# Package Automation v1.1.4

**Roland Spitzer**

roland@familie-spitzer.de

---

## Inhaltsverzeichnis

## 1. Einsatz des Routers als Heim-Automations-Server

1.1. Starten und Stoppen von Servern, die von einem anderen Gerät abhängig sind. Zum Beispiel starte einen/mehrere Server wenn der Laptop oder der Streaming-Client hochgefahren bzw. runtergefahren werden. (siehe sc200 group). Dies geht mittels Magic-Paket (ether-wake) oder auch durch an-/ausschalten der Stromversorgung (hier muß im BIOS das Booten nach Stromausfall aktiviert werden). Die Stop-Kommandos können mit einer Verzögerung versehen werden, damit nicht bei einem temporären Boot oder ziehen eines Netzsteckers die Clilents beendet werden.

1.2. Beim Hoch-/Runterfahren zusätzliche Kommandos ausführen z.B. net use von Laufwerken. Es können unterschiedliche Kommando-Sequenzen für ein „Start/Stop über die Automation“ oder „Start/Stop manuell“ definiert werden. Zum Beispiel beim manuellen Stoppen kein Abschalten der Stromversorgung. (siehe pc001 stop)

1.3. Reduzierung des Standby-Stromverbrauchs beim Einsatz von z.B. EZControl und Schaltsteckdosen.. (Erfahrungswert ca. 17 Watt pro PC im Standby). Der Router soll das einzige Gerät sein, dass permanent läuft. Vorstellbar wäre natürlich auch ein dediziertes System zur Steuerung. Hierzu später mehr.

1.4. Starten und Stoppen von Servern, wenn bestimmte PC’s nicht mehr zur Verfügung stehen z.B. stop Fli4l wenn alle anderen Systeme Offline sind. Wenn gewünscht steuerbar mit Zeitverzögerung. (funktioniert wie OPT_AUTOSHUTDOWN). (siehe router child)

1.5. Probleme durch ungewolltes Stoppen von Windows-PC’s über Energieverwaltung. Manche Prozesse, die im Hintergrund aktiv sind verhindern nicht die Aktivierung des Ruhezustands. Daher keine Steuerung mehr über Energieverwaltung sondern mittels Automation.

1.6. Automatisierung jeglicher Abläufe mit sofortiger oder über Crontab-gesteuerter Ausführung. Jeder Ablauf bekommt einen Namen z.B. „backup“. Über diesen Namen wird er gestartet bzw. angeprochen. Aus jedem Ablauf/Job wird ein Skript generiert, dass durch die Automation oder manuell gestartet werden kann. Ich nutze dies um über den Router meine Backups unattended auszuführen.

Beispiel : Schalte Strom für PC001 ein, Starte PC001, führe mehrere Backups auf PC001 mit Rsync aus, fahre PC001 runter und Strom aus. Dies läuft crontab-gesteuert täglich um 03.00 Uhr. (siehe pc001 backup)

alle Automations-Skripte liegen auf dem Router und können von außen (irgendeinem PC) über plink gestartet werden.

1.7. Bei Einsatz eines Systems wie T-10 der Fa. Rose und Herleth (www.ezcontrol.de) sind auch diverse andere Schaltungen möglich (bei mir Teichpumpe, Licht, …). p.s. ich will hier keine Werbung für irgendeine Hardware machen, berichte nur von meinen Erfahrungen über Dinge, die ich getestet habe. Das Programm für die Ausgabe von entsprechenden Steuercodes habe ich mit Genehmigung der Fa. R&H für den Router umgewandelt.

1.8. Ebenfalls im Einsatz habe ich das IRTrans LAN-Modul (www.irtrans.de) mit dem ich zusätzlich Infrarot-Befehle an die unterschiedlichsten Geräte abgebe. Auch dieses Modul habe ich kompiliert und mit Genehmigung zur Verfügung gestellt. Mit dem irclient kann ich nun alle gelernten Fernbedienungs-Kommandos über OPT_AUTOMATION ausgeben. D.h. ich kann hier x-beliebige Sequenzen zusammenstellen und mit den Kommandos des T-10 (433 Mhz bzw. 868 Mhz) mischen.

1.9. Unterstützt werden alle Fli4l-Version ab 3.0.x bis 3.1.x

1.10. Neues in Version 1.0.2

- Einsatz von fping anstatt ping. Fping wartet keine 10 Sekunden auf ein Gerät das nicht anpingbar ist.
- Einbau Gerätetypen 0 und 2
- Sleep Parameter zur Beeinflussung der Router-Belastung
- Anpassungen für Fli4l Version 3.1
- Diverse Optimierungen der Skripte (Vermeidung von Subshells durch Pipes und anderes)
- Einbau actorpoll für das Checken der Schaltsteckdosen
- Neue Funktionen wie group_start, group_stop für das ausschließliche Steuern der abhängigen Geräte und only_start und only_stop für das ausschließliche Steuern des Geräts ohne abhängige.
- Erweiterung der http-Oberfläche
- Setzen von Variablen in den generierten Skripten, damit diese Variablen zur Steuerung mit genutzt werden können. HOST beinhaltet den Hostnamen des entsprechenden Geräts, FUNKTION die aufgerufene Funktion und STATUS den aktuellen Status. Somit können diese Parameter im Conf-File z.B. in einem if-Konstrukt mit eingebaut werden.

1.11. Neues in Version 1.0.3

- Einbau Parameter AUTOMATION_AUTO_HOSTS. Wenn dieser Parameter auf “yes“ steht werden zusätzlich die Hosts aus /etc/hosts übernommen, die noch nicht überwacht werden.

- Mehrsprachigkeit für Web-Oberfläche und ein „Versuch“ die Doku ins Englische zu übersetzen. Wenn man mir Alternativen in anderen Sprachen zur Verfügung stellen will, bin ich gerne bereit diese aufzunehmen. Dies gilt sowohl für die Konstanten im Bereich lang/ als auch für die Doku.

- Beschreibung eines alternativen Verfahrens bei Windows-Clients mittels netcat wenn ssh zu umständlich ist.

## 1.12. Neues in Version 1.1.0

- Einbau Gerätetyp 3 (Logfile-Adapter)

- Einbau Logfile-Adapter für beliebige Logfiles. Das Suchkommando wird, um eine größtmögliche Flexibilität zu behalten, mit grep und cut zusammengesetzt. Hier wird das Kennen dieser Unix-Befehle vorausgesetzt, wird aber auch anhand von Beispielen gezeigt. Man kann die maximale Anzahl Aktionen pro Stunde und pro Tag bestimmen um ungewollte Massen-Verarbeitungen zu verhindern. Ansonsten wird ein logfile-Adapter bzgl. Start, stop und maintenance wie jedes andere Gerät behandelt.

## 1.13. Neues in Version 1.1.1 / 1.1.2 (bitte die Doku mal mit 1.1.1 durchsuchen)

- Überarbeitung unterschiedlicher Skripte z.B. automation.cgi in verschiedene Files aufgeteilt. Start- und Stop-Themen in eigene Skripte ausgelagert. Ausbau kleinerer Fehler.

- Stop-Skript in rc0.d/rc511.automation eingebaut und somit Ausführung bestimmter Funktion beim Stoppen des Routers.

- Einbau einer Sicherungs-/Wiederherstellungs-Möglichkeit der Status-Files. So bleiben alte Stati erhalten trotz Router-Neustart.

- Einbauen „default“-Kommando mit Eingabemöglichkeit im mini-httpd.

- Möglichkeit in einem Kommando Logging an- und auszuschalten.

- Erweiterung CRON-Feld mit der Möglichkeit Aktionen bei boot, shutdown des Routers oder start bzw. stop der Automation auszuführen.

