PyroTrans

(PyroServer, PyroBatch)

File Transfer und Verzeichnis-Synchronisierung
per Modem, ISDN, Netzwerk und Internet

Dieses Dokument beschreibt das grundsätzliche Konzept und die Funktionsweise von PryroTrans und weist auf Features und Möglichkeiten der Software hin. Auf konkrete Befehle und Einzelfunktionen geht dann die Online-Hilfe zu den jeweiligen Programmteilen ein (speziell das HilfeThema Skript-Befehle zum Programm PyroBatch).

Falls Sie PyroBatch für Projekte einsetzen wollen, und Fragen haben, die sich in diesem Dokument oder der Online-Hilfe hier nicht klären lassen, kontaktieren Sie bitte m.schmidt@emtec.com.

Die PyroTrans Idee

PyroTrans dient zum manuell oder automatisiert Austauch von Dateien zwischen Rechnern über ISDN, Modem oder TCP-basiertem Netzwerk (LAN/Intenet). Üblicherweise handelt es sich hierbei um Datenaustausch zwischen einer Zentrale und Filialen, das Programm eignet sich aber natürlich aber auch zum Austausch von Daten zwischen einem Unternehmen und seinen Kunden (z. B. Druckereien/ Satzstudios). Durch die Fähigkeit der automatischen Komprimierung der übertragenen Daten, ist auch der Einsatz für Außendienstmitarbeitern mit GSM Modems denkbar.

Programmteile

PyroServer: Der Serverteil von PyroTrans ist rein passiv. Ein Server wartet auf Anrufe, prüft Benutzerrechte und kann Daten senden bzw. empfangen (und diese automatisch komprimieren). Jegliche Initiative hierfür geht jedoch vom Client aus. Jedem Benutzer kann ein eigenes Verzeichnis zugeordnet werden, mit unterschiedlichen Rechten (schreiben, lesen, löschen, etc.)

PyroServer/**Pro**: Wie PyroServer, jedoch mit erweiterten Features. PyroServer/Pro unterstützt z.B. zusätzlich Datenverschlüsselung und gleichzeitige Verarbeitung von mehreren Benutzern.

PyroClient: Manuelle Steuerung des Dateitransfers. Mit dem Client kann ein Benutzer bei einem Server anrufen, sich anmelden und Dateien zum Server senden und empfangen, bzw. auch Dateien am Server löschen oder umbenennen.

PyroBatch: Automatisierte Steuerung des Dateitransfers, d.h. skriptgesteuerter Anruf und skriptgesteuerte Dateioperationen (senden, empfangen, löschen, umbenennen).

Hierbei ist jedoch wichtig zu verstehen, daß die Begriffe Server und Client nicht notwendigerweise mit den Unternehmensteilen korrespondieren, d. h. es sind auch Szenarios denkbar, in denen die Unternehmenszentrale per PyroBatch auf mehreren PyroServern anruft, die in den Filialen installiert sind. Im Prinzip könnten die Begriffe PyroServer und PyroClient/Batch durch PyroPassiv und PyroAktiv ersetzt werden.

Konfigurationsbeispiele

Übertragung Filialdaten zur Zentrale (Szenario 1):

Beschreibung: In den Filialen einer Firma läuft nachts ein Programm, das die Verkaufsdaten zur Zentrale schickt.

Konfiguration: In der Zentrale läuft PyroServer mit einem Benutzereintrag pro Filiale. In den Filialen wird nachts zeitgesteuert per Windows Scheduler ein PyroBatch-Skript abgearbeitet, das die Daten zur Zentrale schickt.

Übertragung Filialdaten zur Zentrale (Szenario 2):

Beschreibung: Ein Programm in der Firmenzentrale ruft nachts Daten aus den Filialen ab.

Konfiguration: In den Filialen ist PyroServer installiert (mit jeweils einem Benutzereintrag für die Zentrale). In der Zentrale läuft zeitgetsteuert ein Programm, das per PyroBatch-Skript eine Filiale nach der anderen anruft und die Daten empfängt.

