt erst zurück, wenn die Gegenstelle die Anforderung quittiert hat.
Folgende Rückgabewerte sind dann möglich:

0 : Alles o.k., die Übertragung läuft
-1 : Die Datei ist nicht vorhanden.
-2 : Die Gegenstelle verweigert den Zugriff
-8 : Es besteht keine Verbindung
-9 : Timeout, die Gegenstelle antwortet nicht

Wenn eine Datei gesendet werden soll, signalisiert dies die Komponente über den Event ***PutFile***. Dabei ist es nicht von Bedeutung, ob die Gegenstelle oder die eigene Applikation diese Übertragung initiiert hat.

Wenn der Rückgabewert dieses Eventes nicht ein Leerstring ist, beginnt die eigentliche Übertragung. Dabei wird in ***TransferMode*** der Typ *Send* hintelegt.

Die Übertragung eines Datenblockes wird über den Event ***BlockTransfer*** signalisiert.

Nachdem alle Daten übertragen wurden, löst die Komponente den Event ***FileSend*** aus.

function AbortTransfer : Integer;

Abbrechen einer Dateiübertragung. Nach erfolgreichem Abbruch wird zusätzlich der Event **TransferAbort** aufgerufen.

Read-Only Properties

Dies Property stehen nur zur Laufzeit zur Verfügung und deren Werte können nur gelesen werden.

property TransferMode : TTransferMode

```
TTransferMode = (None, Send, Receive);
```

Dieses Property legt die aktuelle Übertragungsart fest.
Im Zustand *None* ist zur Zeit keine Übertragung aktive, bei *Send* wird eine Datei zur Gegenstelle übertragen und bei *Receive* eine Datei empfangen.

property EFTMode : TEFTMode

```
TEFTMode = (Offline, Master, Slave);
```

Dieses Property legt fest, ob die eigene Applikation als Server (*Master*) oder als Client (*Salve*) arbeitet.
Wenn keine Verbindung besteht, hat ***EFTMode*** den Zustand *Offline*.

property SlaveOptions : TEFTSlaveOptionSet

Wird zur Zeit nicht benutzt.

property CallUserName : TUserName

```
TUserName = String [12];
```

Der Benutzername des Anrufers.
Dieser kann im Event ***ReqConnect*** zusammen mit dem Property ***CallPassword*** überprüft werden.

property CallPassword : TPassword

```
TPassword  = String [12];
```

Das Passwort des Anrufers.
Dies kann im Event ***ReqConnect*** zusammen mit dem Property ***CallUserName*** überprüft werden.

property RemoteAppName : TEFTAppName

Der Name der EFT-Applikation der Gegenstelle.
Die Gegenstelle schickt diese Information beim Verbindungsaufbau, eine weitere Funktion hat sie nicht.

property RemoteAddress : TEFTAddress

Die Adresse der angerufenen EFT-Gegenstelle.
Die Gegenstelle schickt diese Information beim Verbindungsaufbau, eine weitere Funktion hat sie nicht.

property RemoteDirectory : TDirName

Das aktuelle ausgewählte Verzeichnis in der EFT-Gegenstelle.
Mit der Funktion **RequestFile** können daraus Dateien empfangen werden.

property SendFileSize : Cardinal

Dieses Property gibt zu Beginn einer Dateiübertragung an, wie gross die zu sendende Datei ist.

property SendFileProgress : Cardinal

Dieses Property gibt während einer Dateiübertragung an, wie viele Bytes bereits gesendet wurden.

property RecvFileSize : Cardinal

Dieses Property gibt zu Beginn einer Dateiübertragung an, wie gross die zu empfangende Datei ist.

property RecvFileProgress : Cardinal

Dieses Property gibt während einer Dateiübertragung an, wie viele Bytes bereits empfangen wurden.

property DirList : TDirList

Dies Liste enthält den Inhalt des aktuellen Verzeichnisses auf der EFT-Gegenseite.