- C-Programm irlogger zum Empfang von IR-Kommandos mittels IRTrans und schreiben eines Log-Files.

## 1.14. Neues in Version 1.1.3

- Anpassung an fli4l v3.1.2. Da sich die Filesystem-Struktur mit 3.1.2 geändert hat, sind die ssh-files nicht mehr unter /opt/etc/ssh sondern unter /etc/ssh abgelegt. Um hier variabel zu sein wird der Parameter SSH mit dem korrekten Pfad gefüllt.

- Umfangreiche Korrekturen bei den Aktionen bei boot, shutdown des Routers oder start bzw. stop der Automation.
  (Mitwirkung und Test durch T.Albrecht)

- Neuer Parameter AUTOMATION_REFRESH um automatischen refresh der Host-Anzeige

1.15. Neues in Version 1.1.4

- Anpassung in der mini-httpd Anzeige

- Mehrere CRON-Statements in einem Parameter getrennt durch Semikolon.

- Neuer Parameter http. Somit kann aus dem mini-httpd in der Host-Anzeige eine beliebige http-Seite für jeden Host hinterlegt und aufgerufen werden.

- Neues Programm dlogger zur Überwachung von Directories. (Kernel muß mit DNOTIFY yes erstellt werden).

- IRTRANS Parameter entfernt und Start/Stop über automation.txt (boot/shutdown).

- automation_batch.cgi mit Status-Abfrage ergänzt.

---

## 2. Allgemeines

2.1. Die Kommunikation zwischen den PC's erfolgt bei mir generell mittels ssh. Hier habe ich entsprechende Tools im Einsatz. Gute Erfahrung habe ich mit dem ssh-Server copSSH auf den Windows-Systemen gemacht (copSSH ist über SourceForge.net zu beziehen). Ich nutze key-Files ohne passphrase so entfällt die User-ID und Passwort-Abfrage. Wenn hier bedenken bestehen, kann man key-Files generieren, die nur das gewollte Kommando zulassen.

2.2. Alle Stati werden in Dateien abgelegt. Es gibt zwei Directories. Erstens /var/run/automation/pcs. Hier steht für jeden PC eine Datei mit dem Status einem Datum und einem Timestamp der letzten Statusänderung. Zweitens /var/run/automation/maintenance. Hier steht für jeden PC der im Mantenance-Status ist eine Datei.

2.3. Ich habe zur Übersichtlichkeit das Ganze in mehrere kleine Skripte gebaut, die in /usr/local/bin/automation stehen. Hier werden auch die generierten abgelegt.

2.4. Für das Stoppen der Windows-PS's setze ich psshutdown.exe von sysinternals ein.

2.5. Das Paket ersetzt meines Erachtens die Pakete OPT_AUTOSHUTDOWN und OPT_WOL. Dies war nicht mein Ziel. Da diese Funktionen gut zu integrieren waren und für die Automation wichtig sind, habe ich mich entschieden alle Funktionalitäten in einem Paket zusammenzufassen.

2.6. Vielleicht noch ein Hinweis. Mir ist klar, dass das OPT nicht ganz so einfach ist (kann stark parametrisiert werden muss aber nicht). Bei Rückfragen und konstruktiven Vorschlägen bitte Mail an mich. Da das OPT noch recht jung ist denke ich an Erweiterungen wie z.B. Einbau eines Logfile-Adapters, der aufgrund von Einträgen in Logfiles Aktionen anstößt oder auch die Anbindung eines Empfängers wie R-10 von EZControl (im Moment nicht verfügbar) oder auch FHZ1000 um auch andere Zustände abfragen zu können.

2.7. (Doku ergänzt durch T.A.)
***Die Ausführung von Scripten ist vom Erkennen eines Zustandswechsels abhängig.***
*D.h. ist zum Beispiel ein Host einer "Master"-Gruppe bereits gestartet und wird beim Booten des fli4l-Routers mit dem Startzustand ONLINE erfasst, so wird der abhängige Host nicht nachgestartet. Erst wenn der Zustandswechsel offline-->online eines Mitglieds der "Master"-Gruppe des abhängigen Host erkannt wird, wird auch der abhängige Host gestartet. Der abhängige Host kann auch über den Mini-httpd manuell nachgestartet werden.*

---

## 3. Variablen

### OPT_AUTOMATION

Automation aktivieren?

### AUTOMATION_START_AT_BOOT_TIME

Automation beim Booten des Routers starten?

### AUTOMATION_LOG

Auf "yes" setzen wenn das Logging aktiviert werden soll. Auf "no" wenn nicht geloggt werden soll. Wird "yes" eingestellt, wird die Logdatei über den Webserver bereitgestellt.

### AUTOMATION_DEBUG

Auf "yes" setzen wenn das Debugging aktiviert werden soll. Auf "no" wenn nicht debugged werden soll. Debug schreibt den Output ebenfalls in die Logdatei , es werden nur viel mehr Informationen weggeschrieben.

### AUTOMATION_LOGFILE

Gibt den absoluten Pfad für die Logdatei an. Kann auf die RAM-Disk, Festplatte oder CF-Karte verweisen.

### AUTOMATION_WORK

Hier wird der Pfad für die workfiles gesetzt. Sollte auf  '/var/run/automation' gesetzt werden.

**AUTOMATION_WORK_SAVE (neu ab v1.1.1)**

Hier wird der Pfad zum Sichern der workfiles gesetzt z.B '/data/automation' (Festplatte, CF-Karte). Somit bleiben die System-Stati auch beim Reboot erhalten.

**AUTOMATION_IRTRANS_LOGFILE (ab v1.1.4 nach boot/shutdown übernommen)**

**AUTOMATION_IRTRANS_PORT (ab v1.1.4 nach boot/shutdown übernommen)**

**AUTOMATION_SLEEPH (neu ab v1.0.2)**

Anzahl Sekunden, die nach jedem ping gewartet werden soll. Dies verringert die Last auf dem Server. Wenn AUTOMATION_SLEEPV='0' dann ist die Last gleichmäßig ohne Peeks. Default bei Fli4l-Version 3.0.x ist (1) bei Version 3.1 (0) (performt besser).

**AUTOMATION_SLEEPV (neu ab v1.0.2)**

Anzahl Sekunden, die nach jedem Durchlauf (nach einmal ping aller Server) gewartet werden soll. Bei gesetztem Wert und AUTOMATION_SLEEPH='0'  ist der ping-Durchlauf schneller (kurze peeks) und danach wird gewartet entsprechend dem hier eingetragenen Wert. Default ist "0".

**AUTOMATION_REFRESH (neu ab v1.1.3)**

Anzahl Sekunden nachdem die Host-Anzeige im mini-http neu angezeigt wird.

**AUTOMATION_AUTO_HOSTS (neu ab v1.0.3)**

Ergänzt die zu überwachenden Geräte mit den Einträgen aus /etc/hosts. Default ist "no".

**AUTOMATION_HOSTS_N**

Anzahl der zu überwachenden Hosts

**AUTOMATION_HOST_%**

Hostname oder IP-Adresse des zu überwachenden Hosts

**AUTOMATION_HOST_%_DELAY**

Zeit in Sekunden. Dieser Parameter hat eine Doppelbedeutung. Erstens gibt er den Zeitraum an, der gewartet wird bis abhängige Hosts (GROUP) heruntergefahren werden. Somit kann verhindert werden, dass bei einem Reboot auch die abhängigen PC's sofort gestoppt werden. Zweitens gibt man hier die Zeit an nach der z.B. Fli4l runtergefahren wird, wenn kein Host mehr verfügbar ist (CHILD). Default ist "0".