Übertragung Außendienstdaten zur Zentrale:

Beschreibung: Außendienstmitarbeiter übertragen nach Kundenbesuchen die Bestellungen oder Besuchsberichte in der Verkaufsabteilung .

Konfiguration: Im Unternehmen läuft ein PyroServer. Die Laptops der Außendienstmitarbeiter sind mit GSM Modems ausgestattet. Nachdem die Verkäufe manuell in Excel erfaßt wurden, wird mit PyroClient manuell im Unternehmen angerufen und die Datei per Drag-and-Drop auf den Server gelegt.

Konfiguration (alternativ) : Im Unternehmen läuft ein PyroServer. Die Laptops der Außendienstmitarbeiter sind mit GSM Modems ausgestattet. Verkäufe werden in einer Datenbankapplikation erfaßt, die einen Menüpunkt enthält, der eine Datei für die Zentrale erzeugt und dann PyroBatch mit einem vorbereiteten Skript aufruft, um die Daten zur Zentrale zu übertragen.

Druckerei/Satzstudios:

Beschreibung: Druckaufträge werden per Datei von Satzstudios in der Druckerei abgeliefert. Kleinere Satzstudios schicken die Daten manuell, große Studios sammeln die Aufträge tagsüber und schicken sie nachts in einer Übertragung.

Konfiguration: In der Druckerei läuft PyroServer. Die kleineren Satzstudios verwenden PyroClient um ihre Aufträge zu schicken. In den großen Studios läuft nachts PyroBatch. Am Server kann entweder nur ein Benutzereintrag existieren (der dann nur Schreibrechte hat, d.h. alle Anrufer können Daten senden, sehen aber immer nur ein leeres Verzeichnis und können keine Dateien lesen), bzw. für jede Firma ein eigener Benutzer-Eintrag mit

Verzeichnis.

Übertragung Filialdaten zur Zentrale (Szenario 3):

Beschreibung: Wie Szenario 1, jedoch überträgt die Zentrale nach der nächtlichen Verarbeitung aktualisierte Warenbestände an die Filialen zurück.

Konfiguration: Sowohl in der Zentrale als auch in den Filialen läuft PyroServer. Zuerst rufen die Filialen zeitgesteuert via PyroBatch den PyroServer in der Zentrale an und übertragen die Verkaufsdaten. Später nachts läuft in der Zentrale ein Programm, das per PyroBatch die Filialen anruft und die Bestände zurücküberträgt.

Überblick zu PyroBatch

Starten von Skripts

PyroBatch-Skripts können entweder manuell vom Programm-Menü aus gestartet werden, oder von der Kommandozeile: `PYROBATCH /EXEC:<skriptname>`

Kurzübersicht der Skript-Befehle

Connect:	Anruf bei einem PyroServer
Disconnect:	Verbindung abbauen
RequestCallback	Rückruf durch den Server veranlassen
LocalChDir:	Verzeichnis wechseln (auf dem PyroBatch Dateisystem)
RemoteChDir:	Verzeichnis wechseln (auf dem PyroServer Dateisystem)
LocalRename:	Datei auf dem PyroBatch Dateisystem umbenennen
RemoteRename:	Datei auf dem PyroServer Dateisystem umbenennen
LocalDelete:	Datei auf dem PyroBatch Dateisystem löschen
RemoteDelete:	Datei auf dem PyroServer Dateisystem löschen
Get:	Datei vom PyroServer holen
Put:	Datei zum PyroServer schicken
GetMove:	Datei vom PyroServer holen und dort löschen
PutMove:	Datei zum PyroServer schicken und lokal löschen
GetDir:	Verzeichnis vom PyroServer holen
PutDir:	Verzeichnis zum PyroServer schicken
GetSync:	Inhalt von Verzeichnissen mit Server abgleichen
GetSync:	Inhalt von Verzeichnissen mit Server abgleichen
LocalExec:	Befehl am lokalen Rechner ausführen
RemoteExec:	Befehl am PyroServer ausführen
OnError:	Fehlerbehandlung festlegen
Milestone:	Eintrag im Meilenstein-Protokoll
ForEach:	Befehl mehrmals bezogen auf Dateinamen ausführen
SetRetry:	Wiederholungsparameter im Fehlerfall einstellen
TerminateAfterSkript:	Skriptende-Verarbeitung festlegen.
Kommentare:	Kommentare können hinter //, # oder ; plaziert werden.

