

nfRunner Version 2.10

© stroetmann

Download only on www.NaviFriends.com

nfRunner ist ein Scriptinterpreter, der ohne zusätzliche DLLs auskommt. Es kann also nach einem Hardreset auch auf einem PNA ausgeführt werden, um das System wieder herzustellen. Die Datei mit der Endung .nfRun ist eine Textdatei, die die auszuführenden Anweisungen enthält (siehe unter Kommandos).

Die nfRunner.EXE Datei kann beliebig umbenannt werden. Die nfRun Datei muss den gleichen Namen haben und im selben Verzeichnis liegen.

Wenn man die nfRunner.EXE als Autostart benutzen will, müssten die Dateien Autostart.exe und Autostart.nfRun heißen.

Man kann sich so für verschiedene Aufgaben ein Datei-Pärchen anlegen.

z. B. HARDRESET.EXE und HARDRESET.NFRUN, wobei in der HARDRESET.NFRUN nur die Zeile "HARDRESET" steht. Nach einer Sicherheitsabfrage erfolgt dann ein Hard-Reset. Das schont den EIN/AUS-Schalter.

Es geht auch eine komplette Neuinstallation eines PNA-Systems inklusive des POI:Observers, sogar mit mehreren Soft-Resets (siehe Beispiel für die Falk N-Serie).

Kommandos:

Die Parameter müssen in Anführungszeichen stehen.

QPFAD und ZPFAD müssen den kompletten Pfad der Datei enthalten, also auch den führenden Backslash.

Falls der Pfad mit "." beginnt, werden diese beiden Zeichen durch den aktuellen Pfad der nfRun-Datei ersetzt.

Die Befehle COPY,DCOPY,MOVE und DELETE können die Wildcards "?" für ein einzelnes Zeichen und "*" für alle Zeichen im Dateinamen enthalten.

Bei COPY,DCOPY und MOVE muss die Zieldatei dann mit "*" bezeichnet werden.

Z.B.: COPY "\Starage Card\DLLS*.*" "\windows*"

DWORD ist eine vorzeichenlose Zahl, und kann auch in der HEX-Notation von C eingegeben werden. z.B. 0xABCD..

KEY ist der komplette Pfad des Schlüssels in der Registry.

Der Schlüssel muss mit einem der folgenden Kürzeln beginnen:

"HKCR" für HKEY_CURRENT_ROOT

"HKCU" für HKEY_CURRENT_USER

"HKLM" für HKEY_LOCAL_MACHINE

Kommentarzeilen beginnen mit der Raute "#".

Durch die neuen Funktionen und die Variablen sind das Dollarzeichen \$ und das At-Zeichen @ im normalen Text darstellbar, wenn man jeweils zwei dieser Zeichen eingibt. Sie werden dann vom Parser durch ein einzelnes ersetzt.

nfRunner – the Scriptinterpreter

for mobile Windows

Variablen

Variablen-Namen werden im Text mit dem Dollarzeichen \$ eingerahmt. Bei der Ausführung wird dies dann durch den aktuellen Text-Inhalt der Variablen ersetzt. Die Namen können beliebige Zeichen enthalten, und es wird zwischen Groß- und Kleinschreibung unterschieden. Um das Dollarzeichen in einem Text darzustellen, müssen zwei Dollarzeichen hintereinander eingegeben werden. Sie werden dann durch ein Dollarzeichen ersetzt.

Mit SETVAR kann einer Variablen ein Text zugewiesen werden. Wenn der zugewiesene Text mit einer öffnenden runden Klammer beginnt, wird der Text als arithmetischer Ausdruck betrachtet und zunächst berechnet. Das Ergebnis wird dann der Variablen als Text zugewiesen.

Folgende Operatoren stehen zur Verfügung:

Rechen-Operatoren

Zeichen	Bezeichnung	Priorität
*	Multiplikation	3
/	Division	3
%	Modulo Division	3
+	Addition	2
-	Subtraktion	2

Logik-Operatoren

Zeichen	Bezeichnung	Priorität
!	Nicht	4
&	Und	1
	Oder	1
=	Gleich	1
!=	Ungleich	1
>	Größer	1
>=	Größer gleich	1
<	Kleiner	1
<=	Kleiner gleich	1

Wenn die logische Bedingung zutrifft, ist das Ergebnis **-1** ansonsten **0**.

Die Priorität bestimmt die Reihenfolge, mit der die Operatoren ausgeführt werden. Durch Klammern kann man diese Priorität verändern. Im Zweifelsfall sollte man immer Klammern setzen.

System-Variablen

Es gibt folgende vordefinierte System-Variablen:

\$VERSION\$	Enthält die Versionsnummer*100
\$EXE\$	Gibt den kompletten Pfad zur EXE-Datei zurück, die dieses Script interpretiert.
\$LASTERROR\$	Enthält den letzten Fehler bei einer SENDxx Anweisung
\$SENDRESULT\$	Enthält den Rückgabewert der letzten SENDxx Anweisung
\$RUNRESULT\$	Enthält den letzten Rückgabewert von RUNANDWAIT oder NFRUNANDWAIT.
\$SCRIPTRESULT\$	Das ist der Wert, den das Script zum Schluss zurück gibt
\$0\$	Name des Scriptes
\$1\$ - \$9\$	Bis zu 9 durch Komma getrennte Übergabeparameter an das Script

nfRunner – the Scriptinterpreter

for mobile Windows

Funktionen

Funktionen können überall da verwendet werden, wo auch Variablen erlaubt sind. Sie beginnen immer mit dem At-Zeichen @ und müssen auch ohne Parameter immer ein Klammer-Paar haben. Um das At-Zeichen @ in einem Text darzustellen, müssen zwei @ hintereinander eingegeben werden. Sie werden dann durch ein @ ersetzt. Bei den Funktionsnamen wird nicht zwischen Groß- und Kleinschreibung unterschieden.