**AUTOMATION_HOST_%_BOOTTIME**

Zeit in Sekunden. Hier sollte die ungefähre Boot-Zeit eingetragen werden. Das wait-Skript wartet die angegebene Zeit beim Start/Stop bis es weitere Start-/Stop-Befehle nach dem

Boot oder Herunterfahren ausführt. Wenn mehrere Systeme parallel gestartet werden und diese voneinander abhängig sind, kann mit dem wait-Skript die Abhängigkeiten geprüft werden. Default ist “0“.

**AUTOMATION_HOST_%_HTTP (neu ab v1.1.4)**

Hier kann eine URL eingetragen werden, die aus der Host-Anzeige aufgerufen werden kann.

**AUTOMATION_HOST_%_TYPE (neu ab v1.0.2)**

Unterscheidung auf Host-Typen.

- Type 0 wird nicht angepingt/angepollt. Ein Gerät das nicht überwacht werden kann (virtuell). Setze ich für die Geräte ein, die ich mittels Infrarot steuere (IRTrans).
- Type 1 ist ein normales Netzwerk-Gerät das mittels fping überwacht wird.
- Type 2 ist ein Schalter (EZcontrol) der mittels actorpoll überwacht wird.
- Type 3 ist ein Logfile-Adapter der mittels “ps ax“ überwacht wird. (neu ab v1.1.0)

Mit der Einführung von Type 0 und 2 kann man ein System das mittels Infrarot oder EZcontrol geschaltet wird als normales Gerät definieren. D.h. Unterstützung von Group, Child, Verzögerung usw. Unter einem Type 0 (virtuelles Gerät) kann man auch einfach eine Reihe von anderen Kommandos sammeln. Default ist “1“.

**AUTOMATION_HOST_%_TYPE0_DEFAULT (neu ab v1.0.2)**

Nur für Gerätetyp 0 vorgesehen. Setzt den Status des Gerätes beim Start der Automation auf „start“ oder „stop“.

**AUTOMATION_HOST_%_TYPE2_SERVER (neu ab v1.0.2)**

Nur für Gerätetyp 2 vorgesehen. Hier wird der Name oder IP-Adresse des EZcontrol-Servers eingetragen, der diese Steckdose schaltet.

**AUTOMATION_HOST_%_TYPE2_PARM (neu ab v1.0.2)**

Nur für Gerätetyp 2 vorgesehen. Nummer der Schaltsteckdose im EZcontrol (1-20).

**AUTOMATION_HOST_%_TYPE3_FILE (neu ab v1.1.0)**

Nur für Gerätetyp 3 vorgesehen. Pfadname des zu überwachenden Logfiles z.B. /var/log/syslog.log

**AUTOMATION_HOST_%_GROUP_N**

Anzahl der abhängigen Hosts, die parallel gestoppt/gestartet werden. Default ist “0“.

**AUTOMATION_HOST_%_GROUP_%**

Name oder IP-Adresse des abhängigen Hosts.

**(Doku ergänzt durch T. A v1.1.1)**

AUTOMATION_HOST_2_GROUP_N='1'
AUTOMATION_HOST_2_GROUP_1='Server'

*Über die Parameter AUTOMATION_HOST_2_GROUP_X='xxxx' trage ich den Host Host_2 in eine "Master"-Gruppe eines anderen Host´s (hier Server) ein. Hierbei wird der abhängige Host (hier Server) durch seine "Master"-Gruppe folgendermassen gesteuert:*

*Start: der abhängige Host (hier Server) wird gestartet, wenn der Host seiner "Master"-Gruppe online geht*
Frage: ab wann wird ein Host der "Master"-Gruppe als online angezeigt?
*(ein Host der "Master"-Gruppe wird als online gewertet, nach dem ersten erfolgreichen ping auf ihn)*

*Ende: der abhängige Host (hier Server) wird gestoppt, wenn sein Host der "Master"-Gruppe gestoppt wird er (hier Server) in keiner anderen*
*“Master“-Gruppe mehr definiert ist oder in keiner anderen Gruppe mehr gebraucht wird.*

**AUTOMATION_HOST_%_CHILD_N**

Anzahl der Child-Hosts. Wenn alle Child-Hosts beendet sind wird der Parent nach dem Delay heruntergefahren. Default ist “0“.

**AUTOMATION_HOST_%_CHILD_%**

Name oder IP-Adresse des abhängigen Child-Hosts.

**(Doku ergänzt durch T. A v1.1.1)**

AUTOMATION_HOST_1='router'
AUTOMATION_HOST_1_CHILD_N='2'
AUTOMATION_HOST_1_CHILD_1='pc001'
AUTOMATION_HOST_1_CHILD_2='pc002'

*Über die Parameter AUTOMATION_HOST_1_CHILD_X='xxxx' kann man nur das STOPPEN des Host (hier router) in Abhängigkeit von den Child´s steuern. Hierbei wird der Host gestoppt, wenn alle Child´s offline sind und die Delay-Zeiten aller Child´s NULL sind.*

*Child1: wird heruntergefahren 10.23 Uhr und 26 sek, hat ein Delay von 600 sek -> damit ist der Child1 um 10.33 Uhr und 26 sek offline Child2: wird heruntergefahren 10.25 Uhr und 30 sek, hat ein Delay von 50 sek -> damit ist der child2 um 10.26 Uhr und 20 sek offline*
***---> obwohl Child1 vor Child2 runtergefahren wird, wird der Host erst um 10.33 und 26sek (also nach Ablauf des Delay´s von Child1) gestoppt***

*!!!Gestartet werden muss der Host (hier router) manuell.!!!*
*Ausserdem ist der Parameter AUTOMATION_HOST_X_BOOTTIME des Host´s (hier Router) zu beachten. Auch wenn alle Bedingungen für*
*ein STOPPEN des Host´s (hier router) erfüllt sind, wird er erst gestoppt wenn er vollständig gebootet hat ( BOOTTIME wurde auf NULL heruntergezählt).*

**AUTOMATION_HOST_%_PORT_N**

Neben der Überwachung der Child-Hosts kann man hier bestimmte Ports prüfen. Solange diese Ports aktiv sind wird der Parent-Host nicht gestoppt (auch erst nach Ablauf der Delay-Zeit). Default ist “0“.

**AUTOMATION_HOST_%_PORT_%**

Der zu überwachende Port.

**AUTOMATION_HOST_%_COMMAND_N**

Anzahl der Kommandos für diesen Host. Die Kommandos „start“ und „stop“ haben eine besondere Bedeutung. Diese beiden werden bei den automatischen Start-/Stop-Aktionen gesteuert über Group und Child als Start- und Stop-Sequenz genutzt. Ebenso haben die Kommandos „manual_start“ und „manual_stop“ eine besondere Bedeutung. Diese Kommandos werden bei manuellen Aktionen aufgerufen. Z.B. beim automatischen stoppen mit ausschalten der Steckdose beim manuellen nicht. Keins dieser vier Kommandos muß definiert sein, wenn keine besondere Aktion gebraucht wird, hier generiert das OPT Default-Skripte.

Ansonsten ist jede Kommando-Sequenz vorstellbar.  Da ich aus jedem Befehl ein Skript in das automations-Verzeichnis generiere, sind hier ganz allgemeine bash-Befehle möglich (z.B. sleep 10 oder if-Sequenzen, …). Was praktisch ist, man kann hier natürlich auch jedes Skript der Automation selber einbauen. Bei einem backup-Skript für mehrere PC’s werden natürlich auch die „start“ bzw. „stop“-Sequenzen der jeweiligen Systeme, die jeweils in der Automation für diese Systeme definiert sind genutzt. Default ist “0”.