Published Properties

property UserName : TUserName
Mit diesem Namen wird sich beim angerufenen EFT-Server angemeldet.

property Password : TPassword
Mit diesem Passwort wird sich beim angerufenen EFT-Server angemeldet.

property OwnAddress : TEFTAddress
Die eigene EFT-Adresse.
Sie wird beim Verbindungsaufbau zur Gegenstelle übertragen, eine weitere Funktion hat sie nicht.

property OwnAppName : String
Der Name der eigenen EFT-Applikation.

property ServerOptions : TEFTSlaveOptionSet
Die möglichen Option die im Serverbetrieb dem anrufenden Client zur Verfügung stehen.
Zur Zeit nicht unterstützt.

property BaseDirectory : TDirName
Dies ist das Ausgangsverzeichnis, nach einem Verbindungsaufbau.
Alle Pfadangaben müssen relativ zu diesem Verzeichnis erfolgen.

property LocalDirectory : TDirName
Das aktuelle Verzeichnis des eigenen EFT-Servers. Diese Pfadangabe ist immer relativ zum Ausgangsverzeichnis ***BaseDirectroy***.

property ReceiveDirectory : TDirName
In dieses Verzeichnis werden empfangene Dateien abgelegt.

Published Events

property Connect : TNotifyEvent

Dieser Event wird immer dann aufgerufen, wenn eine EFT-Verbindung zustande gekommen ist. Dies bedeutet das sich beide Teilnehmer gegenseitig identifizierte haben, und der Server die Kombination Username/Passwort akzeptiert hat.

property Disconnect : TNotifyEvent

Dieser Event wird immer dann aufgerufen, wenn eine bestehende EFT-Verbindung abgebaut wurde.

property ReqConnect : TEFTReqConnectEvent

TEFTReqConnectEvent = function (Sender:TObject) : Boolean of object;

Dieser Event wird immer dann aufgerufen, wenn die Gegenstelle eine EFT-Verbindung aufbauen möchte.
Hier kann z.B. nach der Überprüfung der Kombination Username/Passwort die Verbindung akzeptiert oder abgelehnt werden.
Mit dem Rückgabewert *true* wird die Verbindung angenommen, mit *false* abgelehnt.

property ReqDirectory : TEFTReqDirectoryEvent

TEFTReqDirectoryEvent = function (Sender:TObject; const DirName : TDirName) : TDirName of object;

Die Gegenstelle möchte den Inhalt eines Verzeichnisses (FileStore) anfordern.
Der Name des angeforderte Verzeichnisses wird im Parameter <*DirName*> übergeben.
Das Ausgangsverzeichnis wird immer als ‚*' übergeben, die weiteren Unterverzeichnis (FileStores) mit dem entsprechenden Namen.
Innerhalb des Events kann der zugehörige Verzeichnisname verändert.
Zurückgegeben werden muss ein Verzeichnisname, oder einen leeren String, wenn der Zugriff auf das gewünschte Verzeichnis nicht zugelassen wird.
Die Gegenstelle erhält in diesem Falle eine negative Quittierung.

property DirReceived : TNotifyEvent

Dieser Event wird immer dann aufgerufen, wenn der Inhalt eines Verzeichnisses (FileStore) empfangen wurde. Der Inhalt befindet sich dann in der Liste **DirList**.
Dein Inhalt eines Verzeichnisses kann man über die Funktion **RequestDirectory** anfordern.

property PutFile : TEFTPutFileEvent

TEFTPutFileEvent = function (Sender:TObject; const FileName : TFileName) : TFileName of object;

Dieser Event wird immer dann aufgerufen, wenn eine Datei zur Gegenstelle geschickt werden soll.
Im Parameter <*FileName*> wird der Name der gewünschten Datei - ohne Pfadangabe - übergeben.
Der Rückgabeparameter gibt den Namen der zu übertragenden Datei vor. Dieser kann anders lauten als der gewünscht und eine zusätzliche Pfadangabe enthalten.
Wenn ein Leerstring zurückgegeben wird, erfolgt keine Übertragung und die Gegenstelle erhält eine negative Quittierung.