Funktion	Beschreibung
@TIME()	Gibt die Uhrzeit zurück. Wird als Parameter eine 1 übergeben, werden alle Trennzeichen entfernt.
@DATE()	Gibt das Datum zurück. Wird als Parameter eine 1 übergeben, werden alle Trennzeichen entfernt und das Format ist JJJJMMTT.
@ScreenX()	Gibt die vertikale Auflösung des Bildschirms in Pixeln zurück.
@ScreenY()	Gibt die horizontale Auflösung des Bildschirms in Pixeln zurück.
@COLOR(x,y[,1])	Gibt die Farbe an der Position x,y zurück. Wird als zusätzlicher Parameter eine 1 übergeben, wird der Wert in Hex-Notation zurückgegeben.
@COLOR=(BGR1,BGR2,Toleranz)	Vergleicht zwei BGR-Werte auf "Ähnlichkeit". Wenn die Blau-, Grün oder Rotwerte von <i>BGR1</i> jeweils nicht mehr als <i>TOLERANZ</i> nach oben oder unten von den Werten von <i>BGR2</i> abweichen, wird -1 zurückgegeben, sonst eine 0.
@SETPOINT(WERT,NK[,1])	Gibt einen Text zurück, in dem die Ganzzahl <i>WERT</i> als Zahl mit <i>NK</i> Nachkommastellen dargestellt wird. Wird als zusätzlicher Parameter eine 1 übergeben, wird ein Komma anstelle des Punktes gesetzt.
@STR=("S1","S2"[,Z])	Vergleicht die beiden Strings <i>S1</i> und <i>S2</i> miteinander. Die Strings müssen in Anführungszeichen stehen. Ist der optionale Parameter <i>Z</i> angegeben, werden nur die ersten <i>Z</i> Zeichen verglichen. Wenn die Strings gleich sind wird -1 zurückgegeben, sonst eine 0.
@USTR=("S1","S2"[,Z])	Wie @STR=, aber es wird nicht zwischen Groß- und Kleinschreibung unterschieden.
@STRCOPY("S1",START,LEN)	Gibt den Teilstring von <i>S1</i> ab der Position <i>START</i> mit der Länge <i>LEN</i> zurück.
@STRLEN("S1")	Gibt die Länge des Strings <i>S1</i> zurück.
@STRUPPER("S1")	Gibt den String <i>S1</i> in Großbuchstaben zurück.
@STRLOWER("S1")	Gibt den String <i>S1</i> in Kleinbuchstaben zurück.
@STRREV("S1")	Gibt den String <i>S1</i> in umgekehrter Reihenfolge zurück.
@STRPOS("S1","S2" [,START])	Sucht nach einem Vorkommen des Strings <i>S2</i> in <i>S1</i> und gibt die Position zurück. Wenn <i>START</i> angegeben ist wird erst ab dieser Position verglichen, der Rückgabewert ist dann relativ zu <i>START</i> . Es wird eine 0 zurückgegeben, wenn <i>S2</i> nicht in <i>S1</i> vorkommt.
@TRIM("STRING")	Entfernt Leer- und Steuerzeichen am Anfang und am Ende von <i>STRING</i>

nfRunner – the Scriptinterpreter

for mobile Windows

Funktion	Beschreibung
@HEX(WERT [,STELLEN])	Gibt den Wert in HEX-Notation ohne führendes "0x" zurück. Wenn STELLEN angegeben sind, wird die Zahl bei Bedarf mit führenden Nullen aufgefüllt.
@ISDIGIT("STRING" [,POS])	Gibt -1 zurück, wenn das Zeichen an Position POS im String eine Ziffer ist. Sonst wird eine 0 zurückgegeben. Ist POS nicht angegeben, wird das erste Zeichen überprüft.
@LOWORD(WERT)	Gibt die untersten 16-Bit von Wert zurück.
@HIWORD(WERT)	Gibt die obersten 16-Bit von Wert zurück.
@BIT(WERT,NR)	Wenn das Bit an der Position NR in WERT gleich 1 ist wird -1 zurückgegeben, sonst eine 0. Das unterste Bit hat die Position 0.
@STRKEY(KEY)	Gibt den Registry-Wert des String Schlüssels KEY zurück.
@DWKEY(KEY)	Gibt den Registry-Wert des DWORD Schlüssels KEY zurück.
@KEYTYPE(KEY)	Gibt den Typ des Schlüssels KEY zurück. 0 = Schlüssel existiert nicht 1 = String 2 = DWORD 3 = BINARY 99 = Unbekannter Typ
@READINI(PFAD,AB,KEY,DEFAULT)	Liest aus der INI-Datei PFAD im Abschnitt [AB] den Wert des Schlüssels KEY aus. Falls die Datei, der Abschnitt oder der Schlüssel nicht existiert, wird DEFAULT zurückgegeben. Wenn bei AB nicht angegeben wird, wird in der ganzen Datei nach dem Schlüssel KEY gesucht.
@GETROTATEPOS()	Gibt die augenblickliche Rotate-Position des Bildschirms zurück. Siehe auch SCREENROTATE
@GETVOLUME()	Gibt die aktuelle Systemlautstärke zurück.
@BATTERY()	Gibt den Ladezustand der Batterie in Prozent zurück. Wenn die Batterie gerade geladen wird, ist der Rückgabewert 1000. Im Fehlerfall wird 0 zurückgegeben.
@OPENFILE("NAME")	Öffnet eine Text-Datei NAME zum lesen. Die Funktion gibt ein Handle auf diese Datei zurück. Im Fehlerfall wird Null zurückgegeben. Die Datei muss dann mit CLOSEFILE wieder geschlossen werden.
@READLN(FHANDLE)	Gibt die nächste Zeile aus der Datei mit dem Handle FHANDLE zurück.
@EOF(FHANDLE)	Prüft auf End Of File . Gibt -1 zurück, wenn das Ende der Datei erreicht ist, sonst Null.
@GETFILEVERSION("Name")	Gibt den File-Versions String der Datei NAME zurück, falls so eine Resource vorhanden ist. Gibt es die VersionsInfo Resource oder die Datei nicht, wird ein leerer String zurückgegeben.
@OEMINFO()	Gibt die Geräte-Bezeichnung des Herstellers zurück.
@GETSTATE(NR)	Gibt den Wert der Systemvariablen NR zurück. Siehe auch unter <i>Remote Schnittstelle</i>

nfRunner – the Scriptinterpreter

for mobile Windows

Remote Schnittstelle

In jedem Script sind 16 Statusvariablen vorhanden. Beim Start sind diese Variablen alle auf 0 gesetzt.

Im Script kann man den Wert dieser Statusvariablen mit **@GetState(NR)** abfragen, und mit **SETSTATE "NR" "WERT"** kann der Wert der Statusvariablen werden. Wobei **NR** eine Zahl von 0 bis 15 ist. **WERT** kann jede beliebige Ganzzahl sein.