**AUTOMATION_HOST_%_COMMAND_% (ergänzt ab v1.1.1)**

Name des Kommndos z.B. start, stop oder backup.
(Neu ab v1.1.1)  ist das Kommando “default“. Ähnlich wie “start“ und “stop“ hat das Kommando “default“ eine eigene Bedeutung. Um die Anzahl der Definitionen zu reduzieren, kann man für Befehle die fast gleich lauten einen “default“-Eintrag machen. Der übergebene Parameter ist in der Variablen $VALUE abgelegt. (siehe Beispiel). Wählt man im mini-http das Kommando “default“ aus wird man aufgefordert den Wert einzugeben kann.

**AUTOMATION_HOST_%_COMMAND_%_CRON (ergänzt ab v1.1.1)**

Crontab-String mit dem das angegebene Kommando in die Crontab übernommen wird. Hier können ab Version 1.1.1 folgende Strings angegeben werden. „boot“ zum Ausführen des Kommandos beim Boot des Routers, „shutdown“ beim Runterfahren, „automation_stop“ beim Stoppen und „automation_start“ beim Starten der Automation. Wollte keinen zusätzlichen Parameter hierfür verwenden, passt aber ganz gut hierhin.

(Neu ab v1.1.4) mehrere Cron-Parameter in einem Statement

**AUTOMATION_HOST_%_COMMAND_%_SEARCH (neu ab v1.1.0)**

Search-String für die Logfile-Verarbeitung Type 3. Hier kann man eine Reihe von grep und cut-Befehle hinterlegen, um bestimmte Inhalte aus dem Log zu filtern. Bei den Beispielen mehr.

**AUTOMATION_HOST_%_COMMAND_%_COUNT (neu ab v1.1.0)**

Count-String für die Logfile-Verarbeitung Type 3. Hier kann man festlegen wie viele Aktionen von diesem Logfile-Adapter ausgewertet werden können. Auch hier in den Beispielen mehr.

**AUTOMATION_HOST_%_COMMAND_%_N**

Anzahl der Kommandozeilen, die zu diesem Befehl gehören. Default ist “0“.

**AUTOMATION_HOST_%_COMMAND_%_%**

Komandozeile.
(Neu ab v1.1.1) Mit dem Kommando LOG=“no“ oder LOG=yes kann das Logging temporär an bzw. abgeschaltet werden. Dies ist hilfreich wenn man längere Kommando-Sequenzen (ganze Skripte) definiert.
Ein Tip für ein ein Skript, das permanent laufen soll :

```
AUTOMATION_HOST_1_COMMAND_1_N='7'
AUTOMATION_HOST_1_COMMAND_1_1='LOG="no"'
AUTOMATION_HOST_1_COMMAND_1_2='(while [ true ]'
AUTOMATION_HOST_1_COMMAND_1_3='do'
AUTOMATION_HOST_1_COMMAND_1_4='    fping www.t-online.de >/dev/null
2>&1'
AUTOMATION_HOST_1_COMMAND_1_5='    sleep 60'
AUTOMATION_HOST_1_COMMAND_1_6='    fping www.vodafone.de >/dev/null
2>&1'
AUTOMATION_HOST_1_COMMAND_1_7='done) &'
```

Bitte auf die Klammer-auf vor dem while und am Ende Klammer-zu und Ampersand beachten. Hiermit wird das Skript im Hintergrund ausgeführt.

---

## 4. Konfigurations-Beispiele

Anmerkungen zu Definitionen anhand von Beispielen

4.1. Bei den Definitionen für den Router (fli4l) selbst ist folgendes anzumerken.

- Eine sehr hohe Delay-Zeit von 86400 Sekunden was 24 Stunden entspricht. Hiedurch wird erreicht, dass erst nach dieser Zeit der Router heruntergefahren wird, wenn kein PC mehr aktiv ist.

- Alle im Netz verfügbaren PC's werden als Child definiert. D.h. wenn keines dieser Geräte mehr aktiv sein sollte wird nach der Delay-Zeit der Router beendet unter weiterer Berücksichtigung der angegebenen Ports.

- Ebenso wird der Router nicht beendet solange eine Verbindung über Port 22 (ssh) aktiv ist.

- Sobald sich eins der angegebenen Systeme innerhalb der Delay-Zeit meldet zählt die Zeit neu.

4.2. Definitionen für einen W2K-Server (pc001)

- Der Server braucht ca. 2 Minuten Bootzeit (mit etwas Puffer 180 Sekunden).

- Es sind drei Kommandos definiert. Der Start geht über das Anschalten der Steckdose. Muß aber nicht so sein. Ich hatte zu Beginn keine Schaltsteckdosen, dann steht hier ein entsprechendes ether-wake mit der Mac-Adresse des PC's. PC muß natürlich entsprechende Wake-On-Lan-Funktionen unterstützen. Beim Stop setze ich auf psshutdown.exe von www.sysinternals.com. Danach warte ich bis der PC runtergefahren ist und schalte dann wiederum die Steckdose aus (muß ebenfalls nicht sein). Bevor jeder PC heruntergefahren wird, wird geprüft ob er in einer anderen Gruppe definiert ist und der Gruppen-Owner ebenfalls aus ist. In diesem Beispiel wird der Server nicht gestoppt wenn der Laptop (pc004) zwischenzeitlich hochgefahren wurde.

- Das Backup ist hier interessant. Es wird in die Crontab eingetragen und läuft täglich um 3.00 Uhr. Es wird der zuvor definierte Start-Befehl ausgeführt, gewartet bis der PC oben ist und in den Maintenance-Mode versetzt damit nichts die Sicherung stören kann. Danach werden über ssh mehrere (Beispiel vereinfacht) Sicherungen nacheinander angestoßen. Wenn fertig wieder Maintenance-Mode aus und PC runterfahren wie gesagt unter Berücksichtigung der anderen Abhängigkeiten.

4.3. Der Linux-Server

- Beim Linux-Server ist zu erwähnen, dass beim Start auch auf den W2K-Server gewartet wird. Danach werden Samba-Verzeichnisse auf dem W2K mittels Batch-File (net use ….) online gesetzt. Ebenfalls werden mounts abgesetzt für Devices bei denen der mount auf dem Linux-System über fstab zu früh kommt (LVM oder auch USB-Devices). P.s. ich habe an dieser Schaltsteckdose mehrere Geräte parallel angeschlossen, neben dem PC noch unterschiedliche USB-Platten, die ebenfalls mit hochgefahren werden.

4.4. Geräte die manuell geschaltet werden wie z.B. der Laptop oder Showcenter

- Bei diesen Geräten Laptop (Ethernet oder WLAN) oder auch Streaming-Client ist zu beachten, dass diese Geräte einen oder mehrere Server brauchen, da die Daten auf den Servern liegen (siehe die group-Definitionen).

4.5. Schalten von Steckdosen mittels EZControl

- Bei diesem Beispiel wird das für Fli4l kompilierte EZconsole genutzt um z.B. die Steckdose für den W2k-Server zu schalten. Diese Befehle sind Bestandteil in der Start-/Stop-Sequenz zum pc001. Dies dient nur als Beispiel. Die Automation funktioniert natürlich auch ohne andere Hardware. Es war für mich nur eine gelungene einfach integrierbare Ergänzung. Der Vorteil des T-10 liegt meines Erachtens in den umfangreichen Nutzungsmöglichkeiten. Ich schalte sowohl Teichpumpe, Zirkulationspumpe, die PC's als auch meine AUDIO-Komponenten. InVerbindung mit dem EZcontrol (T-10) kann man hier schöne Sachen bauen.