Detailierte Beschreibung zu den Befehlen und ihren Parametern, sowie zur Schreibweise von Skripts generell finden Sie in der Online-Hilfe von PyroBatch.

Script Aufbau

Die PyroBatch Skriptsprache verwendet einen flexiblen Syntax und ermöglicht es die Befehle in einer Form zu schreiben, die Sie bereits aus anderen Programmiersprachen wie Windows Batch-Dateien, Visual-Basic oder Perl kennen.

Die folgenden Zeilen sind jeweils korrekte Schreibweisen des PutMove Befehls:

```
PUTMOVE SALES.DAT SALES2.DAT
PutMove "Sales.dat", "Sales2.dat"
PutMove("Sales.dat", "Sales2.dat")
```

Übergabe von Werten für Platzhalter über die Kommandozeile

Falls für mehrere Anrufe eine gleichartige Verarbeitung benötigt wird, ist es möglich hierfür ein gemeinsames Skript zu verwenden, in dem die variablen Teile der Verarbeitung (z.B. Telefonnummer, Kennwort, Dateinamen) durch Platzhalter wie `$(telefon)` oder `$(kennwort)` ersetzt werden. Die Werte dieser Platzhalter bestimmen sich dann über die Kommandozeile, in dem PyroBatch mit Parameter übergeben werden:

```
PYROBATCH /EXEC:transfer.cmd /D:telefon=5554467855 /D:passwort=secret
/D:hostname=filiale01
```

Das Transfer-Skript könnte dann so aussehen:

```
# File Transfer zu Filialen
TerminateAfterScript 1
Connect $(telefon) pyrotrans $(passwort)
LocalChDir c:\incoming\$(hostname)
RemoteChDir outgoing
GetMove sales.dat $(hostname).dat
Disconnect
```

Fehlerverarbeitung

Falls Befehle zu einem Fehler führen, wird die Skriptverarbeitung sofort beendet und die Verbindung unterbrochen. Der Fehlergrund ist im Ablaufprotokoll dokumentiert. Fehlercodes und Fehlerklassen sind in der Online-Hilfe detailliert beschrieben.

Falls Fehler bei einzelnen Befehlen toleriert werden sollen, kann dem Befehl ein „-“ (Minus) vorangestellt werden (z. B. `-LocalDelete alt.dat` wenn sicherheitshalber eine Datei gelöscht werden soll, die jedoch nicht unbedingt vorhanden sein muß.)

Darüber hinaus kann die Fehlerverarbeitung durch `OnError` beeinflußt werden (siehe Beispiele unten).

Wiederholungssteuerung

Falls bei der Verarbeitung eines Skripts, das per `/EXEC:` von der Kommandozeile gestartet wurde, ein Fehler auftritt, kann über den Kommandozeilenparameter `/RETRY:` festgelegt

werden, daß das Skript wiederholt werden soll.

Alternativ kann innerhalb des Skripts kann einem Befehl ein @ vorangestellt werden, um den Befehl im Fehlerfall zu wiederholen. Die Anzahl der Wiederholungen und die Pause zwischen den Versuchen wird durch den Befehl `SetRetry` eingestellt. Die Verwendung von @ macht jedoch nur bei Befehlen Sinn, nämlich dann, wenn zu erwarten steht, daß der Fehler zu einem späteren Zeitpunkt nicht mehr auftreten wird (z.B. bei Anwahl einer Gegenstelle). Die Standardeinstellung für `SetRetry` ist so, daß keine Wiederholung stattfindet und @ ignoriert wird. Die Wiederholung muß also im Skript explizit eingeschaltet werden.