Diese Statusvariablen können von anderen Programmen/Scripten verändert werden, in dem man eine Nachricht mit Nummer 1111 an das Script sendet.

Die Nachricht sieht folgendermaßen aus: **SENDMESSAGE "NAME" "1111,NR,WERT"**

NAME ist der Fensternamen des fremden Scriptes, **NR** ist die Nummer der Statusvariablen des fremden Scriptes, **WERT** ist der neue Wert dieser Variablen. Der alte Wert der Statusvariablen wird von SENDMESSAGE zurückgegeben und steht daher in **\$SENDRESULT\$**.

Wenn die Nummer zwischen 0 und 15 ist, wird der Wert einfach gesetzt.

Wenn bei der Nummer das Bit 8 gesetzt ist, also Nummern zwischen 0x100 und 0x10F ist, wird die entsprechende Statusvariable um den übergebenen Wert erhöht.

SENDMESSAGE "NAME" "1111,0x102,10" würde die Statusvariable 2 um 10 erhöhen.

Hier einmal ein kurzes Beispiel:

```
# Fester Titel für das fremde Script setzen
SETTITEL "TESTSCRIPT"

#INIT Script
SETSTATE "1" "1000"

# Hier sind alle Aktionen, die das Script
# vor dem eigentlichen Dauerlauf erledigen muss

WHILE "@GetState(0)=0"

# Hintergrundscript z.B. zum mitloggen von GPS-Daten
# Läuft solange, bis sich die Statusvariable 0 ändert

    SLEEP "@GetState(1)"
ENDWHILE
#
# Exit Script
# Hier sind alle Aktionen,
# die das Script vor dem Beenden noch machen muss

QUIT
```

Dieses Script würde jetzt ewig laufen, aber durch ein anderes Script mit nur einer Zeile können wir das Script jetzt definiert beenden:

```
SENDMESSAGE "TESTSCRIPT" "1111,0,1"
```

Diese eine Zeile verändert die Statusvariable 0 im fremden Script, dadurch wird die WHILE Schleife verlassen.

Bei diesem Beispiel-Script kann man auch die SLEEP Dauer von außen beeinflussen mit

```
SENDMESSAGE "TESTSCRIPT" "1111,1,10000"
```

erhöht man die SLEEP Dauer in der WHILE-Schleife auf 10 Sekunden.

So kann man jetzt jedes Script fernsteuern und auch Werte abfragen. Die Abfrage funktioniert so, dass man bei der Addition einfach den Wert 0 angibt, dann bekommt man den aktuellen Wert der Statusvariablen zurück.

nfRunner – the Scriptinterpreter

for mobile Windows

IF

Parameter: "*AUSDRUCK*"

AUSDRUCK definiert eine logische Bedingung. Es können alle Rechen- und Logik-Operatoren darin vorkommen. Wenn *AUSDRUCK* nicht Null ist wird der *IF* Zweig ausgeführt, bei Null der *ELSE* Zweig.

```
IF "($Farbe1$=0xC0C0C0) & (($Farbe2$!=0xC0C0C0) |  
($Farbe3$=0x0000FF)) "  
    HIDEWINDOW "EXEC"  
ENDIF
```

IFFILEEXIST

Parameter: "*ZPFAD*"

Prüft ob die Datei *ZPFAD* existiert.

Wenn die Datei existiert, werden die Zeilen bis zum nächsten *ELSE* oder *ENDIF* ausgeführt. Der *ELSE* Teil wird dann bis zum nächsten *ENDIF* übersprungen.

Wenn sie nicht existiert werden alle Zeilen bis zum nächsten *ELSE* oder *ENDIF* übersprungen und der evtl. vorhandene *ELSE* Teil wird bis zum *ENDIF* ausgeführt.

```
IFFILEEXIST ".\start.LOG"  
    MOVE ".\start.LOG" ".\startold.log"  
ENDIF
```

IFYES

Parameter: "*TITEL*" "*FRAGE*"

Es wird eine Dialogbox mit einer JA/NEIN Abfrage geöffnet.

TITEL ist Kopfzeile des Dialogs, *FRAGE* ist der Text der Dialogbox.

Wenn der User auf "JA" klickt, wird der *IF*-Zweig ausgeführt. Bei "NEIN" wird der *IF*-Zweig übersprungen und, falls vorhanden, der *ELSE*-Zweig ausgeführt.

```
IFYES "Dumme Frage" "Soft-Reset jetzt ausführen?"  
    RESET  
ENDIF
```

nfRunner – the Scriptinterpreter

for mobile Windows

IFFILEEQUAL

Parameter: "*PFAD1*" "*PFAD2*"

Wenn der Inhalt der Dateien *PFAD1* und *PFAD2* identisch sind, wird der *IF* Zweig ausgeführt, sonst der evtl. vorhandene *ELSE* Teil.

```
IFFILEEQUAL ".\OBS.CFG" ".\DEVICE.CFG"  
# alles klar  
ELSE  
    DCOPY ".OBS.CFG" ".\DEVICE.CFG"  
ENDIF
```

IFFEQ

Parameter: "*PFAD1*" "*PFAD2*"

Kurzform von *IFFILEEQUAL*.

```
IFFEQ ".\OBS.CFG" ".\DEVICE.CFG"  
# alles klar  
ELSE  
    DCOPY ".OBS.CFG" ".\DEVICE.CFG"  
ENDIF
```

IFCOLOR

Parameter: "*X*, *Y*, *COLOR*"

Prüft, ob auf dem Bildschirm das Pixel an der Position *X*, *Y* die Farbe *COLOR* hat.

```
IFCOLOR "10,10,0"  
    MOUSECLICK "10,10"  
ELSE  
    QUIT  
ENDIF
```

nfRunner – the Scriptinterpreter

for mobile Windows

IFWINDOWEXIST

Parameter: "*NAME*" ["*KLASSE*"]

Prüft ob das Fenster *NAME* existiert.

Wenn das Fenster existiert, werden die Zeilen bis zum nächsten *ELSE* oder *ENDIF* ausgeführt. Der *ELSE* Teil wird dann bis zum nächsten *ENDIF* übersprungen.