(p.s. in der aktuellen C't von 01/2007 steht, dass das T-10 nicht verfügbar sei. Ich habe es vor zwei Wochen Online bestellt, kein Problem)

4.6. Übertragung von Infrarot-Befehlen durch IRTrans

- Dieses Beispiel ist die Einbindung eines IRTrans mit dem Infrarot-Befehle abgesetzt werden können. Dies ist ebenfalls eine weitere schöne Möglichkeit die Automation auszubauen. Die Menge an Fernbedienungen (meine Frau kommt gar nicht mehr klar) im Wohnzimmer war mir immer ein Gräuel. Ich habe durch die Zusammenfassung unterschiedlicher Befehle das Ganze vereinfacht und über eine HTML-Seite integriert (siehe automation_batch.cgi). Das Beispiel zeigt nur einen kleinen Auszug meiner Definitionen für das Beispiel „radio" hören. Man lernt natürlich vorher die Fernbedienungen (dies wird aber in den entsprechenden Dokumentationen erklärt). Auch hier habe ich das irclient mit Genehmigung für fli4l kompiliert und zur Verfügung gestellt.

- (Neu ab v1.1.1) Für das Empfangen der Infrarot-Befehle des IRTrans Ethernet mit Datenbank wurde ein C-Programm erstellt, dass auf einen angegebenen Port hört (AUTOMATION_IRTRANS_PORT) und die empfangenen Infrarot-Kommandos in ein Logfile (AUTOMATION_IRTRANS_LOGFILE) schreibt. Dieses Logfile kann dann mittels Logfile-Adapter (Type 3) ausgewertet werden und somit jede beliebige Aktion mittels Fernbedienung angestoßen werden. Ich nutze es z.B. um Funksteckdosen (EZ-Control) über eine Fernbedienung zu schalten.
  Mit dem Befehl „killall –HUP irlogger" wird ein Event ausgelöst, der das Programm veranlasst ein neues Logifile zu eröffnen, und das alte Log mit einem Timestamp zu versehen. Das Programm wird gestartet wenn AUTOMATION_IRTRANS_LOGFILE gesetzt.

- (Neu ab v1.1.4) Kein Starten des irlogger mittels AUTOMATION_IRTRANS_... Parameter, sondern Start/Stop Befehle in den Bereich boot/shutdown verlagert.

4.7. Beispiel für ein Type 0-Gerät AV-Receiver

- Dieses Beispiel ist für ein Type0-Gerät, dass nicht überwacht werden kann. Es handelt sich um einen AV-Receiver, der mittels Infrarot und IRTrans Befehle erhält. Man kann alle Funktionen wie Groups, Childs nutzen. Kann den Zustand jedoch nicht prüfen. Man kann auch die dem Gerät zugehörigen Infrarot-Befehle unter dem Gerät definieren.

4.8. Beispiel für ein Type 2-Gerät Schalter

- Hier ist ein Type2-Gerät eine Schaltsteckdose definiert, die am EZcontrol T1001 hängt und die Adresse 5 hat. Hier kann man sehen das alle Funktionen unterstützt werden. Schalte abhängige Steckosen schalter1 und schalter2 parallel. Schalte schalter1 aus wenn 2 und 3 ausgeschaltet sind mit einem Delay von 20 Sekunden. Die Überwachung erfolgt hier mittels actorpoll von EZcontrol. Das Ganze funktioniert natürlich nur dann, wenn keine manuellen Schaltvorgänge über eine separate Fernbedienung vorgenommen werden.

4.9. Beispiel für ein Type 3-Gerät Logfile

- In diesem Beispiel wird das syslog überwacht, deshalb bei Hostname “syslog“ gewählt. Ein Logfile-Adapter kann eine beliebige Anzahl von Strings suchen. Jedes Suchkommando bekommt einen Namen im Beispiel “ip“ oder “test“ und jedem Kommando können beliebige Befehls-Zeilen zugeordnet werden, so wie auch bei anderen Geräten üblich. Die Verarbeitung nutzt das tail-Kommando, beginnt also immer am Ende der Datei. Tail muß über das **OPT_TAIL** installiert werden (Bestandteil des Tools-Package). Sobald das Log fortgesetzt wird, springt der Logfile-Adapter an und wertet die Zeilen aus. Wenn das Suchmuster passt werden die entsprechenden Befehle ausgeführt. Die Type 3-Geräte werden auch in der http-Oberfläche wie andere Geräte gehandhabt.

- Als Beispiel für ein Suchkommando nehme ich folgende Zeile. Angenommen das Log hat einen Eintrag

  „**Feb 18 15:01:37 router dropbear[18158]: Child connection from 192.168.0.40:3389**’

  und Sie wollen eine Zeile in eine Datei schreiben bei jedem Connect zum Router. Dann filtern Sie diese Zeilen mit **grep “Child connection from“**. Nun wollen Sie die IP-Adresse des PC’s. Dies geschieht mit **cut –f 9 –d ’ ’**. Dieser Befehl trennt durch Leerzeichen und filtert das neunte Wort. Als Ergebnis erhalten Sie 192.168.0.40:3389. Nun noch den Port abschneiden mit **cut –f 1 –d ’:’** und fertig. Wenn man nun das Ganze mit dem Pipe-Zeichen „|“ zusammenbaut ist der Befehl komplett.

  AUTOMATION_HOST_%_COMMAND_%_SEARCH='grep "Child connection from” | cut -f 9 -d " " | cut -f 1 -d ":'"

  Es ist zu empfehlen sich auf dem Router anzumelden und das Ergebnis zu testen. z.B. cat /var/log/syslog.log | 'grep "Child connection from” | cut -f 9 -d " " | cut -f 1 -d ":". Wenn das Ergebnis stimmt kann man es übernehmen.

- Als Beispiele für Counter nehme ich folgenden String “* 20“. Der erste Wert ist maßgebend für die Stunde, der zweite für den Tag. * bedeutet beliebig oft in der Stunde aber max. 20 mal am Tag. Oder “3 *“ bedeutet max. drei mal pro Stunde ohne Tagesbegrenzung. So kann die Anzahl der Ausführungen begrenzt werden.

AUTOMATION_HOST_%_COMMAND_%_COUNT='3 *’

- Um mit dem gefundenen String was zu machen steht er in der Variablen $VALUE zur Verfügung. So kann man mit einem Befehl z.B. echo $VALUE >> /tmp/myfile die Ergebnisse in eine Datei schreiben.

- Die „start“ und „stop“-Sequenzen aus dem Beispiel müssen angepasst werden. Hier ist der Hostname '$BIN/logfile.sh **syslog**' bei anderen logfile-Adaptern anzupassen. Beim Stop 'kill $(ps ax | grep "tail" | grep **"/var/log/syslog.log"** | cut -c 1-5)' ist der Pfad der Logdatei entsprechend zu ändern.

- Es tut mir leid. Eigentlich nur was für Bastler. Mit diesem Search-String ist aber die maximale Flexibilität gegeben. Man kann jedoch jede beliebige Datei untersuchen lassen und Aktionen anstoßen.

## 4.10. Überwachung von Directories durch dlogger (Neu ab v1.1.4)

1. Mit dem Programm irlogger können Verzeichnis überwacht werden. Alle Änderungen (Open/Close/Neuanlage/Löschen) im angegebenen Directory werden in eine Protokolldatei geschrieben. Diese Protokolldatei kann dann wiederum durch einen Logfile-Adapter überwacht werden und somit auch Aktionen ausgeführt werden. Gestoppt und gestartet wird irlogger z.B. im boot/shutdown ähnlich irlogger.