Protokollierung

Ablaufprotokoll: Für jede Skriptabarbeitung wird ein ausführliches Protokoll erzeugt (Einträge pro Befehl mit Ergebnis und ggf. Zwischenergebnis). Der Aufbau des Protokolls ist so gehalten, daß die Datei sowohl automatisch verarbeitbar, als auch benutzerfreundlich (lesbar) ist. Der Aufbau des Protokolls ist in der Onlinehilfe genau beschrieben.

Meilensteine: Auf Wunsch kann das Skript ein Meilensteinprotokoll generieren, das den Verarbeitungszustand an bestimmten Punkten des Protokolls dokumentiert (z. B. könnte bei einem Skript, das mehrere Filialen anruft, jeweils am Ende der Verarbeitung einer Filiale ein Meilenstein protokolliert werden). Meilensteinprotokolle haben nur wenige Einträge (nämlich genau einen pro Milestone Befehl) und lassen sich damit optimal von der aufrufenden Applikation einlesen und analysieren.

Skript-Verarbeitung mit PyroBatch

Beispielskripts

Filiale/Zentrale (Szenario 1):

Dieses Skript führt einen eine Anruf in der Zentrale durch und speichert die lokale Datei verkäufe.dat dort als filiale1.dat und löscht sie dann auf dem lokalen Rechner.

```
// Anruf in Zentrale, und Login als filiale1
Connect 05553456789 filiale1 geheim

// Upload Datei verkaeufe.dat mit Namen filiale1.dat
LocalChDir c:\outbox
-LocalDelete filiale1.dat
LocalRename verkaeufe.dat filiale1.dat
```

```
Put filiale1.dat
LocalDelete filiale1.dat

// Verbindungsabbau
Disonnect
```

Filiale/Zentrale (Szenario 1), alternativ:

Gleiche Aufgabenstellung, eleganter gelöst.

```
Connect 05553456789 filiale1 geheim
PutMove c:\outbox\verkaeufe.dat filiale1.dat
Disonnect
```

Filialen/Zentrale (Szenario 2):

Zentrale ruft nacheinander in 3 Filialen an und holt die dort als verkaeufe.dat gespeicherten Daten und speichert sie als filiale1/2/3.dat.

```
// Anruf der Zentrale bei Filiale 1
// Download der Datei verkaeufe.dat mit Namen filiale1.dat
Connect 05553456789 firmaxyz geheim
GetMove verkaeufe.dat c:\inbox\filiale1.dat
Disconnect


// Anruf der Zentrale bei Filiale 2
// Download der Datei verkaeufe.dat mit Namen filiale2.dat
Connect 05556666666 firmaxyz geheim
GetMove verkaeufe.dat c:\inbox\filiale2.dat
Disconnect


// Anruf der Zentrale bei Filiale 3
// Download der Datei verkaeufe.dat mit Namen filiale3.dat
Connect 05559876543 firmaxyz geheim
GetMove verkaeufe.dat c:\inbox\filiale2.dat
Disconnect
```

Filialen/Zentrale (Szenario 2) mit Ignorieren von Fehlern (auszugsweise):

Diese Variante stellt sicher, daß Anruf bei allen Filialen versucht wird, und nicht bei Fehler in einer Filiale, der Rest übersprungen wird. Erfolg kann ggf. durch vorhandensein der Dateien c:\inbox*.dat geprüft werden.

```
OnError Ignore

Connect 05553456789 firmaxyz geheim
GetMove verkaeufe.dat c:\inbox\filiale1.dat
Disconnect

    (rest des skripts wie oben)
```

Filialen/Zentrale (Szenario 2) mit Fehlerbehandlung (auszugsweise):