Wenn es nicht existiert, werden alle Zeilen bis zum nächsten *ELSE* oder *ENDIF* übersprungen und der evtl. vorhandene *ELSE* Teil wird bis zum *ENDIF* ausgeführt.

Optional kann auch der Klassenname des Fensters angegeben werden.

```
IFWINDOWEXIST "POI Observer"  
  HIDEWINDOW "EXEC"  
ENDIF
```

IFWE

Parameter: "*NAME*" ["*KLASSE*"]

Kurzform von *IFWINDOWEXIST*.

```
IFWE "POI Observer"  
  HIDEWINDOW "EXEC"  
ENDIF
```

IFWINDOWVISIBLE

Parameter : "*NAME*" ["*KLASSE*"]

Prüft, ob das Fenster *NAME* sichtbar ist.

Wenn das Fenster sichtbar ist, werden die Zeilen bis zum nächsten *ELSE* oder *ENDIF* ausgeführt. Der *ELSE* Teil wird dann bis zum nächsten *ENDIF* übersprungen.

Wenn es nicht sichtbar ist, werden alle Zeilen bis zum nächsten *ELSE* oder *ENDIF* übersprungen und der evtl. vorhandene *ELSE* Teil wird bis zum *ENDIF* ausgeführt.

Optional kann auch der Klassenname des Fensters angegeben werden.

```
IFWINDOWVISIBLE "POBS"  
  MOUSECLICK "1,1" "POBS"  
ENDIF
```

IFWV

Parameter: "*NAME*" ["*KLASSE*"]

Kurzform von *IFWINDOWVISIBLE*.

```
IFWV "POBS"  
  MOUSECLICK "1,1" "POBS"  
ENDIF
```

nfRunner – the Scriptinterpreter

for mobile Windows

IFSTRKEY

Parameter: "*KEY*" "*WERT*"

Prüft, ob der Registry Schlüssel *KEY* die Zeichenfolge *WERT* hat. Groß- und Kleinschreibung werden ignoriert und der Schlüssel wird auch nur bis zur Länge von *WERT* verglichen.

```
IFSTRKEY "HKLM\Platform\Name" "Falk P-Series"  
  SHOWMSG "P300" "Es ist ein P-Serien Gerät von Falk"  
ENDIF
```

IFDWKEY

Parameter: "*KEY*" "*DWORD*"

Prüft, ob der Registry Schlüssel *KEY* den Zahlenwert *DWORD* hat.

```
IFDWKEY "HKLM\Explorer\ShowSys" "1"  
  SHOWMSG "Explorer" "System Dateien werden angezeigt"  
ENDIF
```

ELSE

Leitet den *ELSE* Teil ein

```
ELSE
```

ELSEIF

Leitet den *ELSE* Teil mit gleichzeitiger *IF*-Abfrage ein. Das erleichtert das Verschachteln von *IF*-Abfragen.

```
IF "$Wert$=1"  
  SETVAR "Text" "Eins"  
ELSEIF "$Wert$=2"  
  SETVAR "Text" "Zwei"  
ELSEIF "$Wert$=3"  
  SETVAR "Text" "Drei"  
ELSE  
  SETVAR "Text" "Ich kann nur bis drei zählen"  
ENDIF
```

ENDIF

Beendet eine *IF* Abfrage

```
ENDIF
```

nfRunner – the Scriptinterpreter

for mobile Windows

WHILE

Parameter: "*AUSDRUCK*"

AUSDRUCK definiert eine logische Bedingung. Es können alle Rechen- und Logik-Operatoren darin vorkommen. Wenn *AUSDRUCK* nicht Null ist, wird der *WHILE* Zweig ausgeführt, bei Null geht die Ausführung nach *ENDWHILE* weiter.

```
WHILE "0!=@EOF($Datei$) "  
    SHOWMSG "INFO" "@READLN($Datei$) "  
ENDWHILE
```

ENDWHILE

Beendet eine *WHILE* Schleife, springt nach *WHILE* zurück und prüft die Bedingung erneut.

```
WHILE "@EOF($Datei$) !=0 "  
    SHOWMSG "INFO" "@READLN($Datei$) "  
ENDWHILE
```

RUN

Parameter: "*QPFAD*" [*"PARA"*]

Das Programm *QPFAD* wird gestartet, und falls *PARA* angegeben ist, wird es als Parameter an das Programm übergeben, dabei werden einfache Anführungszeichen durch doppelte ersetzt

```
RUN "\windows\explorer.exe"
```

RUNANDWAIT

Parameter: "*QPFAD*" [*"PARA"*]

Das Programm *QPFAD* wird gestartet, und falls *PARA* angegeben ist, wird es als Parameter an das Programm übergeben, dabei werden einfache Anführungszeichen durch doppelte ersetzt. Das Script wartet bis das Programm beendet wird.

```
RUNANDWAIT "\windows\explorer.exe" ""
```

RAW

Parameter: "*QPFAD*" [*"PARA"*]

Ist die Kurzform von *RUNANDWAIT*

```
RAW "\windows\explorer.exe" ""
```

nfRunner – the Scriptinterpreter

for mobile Windows

NFRUN

Parameter: "*QPFAD*"

Das nfRunner-Script *QPFAD* wird gestartet. Parameter für das Script werden durch Komma getrennt an den Namen angehängt.

```
NFRUN ".\script.nfRun"  
NFRUN ".\script.nfRun,Parameter1,Parameter2"
```

NFRUNANDWAIT

Parameter : "*QPFAD*"

Das nfRunner-Script *QPFAD* wird gestartet.

Das aktuelle Script wartet, bis das andere beendet ist.