   Start : $BIN_ROOT/dlogger /tmp /var/log/dlogger_tmp.log.
   Hier wird das Verzeichnis /tmp überwacht und die Änderungen werden in /var/log/dlogger_tmp.log geschrieben.
   Beenden : /usr/bin/killall -HUP dlogger
   for pid in `ps | grep "dlogger" | grep -v grep | cut -c 1-5`; do kill $pid; done

```
#########################################################################
###
###  Allgemein
#########################################################################
###
OPT_AUTOMATION='yes'
AUTOMATION_START_AT_BOOT_TIME='yes'
AUTOMATION_LOG='yes’
AUTOMATION_DEBUG='no'
AUTOMATION_LOGFILE='/var/log/automation.log'
AUTOMATION_WORK='/var/run/automation'
AUTOMATION_WORK_SAVE='/data/automation'
##### AUTOMATION_IRTRANS_LOGFILE='/var/log/irlogger.log'
##### AUTOMATION_IRTRANS_PORT='21001'
AUTOMATION_REFRESH='15'
AUTOMATION_SLEEPH='1'
AUTOMATION_SLEEPV='0'
AUTOMATION_AUTO_HOSTS='no'
AUTOMATION_HOSTS_N='7'
```

```
#############################################################################
###
###  router  (FLI4L)
#############################################################################
###
AUTOMATION_HOST_1='router'
AUTOMATION_HOST_1_DELAY='86400'
AUTOMATION_HOST_1_BOOTTIME='120'
AUTOMATION_HOST_1_CHILD_N='7'
AUTOMATION_HOST_1_CHILD_1='pc001'
AUTOMATION_HOST_1_CHILD_2='pc002'
AUTOMATION_HOST_1_CHILD_3='pc004'
AUTOMATION_HOST_1_CHILD_4='pc004w'
AUTOMATION_HOST_1_CHILD_5='sc200'
AUTOMATION_HOST_1_CHILD_6='t1001'
AUTOMATION_HOST_1_CHILD_7='irt01'
AUTOMATION_HOST_1_PORT_N='1'
AUTOMATION_HOST_1_PORT_1='22'
AUTOMATION_HOST_1_COMMAND_N='3'

AUTOMATION_HOST_1_COMMAND_1='start'
AUTOMATION_HOST_1_COMMAND_1_N='1'
AUTOMATION_HOST_1_COMMAND_1_1='ether-wake 00:B0:D0:A9:E1:91'

AUTOMATION_HOST_1_COMMAND_2='stop'
AUTOMATION_HOST_1_COMMAND_2_N='1'
AUTOMATION_HOST_1_COMMAND_2_1='shutdown -h now'

AUTOMATION_HOST_1_COMMAND_1='boot'
AUTOMATION_HOST_1_COMMAND_1_CRON='boot'
AUTOMATION_HOST_1_COMMAND_1_N='1'
AUTOMATION_HOST_1_COMMAND_1_1='$BIN_ROOT/irlogger 21001
/var/log/irlogger.log'
AUTOMATION_HOST_1_COMMAND_1_2='$BIN_ROOT/dlogger /tmp
/var/log/dlogger_tmp.log'

AUTOMATION_HOST_1_COMMAND_2='shutdown'
AUTOMATION_HOST_1_COMMAND_2_CRON='shutdown'
AUTOMATION_HOST_1_COMMAND_2_N='4'
AUTOMATION_HOST_1_COMMAND_2_1='/usr/bin/killall -HUP irlogger'
AUTOMATION_HOST_1_COMMAND_2_2='for pid in `ps | grep "irlogger" |  grep -v
grep | cut -c 1-5`; do kill $pid; done'
AUTOMATION_HOST_1_COMMAND_2_3='/usr/bin/killall -HUP dlogger'
AUTOMATION_HOST_1_COMMAND_2_4='for pid in `ps | grep "dlogger" |  grep -v
grep | cut -c 1-5`; do kill $pid; done'

AUTOMATION_HOST_1_COMMAND_3='ping'
AUTOMATION_HOST_1_COMMAND_3_CRON='automation_start'
AUTOMATION_HOST_1_COMMAND_3_N='1'
AUTOMATION_HOST_1_COMMAND_3_1='fping www.t-online.de >/dev/null 2>&1'
```

```
#############################################################################
###
###  pc001 (Windows 2000)
#############################################################################
###
AUTOMATION_HOST_2='pc001'
AUTOMATION_HOST_2_BOOTTIME='180'
AUTOMATION_HOST_3_HTTP='http://pc001'
AUTOMATION_HOST_2_COMMAND_N='3'

AUTOMATION_HOST_2_COMMAND_1='start'
AUTOMATION_HOST_2_COMMAND_1_N='1'
AUTOMATION_HOST_2_COMMAND_1_1='$BIN/command.sh t1001
Schalter_01_On'

AUTOMATION_HOST_2_COMMAND_2='stop'
AUTOMATION_HOST_2_COMMAND_2_N='3'
AUTOMATION_HOST_2_COMMAND_2_1='ssh -i $SSH/router_rsa -l administrator
pc001  D:/Start_Stop/psshutdown.exe -k -t 5'
AUTOMATION_HOST_2_COMMAND_2_2='$BIN/wait.sh pc001 off'
AUTOMATION_HOST_2_COMMAND_2_3='$BIN/command.sh t1001
Schalter_01_Off'

AUTOMATION_HOST_2_COMMAND_3='backup'
AUTOMATION_HOST_2_COMMAND_3_CRON='00 03 * * *;00 12 * * *'
AUTOMATION_HOST_2_COMMAND_3_N='6'
AUTOMATION_HOST_2_COMMAND_3_1='$BIN/command.sh pc001 start'
AUTOMATION_HOST_2_COMMAND_3_2='$BIN/wait.sh pc001 on'
AUTOMATION_HOST_2_COMMAND_3_3='$BIN/status.sh pc001 maint_on'
AUTOMATION_HOST_2_COMMAND_3_4='ssh -i $SSH/router_rsa -l administrator
pc001 D:/Rsync-Backup/PC001/PC001_Doku.bat'
AUTOMATION_HOST_2_COMMAND_3_5='$BIN/status.sh pc001 maint_off'
AUTOMATION_HOST_2_COMMAND_3_6='$BIN/group.sh pc001 stop'
#############################################################################
#######
### pc002 (Linux)
#############################################################################
#######
AUTOMATION_HOST_3='pc002'
AUTOMATION_HOST_3_BOOTTIME='180'
AUTOMATION_HOST_3_COMMAND_N='2'

AUTOMATION_HOST_3_COMMAND_1='start'
AUTOMATION_HOST_3_COMMAND_1_N='5'
AUTOMATION_HOST_3_COMMAND_1_1='ether-wake 00:0C:76:57:A6:C9'
AUTOMATION_HOST_3_COMMAND_1_2='$BIN/wait.sh pc002 1'
AUTOMATION_HOST_3_COMMAND_1_3='$BIN/wait.sh pc001 1'
AUTOMATION_HOST_3_COMMAND_1_4='ssh -i $SSH/router_rsa -l administrator
pc001 D:/Start_Stop/PC002_Mount.bat'
AUTOMATION_HOST_3_COMMAND_1_5='ssh -i $SSH/router_rsa -l administrator
mount /dev/sdb1 /Test'
```

```
AUTOMATION_HOST_3_COMMAND_2='stop'
AUTOMATION_HOST_3_COMMAND_2_N='3'
AUTOMATION_HOST_3_COMMAND_2_1='ssh -i $SSH/router_rsa -l administrator
pc001 D:/Start_Stop/PC002_UMount.bat'
AUTOMATION_HOST_3_COMMAND_2_2='sleep 5'
AUTOMATION_HOST_3_COMMAND_2_3='ssh -i $SSH/router_rsa -l root pc002
shutdown -h now'
##########################################################################
#######
### pc004 (Laptop Ethernet)
##########################################################################
#######
AUTOMATION_HOST_4='pc004'
AUTOMATION_HOST_4_DELAY='10'
AUTOMATION_HOST_4_BOOTTIME='180'
AUTOMATION_HOST_4_GROUP_N='1'
AUTOMATION_HOST_4_GROUP_1='pc001'
##########################################################################
#######
### pc004w (Laptop WLAN)
##########################################################################
#######
AUTOMATION_HOST_5='pc004w'
AUTOMATION_HOST_5_DELAY='10'
AUTOMATION_HOST_5_BOOTTIME='180'
AUTOMATION_HOST_5_GROUP_N='1'
AUTOMATION_HOST_5_GROUP_1='pc001'
##########################################################################
#######
### sc200 (Showcenter)
##########################################################################
#######
AUTOMATION_HOST_6='sc200'
AUTOMATION_HOST_6_DELAY='40'
AUTOMATION_HOST_6_BOOTTIME='60'
AUTOMATION_HOST_6_GROUP_N='2'
AUTOMATION_HOST_6_GROUP_1='pc001'
AUTOMATION_HOST_6_GROUP_2='pc002'
##########################################################################
#######
### t1001 (Schalter)
##########################################################################
#######
AUTOMATION_HOST_7='t1001'
AUTOMATION_HOST_7_COMMAND_N='2'

AUTOMATION_HOST_7_COMMAND_1='Schalter_01_On'
AUTOMATION_HOST_7_COMMAND_1_N='1'
AUTOMATION_HOST_7_COMMAND_1_1=/usr/local/bin/EZconsole t1001 -p 2 255’
```