In diesem Beispiel wird `OnError SkipTo` verwendet, um bei einem Fehler in einer Filiale bei der folgenden weiterzumachen. Der OnError Sprung ist so angelegt, daß je nach Online/Offline Status das `Disconnect` verarbeitet wird, oder nicht.

```
// Anruf der Zentrale bei Filiale 1
OnError SkipTo NaechsteFiliale
Connect 05553456789 firmaxyz geheim

OnError SkipTo VerbindungsAbbau
GetMove verkaeufe.dat c:\inbox\filiale1.dat
; hier könnten weitere Online-Befehle stehen
:VerbindungsAbbau
Disconnect



:NaechsteFiliale
// Anruf der Zentrale bei Filiale 2
OnError SkipTo NaechsteFiliale
Connect 0555666666 firmaxyz geheim

OnError SkipTo VerbindungsAbbau
GetMove verkaeufe.dat c:\inbox\filiale2.dat
; hier könnten weitere Online-Befehle stehen
:VerbindungsAbbau
Disconnect
```

```
:NaechsteFiliale
// Anruf der Zentrale bei Filiale 3
OnError SkipTo Ende
Connect 05559876543 firmaxyz geheim

OnError SkipTo VerbindungsAbbau
GetMove verkaeufe.dat c:\inbox\filiale2.dat
; hier könnten weitere Online-Befehle stehen
:VerbindungsAbbau
Disconnect

:Ende
```

Filialen/Zentrale (Szenario 2) mit Fehlerbehandlung (elegant):

In diesem Beispiel wird `OnError SkipTo` verwendet, um bei einem Fehler in einer Filiale mit der darauf folgenden weiterzumachen.
Im Fehlerfall wird immer zur darauffolgenden ":NaechsteFiliale" Marke gesprungen. ":NaechsteFiliale" ist vor dem `Disconnect` platziert, damit im Fehlerfall eine ggf noch. bestehende Verbindung abgebaut wird. Disconnect darf auch verarbeitet werden, wenn keine Verbindung besteht, liefert dann aber Fehler. Deshalb wird ein evtl. Fehler durch `-Disconnect` ignoriert.

```
OnError SkipTo NaechsteFiliale


// Anruf der Zentrale bei Filiale 1
// Download der Datei verkaeufe.dat mit Namen filiale1.dat
Connect 05553456789 firmaxyz geheim
GetMove verkaeufe.dat c:\inbox\filiale1.dat
:NaechsteFiliale
-Disconnect


// Anruf der Zentrale bei Filiale 2
// Download der Datei verkaeufe.dat mit Namen filiale2.dat
Connect 05556666666 firmaxyz geheim
GetMove verkaeufe.dat c:\inbox\filiale2.dat
:NaechsteFiliale
-Disconnect
```

```
// Anruf der Zentrale bei Filiale 3
// Download der Datei verkaeufe.dat mit Namen filiale3.dat
Connect 05559876543 firmaxyz geheim
GetMove verkaeufe.dat c:\inbox\filiale2.dat
:NaechsteFiliale
-Disconnect
```

Filialen/Zentrale (Szenario 2) mit Fehlerbehandlung und Meilensteinen:

Wg. Fehlerbehandlung siehe Kommentar zum vorherigen Beispiel.
Die Milestone-Befehle vor jedem ":NaechsteFiliale" dokumentieren den Fehlerzustand an diesem Punkt. Entweder "#200 OK" vom Transfer, oder den Fehler, der OnError dazu veranlaßt hat, die "NaechsteFiliale" Marke anzuspringen (z.B. #400 BUSY").

```
OnError SkipTo NaechsteFiliale
Milestone Filename transfer.log


// Anruf der Zentrale bei Filiale 1
// Download der Datei verkaeufe.dat mit Namen filiale1.dat
Connect 05553456789 firmaxyz geheim
GetMove verkaeufe.dat c:\inbox\filiale1.dat
:NaechsteFiliale
-Disconnect
Milestone Filiale1


// Anruf der Zentrale bei Filiale 2
// Download der Datei verkaeufe.dat mit Namen filiale2.dat
Connect 05556666666 firmaxyz geheim
GetMove verkaeufe.dat c:\inbox\filiale2.dat
:NaechsteFiliale
-Disconnect
Milestone Filiale2


// Anruf der Zentrale bei Filiale 3
// Download der Datei verkaeufe.dat mit Namen filiale3.dat
Connect 05559876543 firmaxyz geheim
GetMove verkaeufe.dat c:\inbox\filiale2.dat
:NaechsteFiliale
Milestone Filiale3
```