Parameter für das Script werden durch Komma getrennt an den Namen angehängt.

```
NFRUNANDWAIT ".\script.nfRun"  
NFRUNANDWAIT ".\script.nfRun,Parameter1,Parameter2"
```

NFRAW

Parameter : "*QUPFAD*"

Ist die Kurzform von *NFRUNANDWAIT*

```
NFRAW ".\script.nfRun"
```

NFRUNAGAIN

Führt das aktuelle Script wieder vom Anfang an aus.

```
NFRUNAGAIN
```

COPY

Parameter : "*QPFAD*" "*ZPFAD*"

Kopiert Datei *QPFAD* nach *ZPFAD*. Wenn die Datei schon vorhanden ist, wird sie nicht überschrieben.

```
COPY "\Storage Card\DLLs\AYGSHELL.DLL" "\windows\AYGSHELL.DLL"
```

nfRunner – the Scriptinterpreter

for mobile Windows

DCOPY

Parameter : "*QPFAD*" "*ZPFAD*"

Wie *COPY*, vorhandene Dateien werden überschrieben.

```
DCOPY "\Storage Card\DLLs\AYGSHELL.DLL" "\windows\AYGSHELL.DLL"
```

MOVE

Parameter: "*QPFAD*" "*ZPFAD*"

Verschiebt eine Datei von *QPFAD* nach *ZPFAD*. Kann auch zum Umbenennen benutzt werden.

```
MOVE "\Storage Card\DLLs\AYGSHELL.DLL" "\windows\AYGSHELL.DLL"
```

DELETE

Parameter: "*ZPFAD*"

Löscht die Datei *ZPFAD*

```
DELETE "\Storage Card\DLLs\AYGSHELL.DLL"
```

LINK

Parameter : "*ZPFAD*" "*QPFAD*" [*"PARA"*]

Erzeugt eine Verknüpfung an der Position *ZPFAD* die auf die Datei in *QPFAD* verweist. Das optionale *PARA* ist der Parameter, der an das Programm übergeben wird, dabei werden einfache Anführungszeichen durch doppelte ersetzt.

Eine bereits vorhandene Verknüpfung wird nicht überschrieben.

```
LINK "\windows\autostart\observer" ".\POIObserver.exe"
```

DLINK

Parameter : "*ZPFAD*" "*QPFAD*"

Wie *LINK*, vorhandene Verknüpfung wird überschrieben.

```
DLINK "\windows\autostart\observer" ".\POIObserver.exe"
```

nfRunner – the Scriptinterpreter

for mobile Windows

MKDIR

Parameter : "*ZPFAD*"

Erzeugt ein Verzeichnis *ZPFAD*. Am Ende darf kein Backslash stehen!

```
MKDIR "\windows\Mein Verzeichnis"
```

RMDIR

Parameter : "*ZPFAD*"

Löscht ein leeres ein Verzeichnis *ZPFAD*. Am Ende darf kein Backslash stehen!

```
RMDIR "\windows\Mein Verzeichnis"
```

SETVAR

Parameter: "*NAME*" "*WERT*"

Der Variablen *NAME* wird der Text, der in *WERT* steht zugewiesen. Pfadangaben werden hier auch expandiert. Bei *NAME* wird zwischen Groß- und Kleinschreibung unterschieden. Wenn *WERT* mit einer öffnenden runden Klammer beginnt, wird es als *AUSDRUCK* betrachtet und vor der Zuweisung berechnet. Es können alle Rechen- und Logik-Operatoren darin vorkommen.

```
SETVAR "Pfad" ".\"
SETVAR "Farbe" "($Blau$*256*256+$Gruen$*256+$Rot$) "
SETVAR "Gleich" "($Farbe1$=0xC0C0C0) & ($Farbe2$!=0xC0C0C0) "
SETVAR "HG" "@STRKEY(HKCU\ControlPanel\Desktop\wallpaper) "
```

SETSTRKEY

Parameter : "*KEY*" "*STRING*"

Schreibt den String-Wert *STRING* in den Registry-Schlüssel *KEY*.
Einfache Anführungszeichen werden durch doppelte Anführungszeichen ersetzt.

```
SETSTRKEY "HKCU\Software\Hasotec\VirtComMgrTMC\Source" "COM1 : "
```

nfRunner – the Scriptinterpreter

for mobile Windows

SETDWKEY

Parameter : "*KEY*" "*DWORD*"

Schreibt die Zahl *DWORD* in den Registry-Schlüssel *KEY*

```
SETDWKEY "HKLM\Explorer\ShowExt" "1"
```

SETPATHKEY

Parameter : "*KEY*" "*QPFAD*"

Wenn *QPFAD* mit ". \ " beginnt, werden diese Zeichen durch den Pfad der nfRun Datei ersetzt.
Einfache Anführungszeichen werden durch doppelte Anführungszeichen ersetzt.

```
SETPATHKEY "HKCU\Software\Hasotec\VirtComMgrTMC\ModuleFileName"  
".\VirtComMgrTMC.exe"
```

SETBINKEY

Parameter : "*KEY*" "*DATA*"

Schreibt *DATA* als binäre Daten in den Registry-Schlüssel *KEY*.

DATA ist eine Zeichenkette, bei der jeweils 2 HEX-Zeichen ein Byte darstellen.

Das Beispiel setzt die Mitteleuropäische Zeitzone. (Das ganze muss natürlich in einer Zeile stehen).
Es ist darauf zu achten, dass die maximale Zeilenlänge von 500 nicht überschritten wird.

```
SETBINKEY "HKLM\Time\TimeZoneInformation"  
"c4ffffff570065007300740065007500 72006f007000e4007300630068006500  
20004e006f0072006d0061006c007a00 65006900740000000000000000000000  
000000000000a000000050003000000 00000000000000005700650073007400  
6500750072006f007000e40073006300 68006500200053006f006d006d006500  
72007a00650069007400000000000000 0000000000000000000030000000500  
0200000000000000c4ffffff"
```

DELETEKEY

Parameter : "*KEY*"

Löscht den Registry-Schlüssel *KEY*. Falls der Schlüssel Unterschlüssel besitzt, werden diese auch gelöscht. Darum sollte man diesen Befehl mit Vorsicht anwenden.

```
DELETEKEY "HKCU\Software\Hasotec\VirtComMgrTMC"
```

nfRunner – the Scriptinterpreter

for mobile Windows

SETSTATE

Parameter: "*NR*" "*WERT*"

Setzt den Wert der Statusvariablen *NR*. Siehe auch *Remote Schnittstelle*:

```
SETSTATE "0" "1234"
```

SENDMESSAGE

Parameter: "*NAME*" "*MSG*,*WPAR*,*LPAR*"

Sendet eine Windows-Message *MSG* mit den Parametern *WPAR* und *LPAR* an das Fenster *NAME* und wartet auf die Bearbeitung durch die Anwendung. Das Ergebnis wird in der Variablen *SENDRESULT* abgelegt und evtl. Fehler stehen in der Variablen *LASTERROR*.

```
SENDMESSAGE "POI Observer" "273,1000,0"
```