AUTOMATION_HOST_7_COMMAND_2='Schalter_01_Off'
AUTOMATION_HOST_7_COMMAND_2_N='1'
AUTOMATION_HOST_7_COMMAND_2_1='/usr/local/bin/EZconsole t1001 -p 2 0’
##########################################################################
#######
### irt01 (IRTrans)
##########################################################################
#######
AUTOMATION_HOST_7='t1001'
AUTOMATION_HOST_7_COMMAND_N='4'

AUTOMATION_HOST_7_COMMAND_1='kathrein_power'
AUTOMATION_HOST_7_COMMAND_1_N='1'
AUTOMATION_HOST_7_COMMAND_1_1='/usr/local/bin/irclient irt01 kathrein
power'

AUTOMATION_HOST_7_COMMAND_2='receiver_poweron'
AUTOMATION_HOST_7_COMMAND_2_N='1'
AUTOMATION_HOST_7_COMMAND_2_1='/usr/local/bin/irclient irt01 receiver
poweron'

AUTOMATION_HOST_7_COMMAND_3='receiver_dbs'
AUTOMATION_HOST_7_COMMAND_3_N='1'
AUTOMATION_HOST_7_COMMAND_3_1='/usr/local/bin/irclient irt01 receiver dbs'

AUTOMATION_HOST_7_COMMAND_4=**'radio'**
AUTOMATION_HOST_7_COMMAND_4_N='3'
AUTOMATION_HOST_7_COMMAND_4_1='$BIN/command.sh irt01 kathrein_power'
AUTOMATION_HOST_7_COMMAND_4_2='$BIN/command.sh irt01
receiver_poweron'
AUTOMATION_HOST_7_COMMAND_4_3='$BIN/command.sh irt01 receiver_dbs'
##########################################################################
#######
### AV-Receiver
##########################################################################
#######
AUTOMATION_HOST_8='3805'
AUTOMATION_HOST_8_TYPE='0'
AUTOMATION_HOST_8_COMMAND_N='4'

AUTOMATION_HOST_8_COMMAND_1='start'
AUTOMATION_HOST_8_COMMAND_1_N='1'
AUTOMATION_HOST_8_COMMAND_1_1='/usr/local/bin/irclient irt01 3805 poweron'

AUTOMATION_HOST_8_COMMAND_2='stop'
AUTOMATION_HOST_8_COMMAND_2_N='1'
AUTOMATION_HOST_8_COMMAND_2_1='/usr/local/bin/irclient irt01 3805 poweroff'

AUTOMATION_HOST_8_COMMAND_3='tv'
AUTOMATION_HOST_8_COMMAND_3_N='1'
AUTOMATION_HOST_8_COMMAND_3_1='/usr/local/bin/irclient irt01 3805 tv'

```
AUTOMATION_HOST_8_COMMAND_4='default'
AUTOMATION_HOST_8_COMMAND_4_N='1'
AUTOMATION_HOST_8_COMMAND_4_1='/usr/local/bin/irclient  irt01 3805
$VALUE'
#########################################################################
#######
### Schaltsteckdosen Nr. 1
#########################################################################
#######
AUTOMATION_HOST_9='schalter1'
AUTOMATION_HOST_9_DELAY='20'
AUTOMATION_HOST_9_TYPE='2'
AUTOMATION_HOST_9_TYPE2_SERVER='t1001'
AUTOMATION_HOST_9_TYPE2_PARM='5'
AUTOMATION_HOST_9_GROUP_N='2'
AUTOMATION_HOST_9_GROUP_1='schalter2'
AUTOMATION_HOST_9_GROUP_2='schalter3'
AUTOMATION_HOST_9_CHILD_N='2'
AUTOMATION_HOST_9_CHILD_1='schalter2'
AUTOMATION_HOST_9_CHILD_2='schalter3'
AUTOMATION_HOST_9_COMMAND_N='2'

AUTOMATION_HOST_9_COMMAND_1='start'
AUTOMATION_HOST_9_COMMAND_1_N='1'
AUTOMATION_HOST_9_COMMAND_1_1='/usr/local/bin/EZconsole t1001 -p 5 255'

AUTOMATION_HOST_9_COMMAND_2='stop'
AUTOMATION_HOST_9_COMMAND_2_N='1'
AUTOMATION_HOST_9_COMMAND_2_1='/usr/local/bin/EZconsole t1001 -p 5 0'
#########################################################################
#######
### syslog Logfile-Adapter
#########################################################################
#######
AUTOMATION_HOST_10='syslog'
AUTOMATION_HOST_10_TYPE='3'
AUTOMATION_HOST_10_TYPE3_FILE='/var/log/syslog.log'
AUTOMATION_HOST_10_COMMAND_N='4'

AUTOMATION_HOST_10_COMMAND_1='start'
AUTOMATION_HOST_10_COMMAND_1_N='1'
AUTOMATION_HOST_10_COMMAND_1_1='$BIN/logfile.sh syslog'

AUTOMATION_HOST_10_COMMAND_2='stop'
AUTOMATION_HOST_10_COMMAND_2_N='1'
AUTOMATION_HOST_10_COMMAND_2_1='kill $(ps ax | grep "tail" | grep
"/var/log/syslog.log" | cut -c 1-5)'

AUTOMATION_HOST_10_COMMAND_3='ip'
```

```
AUTOMATION_HOST_10_COMMAND_3_SEARCH='grep "139.7.30.125:53" | cut -f 8
-d " " | cut -f 1 -d ":"'
AUTOMATION_HOST_10_COMMAND_3_COUNT='* *'
AUTOMATION_HOST_10_COMMAND_3_N='1'
AUTOMATION_HOST_10_COMMAND_3_1='echo "$VALUE" > /tmp/ip.tmp'

AUTOMATION_HOST_10_COMMAND_4='test'
AUTOMATION_HOST_10_COMMAND_4_SEARCH='grep "cpmvrmlog_check" | cut -
c 1-40'
AUTOMATION_HOST_10_COMMAND_4_COUNT='3 *'
AUTOMATION_HOST_10_COMMAND_4_N='1'
AUTOMATION_HOST_10_COMMAND_4_1='echo "$VALUE" >> /tmp/test.tmp'
##########################################################################
#######
# irtrans Logfile-Adapter
##########################################################################
#######
AUTOMATION_HOST_24='irtrans'
AUTOMATION_HOST_24_TYPE='3'
AUTOMATION_HOST_24_TYPE3_FILE='/var/log/irlogger.log'
AUTOMATION_HOST_24_COMMAND_N='3'

AUTOMATION_HOST_24_COMMAND_1='start'
AUTOMATION_HOST_24_COMMAND_1_N='1'
AUTOMATION_HOST_24_COMMAND_1_1='$BIN/logfile.sh irtrans'

AUTOMATION_HOST_24_COMMAND_2='stop'
AUTOMATION_HOST_24_COMMAND_2_N='1'
AUTOMATION_HOST_24_COMMAND_2_1='kill $(ps ax | grep "tail" | grep
"/var/log/irlogger.log" | cut -c 1-5)'

AUTOMATION_HOST_24_COMMAND_3='command'
AUTOMATION_HOST_24_COMMAND_3_SEARCH='grep "21000" | cut -f 4 -d ":" |
cut -f 2- -d " " | grep " "'
AUTOMATION_HOST_24_COMMAND_3_COUNT='* *'
AUTOMATION_HOST_24_COMMAND_3_N='1'
AUTOMATION_HOST_24_COMMAND_3_1='$BIN/command.sh $VALUE'
############################### Ende
##########################################
```