property GetFile : TEFTGetFileEvent

TEFTGetFileEvent = function (Sender:TObject; const FileName : TFileName) : TFileName of object;

Dieser Event wird immer dann aufgerufen, wenn eine Datei empfangen werden soll.
Im Parameter <*FileName*> wird der Name der gewünschten Datei - ohne Pfadangabe - übergeben. Der Rückgabeparameter gibt den Namen vor, unter welchem die Datei abgespeichert werden soll. Dieser kann anders lauten als der Originalname und eine zusätzliche Pfadangabe enthalten.

Wenn ein Leerstring zurückgegeben wird, erfolgt keine Übertragung durch die Gegenstelle, die in diesem Falle eine negative Quittierung erhält.

property FileReceived : TNotifyEvent

Dieser Event wird immer dann aufgerufen, wenn einer Datei komplett empfangen wurde.

property FileSend : TNotifyEvent

Dieser Event wird immer dann aufgerufen, wenn einer Datei komplett gesendet wurde.

property BlockTransfer : TNotifyEvent

Dieser Event wird immer dann aufgerufen, wenn eine weiterer Datenblock korrekt übertragen wurde. Ob es sich dabei um ein Senden oder Empfangen handelt, kann man an Hand des Properties ***TransferMode*** feststellen.

property TransferAbort : TNotifyEvent

Dieser Event wird immer dann aufgerufen, wenn die Gegenstelle eine Übertragung vorzeitig abgebrochen hat, oder die Funktion **AbortTransfer** aufgerufen wurde.

Komponente TDirList

Die Komponente TDirList Verwaltet den Inhalt eines Dateiverzeichnisses.
Sie bietet Funktionen zum Erstellen und Sortieren der Liste.

Die Liste enthält für jede Datei und für jedes vorhandene Unterverzeichnisse einen Eintrag
vom Typ TDirListEntry.

```
TDirListEntry = record
  Name  : TDirName;
  Size  : LongInt;
  Attr  : Integer;
  Time  : LongInt;
end;
```

Name : Der Datei- oder Unterverzeichnis-Name

Size : Die Grösse der Datei in Bytes, bei Unterverzeichnisses -1.

Attr : Das Attribut des Verzeichnisseintrages.
Zum Testen der einzelne Attribute können die Konstanten aus der Unit SysUtils benutzt werden:

faReadOnly	$00000001	Schreibgeschützte Datei
faHidden	$00000002	Verborgene Datei
faSysFile	$00000004	Systemdatei
faVolumeID	$00000008	Laufwerks-ID
faDirectory	$00000010	Verzeichnis
faArchive	$00000020	Archivdatei

Time : Datum/Uhrzeit des letzten Zugriffes.

Funktionen

constructor Create;
Der Konstruktor.

destructor Destroy;
Der Destruktor.

procedure Clear;
Löschen der gesamten Liste.

function Add (Item: TDirListEntry): Integer;
Hinzufügen eines Eintrages. Der neue Eintrag wird an Hand des Sortierkriterimus **SortBy** in die Liste eingefügt

Eigenschaften

property DirName : TFileName;
Der Name des Verzeichnisses.

property SortBy : TSortType;
Mit dieser Eigenschaft kann die Sortierreihenfolge der Eintrage festgelegt werden.
Sobald die Eigenschaft verändert wird, erfolgt ein Umsortieren der Einträge.

property Count : Integer;
Die Anzahl der Einträge in der Liste. Nach dem Aufruf der Funktion Clear ist sie 0.

property Items [Index: Integer]: TDirListEntry;

Diese Eigenschaft enthält alle Einträge in der Liste.
Der Index geht von 0 bis **Count** - 1.