POSTMESSAGE

Parameter: "*NAME*" "*MSG*,*WPAR*,*LPAR*"

Sendet eine Windows-Message *MSG* mit den Parametern *WPAR* und *LPAR* an das Fenster *NAME*, ohne auf die Bearbeitung zu warten..

```
POSTMESSAGE "POI Observer" "273,1000,0"
```

SENDCOMMAND

Parameter: "*NAME*" "*DWORD*"

Sendet ein *WM_COMMAND* Nachricht an das Fenster *NAME* und wartet bis die Anwendung die Nachricht bearbeitet hat. *DWORD* ist der Kommando Code. Das Ergebnis wird in der Variablen *SENDRESULT* abgelegt und evtl. Fehler stehen in der Variablen *LASTERROR*. Das Beispiel beendet den POI:Observer.

```
SENDCOMMAND "POI Observer" "1000"
```

POSTCOMMAND

Parameter: "*NAME*" "*DWORD*"

Sendet ein *WM_COMMAND* Nachricht an das Fenster *NAME*, ohne auf die Bearbeitung zu warten

```
POSTCOMMAND "POI Observer" "1000"
```

nfRunner – the Scriptinterpreter

for mobile Windows

CLOSEWINDOW

Parameter: "*NAME*" [*"KLASSE"*]

Sendet eine WM_CLOSE Nachricht an das Fenster *NAME*. Entspricht dem Anklicken des Close-Buttons. (Der POI:Observer lässt sich so nicht beenden!)

Optional kann auch der Klassenname des Fensters angegeben werden.

```
CLOSEWINDOW "\Storage Card"
```

ACTIVATEWINDOW

Parameter: "*NAME*" [*"KLASSE"*]

Bringt das Fenster *NAME* in den Vordergrund und aktiviert es.

Optional kann auch der Klassenname des Fensters angegeben werden.

```
ACTIVATEWINDOW "MainShell"
```

HIDEWINDOW

Parameter : "*NAME*" [*"KLASSE"*]

Macht das Fenster *NAME* unsichtbar.

Optional kann auch der Klassenname des Fensters angegeben werden.

```
HIDEWINDOW "EXEC"
```

SHOWWINDOW

Parameter: "*NAME*" [*"KLASSE"*]

Macht das Fenster *NAME* wieder sichtbar.

Optional kann auch der Klassenname des Fensters angegeben werden.

```
SHOWWINDOW "EXEC"
```

SETWINDOWPOS

Parameter: "*NAME*" "*X1*,*Y1*,*BREITE*,*HÖHE*" [*"KLASSE"*]

Setzt die Position des angegebenen Fensters. Die linke obere Ecke ist bei *X1*, *Y1* bestimmen die obere linke Ecke des Fensters. *BREITE*, *HÖHE* bestimmen die Breite und die Höhe des Fensters.

Optional kann auch der Klassenname des Fensters angegeben werden.

```
SETWINDOWPOS "" "0,0,100,100" "#NETCF_AGL_BASE_"
```

nfRunner – the Scriptinterpreter

for mobile Windows

SETTITEL

Parameter: "*TITEL*"

Ändert den Titel des Script-Fensters auf *TITEL*. Dieser Titel wird dann anstelle des Script-Namens in der Fensterliste angezeigt, wenn es nicht mit HIDEWINDOW ausgeblendet wurde.

```
SETTITEL "Mein Script"
```

SCREENREFRESH

Erzwingt ein Neuzeichnen des Bildschirms.

```
SCREENREFRESH
```

MOUSECLICK

Parameter: "*X*, *Y*" [*"NAME"*]

Simuliert einen Mausklick an der Position *X*, *Y*. Ist *NAME* angegeben, wird der Klick relativ zu linken oberen Ecke des Fensters *NAME* ausgeführt.

Sonst wird der Klick relativ zur linken oberen Ecke des Bildschirms ausgeführt.

```
MOUSECLICK "10,10"  
MOUSECLICK "1,1" "POBS"
```

MOUSEDRAW

Parameter: "*X1*, *Y1*, *Y2*, *Y2*" [*"NAME"*]

Simuliert ein "ziehen" mit der Maus von der Position *X1*, *Y1* zu der Position *X2*, *Y2*. Ist *NAME* angegeben, wird die Aktion relativ zu linken oberen Ecke des Fensters *NAME* ausgeführt.

Sonst wird die Aktion relativ zur linken oberen Ecke des Bildschirms ausgeführt.

```
MOUSEDRAW "10,10,100,100"
```

KEYDOWN

Parameter: "*VKCODE*"

Es wird simuliert, das die Taste mit dem virtuellen Key-Code *VKCODE* heruntergedrückt wird.

```
KEYDOWN "16"  
KEYTEXT "abcde"  
KEYUP "16"
```

nfRunner – the Scriptinterpreter

for mobile Windows

KEYUP

Parameter: "*VKCODE*"

Es wird simuliert, das die Taste mit dem virtuellen Key-Code *VKCODE* losgelassen wird.

```
KEYDOWN "16"  
  KEYTEXT "abcde"  
KEYUP "16"
```

KEYPRESS

Parameter: "*VKCODE*"

Es wird simuliert, das die Taste mit dem virtuellen Key-Code *VKCODE* heruntergedrückt und wieder losgelassen wird.

```
KEYPRESS "13"
```

KEYTEXT

Parameter: "*TEXT*"

Es wird simuliert, das der *TEXT* über die Tastatur eingegeben wird.

```
KEYTEXT "Dieser Text wird automatisch getippt"
```

WAITFORWINDOW

Parameter: "*NAME*" "*DWORD*" [*"KLASSE"*]

Wartet *DWORD* Millisekunden darauf, dass das Fenster *NAME* existiert.
Wenn *DWORD* = 0 ist wartet das Script solange, bis das Fenster existiert.
Optional kann auch der Klassenname des Fensters angegeben werden.

```
WAITFORWINDOW "POI Observer" "5000"
```

WFW

Parameter: "*NAME*" "*DWORD*" [*"KLASSE"*]

Kurzform für *WAITFORWINDOW*.

```
WFW "POI Observer" "5000"
```

nfRunner – the Scriptinterpreter

for mobile Windows

WAITFORCOLOR

Parameter: "*X,Y,COLOR[, MS]* "