---

## 5. Einiges rund um ssh

Beipiele für eine ssh-Umgebung. Ich setze im Router-Umfeld rsa-keys ein. Das ganze ssh-Thema ist nicht ganz trivial, doch die Arbeit lohnt.

### 5.1. Router

- Create neue Hostkeys für den Router wie in der Dokumentation von OPT_SSHD beschrieben (SSHD_CREATEHOSTKEYS). Übernahme dieser generierten Schlüssel in das Installations-Verzeichnis. ..\etc\ssh. Ich arbeite um die Passworteingabe zu umgehen ohne Passphrase. Um nochmals die Beschreibungen aus OPT_SSHD zu wiederholen. Die Schlüssel sind sorgfältig aufzuheben.

- Der Server braucht die public-Keys der Client behält die private-Keys. Auf dem Server werden alle für diesen User gültigen public-keys in einer Datei authorized_keys zusammengefasst. Dies geschieht unter Windows mit einem Editor unter Unix mit dem cat-Befehl.

- Um also dem SSH-Server die Möglichkeit zu geben Kommandos an andere Systeme mittels ssh abzugeben, müssen noch die public-Keys der Ziel-Systeme dem Router zur Verfügung gestellt werden. Dies geschieht indem man diese public-Keys ebenfalls in das Verzeichnis ..\etc\ssh übernimmt und im sshd.txt als (SSHD_PUBLIC_KEYSFILES_N und SSHD_PUBLIC_KEYFILE_x) übernimmt.

- Ebenso habe ich die known_hosts nach ../etc/ssh übernommen wie unter (OPT_SSH_CLIENT) beschrieben.

5.2. Windows

- copSSH als SSH-Server für Windows-Systeme von sourceforge.net.

- Nach der Installation des copSSH-Servers auf dem Windows-System wird ein User eingerichtet. Hier schlägt copSSH die verfügbaren Windows-User vor. Dieser User wird dann für die Ausführung der ssh-Kommandos, die vom Router kommen gebraucht. Dieser User braucht also die entsprechenden Berechtigungen für die Kommandos.

- Danach existiert im copSSH-Verzeichnis folgender Bereich \copSSH\home\User\.ssh. In dieses Verzeichnis werden dann alle öffentlichen Schlüssel der Clients hinterlegt, die sich an den SSH-Server mit diesem User wenden. Diese keys aus dem Verzeichnis \copSSH\home\User\.ssh werden dann mit einem Editor in einem File mit dem Namen authorized_keys zusammengefasst.

- Ebenso wird der von copSSH generierte rsa-public-key aus \copSSH\etc\ssh_host_rsa_key.pub in das Router-Verzeichnis \etc\.ssh kopiert und wie unter 5.1. beschrieben übernommen.

5.3. Linux

- Generieren von keys unter Linux mit ssh-keygen –t rsa –b 2048 ~/rsa_key. Die Abfragen immer mit Return. Danach exiistieren zwei Dateien rsa_key und rsa_key.pub. Auch dieser generierte public-key muss auf dem Router nach ..\etc\.ssh kopiert und wie unter 5.1 beschrieben übernommen werden.

5.4. Windows und netcat

- Eine Alternative im Windows-Bereich ohne ssh ist der Einsatz von netcat. Dies entspricht nicht den Sicherheits-Anforderungen wie die ssh-Lösung ist aber erheblich einfacher einzurichten.

- Download netcat for windows von der Seite www.vulnwatch.org/netcat/ und Installation. Man braucht nur das nc.exe. Mit nc.exe –h bekommt man eine kurze Beschreibung.

- Einsatz netcat auf fli4l aus dem Tools-Paket.

- Eintragen der gewünschten Kommandos in die Windows-Registry unter HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run als Charakter-String z.B.

    1. String -> **automation_shutdown** Wert -> **C:\Programme\nc.exe –d –l –p 4711 –t –e "C:\Programme\psshutdown.exe –k –t 10"** zum runterfahren mit psshutdown wenn port 4711 angesprochen wird.

    2. String -> **automation_backup** Wert -> **C:\Programme\nc.exe –d –L –p 4712 –t –e "C:\WINDOWS\system32\cmd.exe /c backup.bat"** zum Starten eines bat-files wenn port 4712 angesprochen wird. (Achtung großes –L damit mehrfacher Aufruf möglich).

- Nun kann man mittels netcat pc002 4711 in der Automation den pc002 herunterfahren.

---

## 6. Mini-httpd Oberfläche automation.cgi

Die Weboberfläche im httpd ist selbstsprechend. Hier kann OPT-AUTOMATION ein- und ausgeschaltet und die Logdatei eingesehen werden. Ebenso werden die Parameter angezeigt und der Status der einzelnen Systeme farbig sichtbar. Hier kann man alle Kommandos eines Systems über die Oberfläche ausführen. Im Wartungsmodus werden an dieses System keine Automations-Kommandos gesendet.

---

## 7. Beschreibung des automation_batch.cgi

Ich habe ein weiteres cgi-Skript gebaut, dass über den Browser als Kommando genutzt werden kann. z.B. http://router/automation_batch.cgi?action=pc001,stop,back. Dieses Modul habe ich angepasst damit man vernünftige HTML-Seiten bestückt mit Kommandos aufbauen kann (ohne Logon). Die Parameter sind das Zielgerät und das Kommando aus der Conf-Datei, und die Ergänzung back = Rücksprung, echo = Ausgabe von Informationen und quiet = keine Ausgaben. Ich werde als nächstes einige HTML-Seiten bauen, die ich dann über WLAN und einer Sony PSP evtl. Nokia 770 an den Router gebe. Für den back-Parameter muß Java-Skript im Browser zugelassen werden.

Für das Auslösen eines Automations-Skripts von einem anderen PC aus hat sich bei mir eher ein anderes Verfahren über plink etabliert. z.B. C:\Programme\PuTTY\plink.exe -load router "exec /usr/local/bin/automation/command.sh pc001 stop"

(Neu ab v1.1.4) Einbau einer Status-Abfrage.

```
status=`$BIN/status.sh $host status`
case $status in
    0)    status_text="off" ;;
    1)    status_text="on" ;;
    2)    status_text="starting" ;;
    3)    status_text="stopping" ;;
    4)    status_text="group_wait" ;;
    5)    status_text="parent_wait" ;;
    6)    status_text="parent_wait" ;;
    9)    status_text="init" ;;
    10)   status_text="manual_off" ;;
    11)   status_text="manual_on" ;;
    12)   status_text="manual_starting" ;;
    13)   status_text="manual_stopping" ;;
    14)   status_text="manual_group_wait" ;;
esac
```

---

## 8. Lizenz



Roland Spitzer roland@familie-spitzer.de