Wartet darauf, das an der Position *X,Y* das Pixel die Farbe *COLOR* annimmt. Wenn *MS* angegeben wird, wird maximal *MS* Millisekunden gewartet. Ohne *MS* wird ewig gewartet.

```
WAITFORCOLOR "10,10,0,5000"
```

WFC

Parameter: "*X,Y,COLOR[, MS]* "

Kurzform von *WAITFORCOLOR*

```
WFC "10,10,0,5000"
```

WAITFORNOTCOLOR

Parameter: "*X,Y,COLOR[, MS]* "

Wartet darauf, das die Farbe *COLOR* an der Position *X,Y* verschwindet.
Wenn *ms* angegeben wird, wird maximal *MS* Millisekunden gewartet. Ohne *ms* wird ewig gewartet.

```
WAITFORNOTCOLOR "10,10,0,5000"
```

WFNC

Parameter: "*X,Y,COLOR[, MS]* "

Kurzform von *WAITFORNOTCOLOR*

```
WFNC "10,10,0,5000"
```

WAITFORFILE

Parameter: "*ZPFAD*" "*DWORD*"

Wartet *DWORD* Millisekunden auf die Datei *ZPFAD*, falls sie noch nicht existiert.
Wenn *DWORD* = 0 ist, wartet das Script solange, bis die Datei existiert.

```
WAITFORFILE "\\Storage Card" "0"
```

nfRunner – the Scriptinterpreter

for mobile Windows

WFF

Parameter: "*ZPFAD*" "*DWORD*"

Kurzform von *WAITFORFILE*.

```
WFF "\Storage Card" "0"
```

CREATEFILE

Parameter: "*ZPFAD*"

Erzeugt eine Textdatei mit Namen *ZPFAD*. Alle Zeilen danach werden in diese Datei geschrieben, bis zu Anweisung *ENDFILE*.

Führende Steuer und Leerzeichen bleiben erhalten

```
CREATEFILE "\Storage Card\MeinScript.nfRun"  
  IFYES "Mein Script" "Version anzeigen?"  
    SHOWVERSION  
  ENDIF  
ENDFILE
```

APPENDLINE

Parameter: "*ZPFAD*" "*TEXT*"

An die Datei *ZPFAD* wird eine Zeile mit *TEXT* angehängt. Anstatt doppelter müssen einfache Anführungszeichen genommen werden. Sie werden dann vom Programm ersetzt.

```
APPENDLINE "\Storage Card\MeinScript.nfRun" "RUN 'EXPLORER.EXE' "
```

APPENDFILE

Parameter: "*ZPFAD*"

Fügt der Datei *ZPFAD* Zeilen an.

Alle Zeilen danach werden in diese Datei geschrieben, bis zu Anweisung *ENDFILE*. Führende Leer- und Steuerzeichen bleiben erhalten.

```
APPENDFILE "\Storage Card\MeinScript.nfRun"  
# Das wird angefügt  
  IFYES "Mein Script" "Version anzeigen?"  
    SHOWVERSION  
  ENDIF  
ENDFILE
```

nfRunner – the Scriptinterpreter

for mobile Windows

CREATEINIFILE

Parameter: "*ZPFAD*"

Erzeugt eine Textdatei mit Namen *ZPFAD*. Alle Zeilen danach werden in diese Datei geschrieben, bis zu Anweisung *ENDFILE*. Führende Leer- und Steuerzeichen werden entfernt.

```
CREATEFILE "\\Storage Card\xy.ini"  
[GPS]  
COMPORT=COM2  
BAUD=4800  
ENDFILE
```

APPENDINIFILE

Parameter: "*ZPFAD*"

Fügt der Datei *ZPFAD* Zeilen an.

Alle Zeilen danach werden in diese Datei geschrieben, bis zu Anweisung *ENDFILE*. Führende Leer- und Steuerzeichen werden entfernt.

```
APPENDINIFILE "\\Storage Card\xy.ini"  
[GPS]  
COMPORT=COM2  
BAUD=4800  
ENDFILE
```

CLOSEFILE

Parameter: "*FHANDLE*"

Schließt die Datei mit dem Dateihandle *FHANDLE*, das von der Funktion *@OPENFILE* zurückgegeben wurde.

```
SETVAR "Datei" "@OPENFILE (".\Test.txt")  
CLOSEFILE "$Datei$"
```

WRITEWFC

Parameter: "*ZPFAD*" "*X* , *Y* [*,* *MS*]"

nfRunner – the Scriptinterpreter

for mobile Windows

Fügt eine Zeile an die Datei *ZPFAD* mit der Script Anweisung

WFC "X ,Y, COLOR [, MS] " an. Als *COLOR* wird die aktuelle Farbe an der Position *X, Y* genommen.

Alle Leer und Steuerzeichen die zwischen dem ersten Anführungszeichen und dem *X* Parameter stehen, werden an den Anfang der erzeugten Zeile gestellt.

```
WRITWFC "20,20"  
  
WRITWFC "40,36,5000"
```

WRITEWFNC

Parameter: "*ZPFAD*" "*X , Y [, MS] "*

Das gleiche wie bei *WRITEWFC*, nur hier wird die Anweisung *WFNC* geschrieben.

Alle Leer und Steuerzeichen die zwischen dem ersten Anführungszeichen und dem *X* Parameter stehen, werden an den Anfang der erzeugten Zeile gestellt.

```
WRITEWFNC "20,20"  
  
WRITEWFNC "40,36,5000"
```

WRITEIFC

Parameter: "*ZPFAD*" "*X , Y "*

fügt eine Zeile an die Datei *ZPFAD* mit der Script Anweisung

IFCOLOR "X ,Y, COLOR " an. Als *COLOR* wird die aktuelle Farbe an der Position *X, Y* genommen.

Alle Leer und Steuerzeichen die zwischen dem ersten Anführungszeichen und dem *X* Parameter stehen, werden an den Anfang der erzeugten Zeile gestellt.

```
WRITIFC "20,20"
```

RESET

Führt eine Soft-Reset aus.

```
RESET
```

HARDRESET

nfRunner – the Scriptinterpreter

for mobile Windows

Führt nach einer Sicherheitsabfrage einen Hard-Reset aus.

```
HARDREST
```

SLEEP

Parameter: "*DWORD*"

Wartet die angegebene Zeit in Millisekunden

```
SLEEP "1000"
```

SETSTORAGEMEMORY

Parameter: "*DWORD*"

Setzt die Größe des Storage Memorys (RAM-Disk) auf *DWORD* KB. Der restliche Hauptspeicher ist der Programmspeicher

```
SETSTORAGEMEMORY "1000"
```

SCREENROTATE

Parameter: "*INT*"

Dreht den gesamten Bildschirm. Bei positiven Werten wird der Bildschirm auf *INT* * 90° gedreht. Bei negativen Werten wird der Bildschirm relativ zu gegenwärtigen Ausrichtung um $\text{ABS}(\text{INT}) * 90^\circ$ gedreht.

```
SCREENROTATE "1"  
SCREENROTATE "-2"
```

SETVOLUME

Parameter: "*WERT*"

Setzt die Systemlautstärke. Das obere WORD ist der rechte und das untere WORD der linke Kanal. Die Werte gehen jeweils von 0 bis 0xFFFF.

```
SETVOLUME "0xFFFF0000"  
SETVOLUME "0x0000FFFF"
```

nfRunner – the Scriptinterpreter

for mobile Windows

PLAYSOUND

Parameter: "*PFAD*" [*"FLAG"*]

Spielt die WAV-Datei *PFAD* ab. Ist der optionale Parameter *FLAG* = 1, wird gewartet bis die Datei vollständig abgespielt ist. Sonst wird die Datei im Hintergrund abgespielt.

```
PLAYSOUND ".\Ansage.wav"  
PLAYSOUND "\Storage Card\POIObserver\Sounds\Altersheim.wav"
```

DEBUG

Schaltet den *DEBUG*-Modus ein.

Eine vorhandene LOG-Datei wird gelöscht. Auftretende Fehler werden in eine LOG-Datei geschrieben. Diese Datei hat den gleichen Namen wie die nfRun Datei mit der Endung LOG.

```
DEBUG
```

NODEBUG

Schaltet den *DEBUG*-Modus aus.

```
NODEBUG
```

MSG

Parameter: "*TEXT*"

Wenn *DEBUG* aktiviert ist, wird der *TEXT* in die LOG-Datei geschrieben

```
MSG "Es ist ein Fehler aufgetreten"
```

SHOWMSG

Parameter: "*TITEL*" "*TEXT*"

Es wird eine Dialogbox mit einem OK-Button geöffnet.

TITEL ist Kopfzeile des Dialogs, *TEXT* ist der Text der Dialogbox.

```
SHOWMSG "Obsinstaller" "Alles fertig"
```

nfRunner – the Scriptinterpreter

for mobile Windows

SHOWVERSION

Zeigt eine Dialogbox mit der aktuellen Programm-Version an.

```
SHOWVERSION
```

SHOWVMSG

Parameter: "*TITEL*" "*TEXT*"

Zeigt eine Dialogbox mit der aktuellen Programm Version an. Zusätzlich werden auch noch der *TITEL* und der *TEXT* angezeigt.

```
SHOWVMSG "nfRunner" "Der nfRunner ist registriert"
```

LOADTOMEMORY

Die Anweisung lädt den Rest des Scriptes in den Hauptspeicher, schließt die Script-Datei und führt anschließend das Script aus dem Hauptspeicher weiter aus. Diese Anweisung kann nicht innerhalb einer Bedingung oder Schleife erfolgen.

```
LOADTOMEMORY
```

QUIT

Parameter: "*[RET]*"

Beendet die Ausführung des Scripts und gibt den Wert der Variablen *SCRIPTRESULT* an das aufrufende Programm zurück. Wenn der optionale Parameter *RET* angegeben ist, wird dieser zurückgegeben.

```
QUIT
```

nfRunner – the Scriptinterpreter

for mobile Windows

A

ACTIVATEWINDOW.....	16
APPENDFILE.....	20
APPENDLINE.....	20
APPENINIDFILE.....	21

C

CLOSEFILE	21
CLOSEWINDOW.....	16
COPY.....	11
CREATEFILE.....	20
CREATEINIFILE.....	21

D

DCOPY.....	12
DEBUG.....	24
DELETE.....	12
DELETEKEY.....	14
DLINK.....	12

E

ELSE.....	9
ELSEIF.....	9
ENDIF.....	9
ENDWHILE	10

H

HARDRESET.....	22
HIDEWINDOW.....	16

I

IF	6
IFCOLOR.....	7
IFDWKEY.....	9
IFFEQ.....	7
IFFILEEQUAL.....	7
IFFILEEXIST.....	6
IFSTRKEY.....	9
IFWE.....	8
IFWINDOWEXIST.....	8
IFWINDOWVISIBLE.....	8
IFWV.....	8
IFYES.....	6

K

KEYDOWN.....	17
KEYPRESS.....	18
KEYTEXT.....	18
KEYUP.....	18

L

LINK.....	12
LOADTOMEMORY.....	25

M

MKDIR.....	13
MOUSECLICK.....	17
MOUSEDRAW.....	17
MOVE.....	12
MSG.....	24

N

NFRAW.....	11
NFRUN.....	11
NFRUNAGAIN.....	11
NFRUNANDWAIT.....	11
NODEBUG.....	24

P

PLAYSOUND.....	24
POSTCOMMAND.....	15
POSTMESSAGE.....	15

Q

QUIT.....	25
-----------	----

R

RAW.....	10
Remote Schnittstelle.....	5
RESET.....	22
RMDIR.....	13
RUN.....	10
RUNANDWAIT.....	10

S

SCREENREFRESH.....	17
SCREENROTATE.....	23
SENDCOMMAND.....	15
SENDMESSAGE.....	15
SETBINKEY.....	14
SETDWKEY.....	14
SETPATHKEY.....	14
SETSTATE.....	15
SETSTORAGEMEMORY.....	23
SETSTRKEY.....	13
SETTITEL.....	17
SETVAR.....	13
SETVOLUME.....	23
SETWINDOWPOS.....	16
SHOWMSG.....	24
SHOWVERSION.....	25
SHOWVMSG.....	25
SHOWWINDOW.....	16
SLEEP.....	23

W

WAITFORCOLOR.....	19
WAITFORFILE.....	19
WAITFORNOTCOLOR.....	19
WAITFORWINDOW.....	18
WFC.....	19
WFF.....	20
WFNC.....	19
WFW.....	18
WHILE	10
WRITEIFC.....	22
WRITEWFC.....	21
WRITEWFNC.....	22