Befehlssprache für die Skriptverarbeitung des DFÜ-Netzwerks

Inhalt

1.0 Überblick

Um die Verbindung mit einem Internet-Dienstanbieter oder Online-Dienst herzustellen, ist in der Regel die Eingabe eines Benutzernamens und eines Kennworts erforderlich.
Dank der Skriptunterstützung für DFÜ-Netzwerke können Sie diesen Anmeldevorgang skriptgesteuert automatisieren.

Ein Skript ist eine Textdatei mit den Befehlen, Parametern und Ausdrücken, die zum Herstellen der Verbindung mit dem Internet-Dienstanbieter sowie zur Nutzung des Dienstes erforderlich sind. Zum Erstellen einer Skriptdatei können Sie beispielsweise den Microsoft-Editor verwenden.
Mit der erstellten Skriptdatei können Sie dann unter Verwendung des Programms “DFÜ-Skriptverwaltung” eine DFÜ-Netzwerkverbindung aufbauen.

2.0 Grundstruktur eines Skripts

Der Befehl ist das grundlegende Element eines Skripts. Einige Befehle erfordern die Angabe von Parametern, die näher definieren, welche Funktion der Befehl erfüllen soll. Ein Ausdruck ist eine Zusammenstellung von Operatoren und Argumenten, die ein Ergebnis haben. Ausdrücke können in Befehlen als Werte verwendet werden. Ausdrücke sind beispielsweise arithmetische und relationale Vergleiche und Verkettungen von Zeichenfolgen.

Die Grundstruktur eines Skripts wird im folgenden veranschaulicht:

```
;
; Ein Kommentar beginnt mit einem Semikolon und
; endet am Ende der Zeile.
;

proc main
       ;  Ein Skript enthält eine beliebige Anzahl Variablen
       ;  und Befehle

       Variablendeklaration

       Befehlsblock
```

```
endproc
```

Jedes Skript muß eine Hauptprozedur haben, die mit dem Schlüsselwort **proc** beginnt und dem Schlüsselwort **endproc** endet.

Sie müssen zunächst Variablen deklarieren, bevor Sie Befehle hinzufügen. Die Befehle der Hauptprozedur werden nacheinander ausgeführt, und zwar in der Reihenfolge, in der sie im Skript stehen. Die Skriptverarbeitung wird beendet, wenn das Ende der Hauptprozedur erreicht ist.

3.0 Variablen

Skripts können Variablen enthalten. Die Namen der Variablen müssen mit einem Buchstaben oder Unterstrich ('_') beginnen und dürfen eine beliebige Kombination aus Groß- und Kleinbuchstaben, Zahlen und Unterstrichen enthalten. Reservierte Wörter sind als Variablennamen nicht zugelassen. Die Liste der reservierten Wörter finden Sie am Ende dieses Dokuments.

Variablen müssen deklariert werden, bevor Sie sie verwenden können. Bei der Variablendeklaration definieren Sie u.a. den Typ der Variablen. Einer Variable bestimmten Typs darf nur ein Wert des angegebenen Typs zugewiesen werden. Folgende drei Variablentypen werden unterstützt:

Typ	Beschreibung
Integer	Eine negative oder positive Ganzzahl, wie 7, -12 oder 5698.
String	Eine Zeichenfolge in Anführungszeichen; z.B. "Benutzerkennung:" oder "Enter password:".
Boolean	Ein boolscher logischer Ausdruck, wie TRUE (WAHR) oder FALSE (FALSCH).

Variablen stehen für die zugewiesenen Werte, die folgendermaßen festgelegt werden:

Variable **=** *Ausdruck*

Die Variable nimmt den im Ausdruck berechneten Wert an.

Beispiele:

```
integer count = 5
integer timeout = (4 * 3)
integer i

boolean bDone = FALSE

string szIP = (getip 2)

set ipaddr szIP
```

3.1 Systemvariablen

Der Wert einer Systemvariablen wird durch einen Skriptbefehl bestimmt oder durch Informationen, die Sie beim Einrichten einer DFÜ-Netzwerkverbindung eingegeben haben. Systemvariablen haben Schreibschutzstatus, d.h. sie können innerhalb des Skripts nicht geändert werden.
Die Systemvariablen werden im folgenden aufgeführt:

Name	Typ	Beschreibung

$USERID	String	Ihre Benutzerkennung für die gewählte Verbindung. Der Wert dieser Variable ist der Benutzername, der im Dialogfeld **Verbinden mit** des DFÜ-Netzwerks als **Benutzername** eingegeben wurde.
$PASSWORD	String	Das Kennwort für die gewählte Verbindung. Der Wert dieser Variable ist das Kennwort, das im Dialogfeld **Verbinden mit** des DFÜ-Netzwerks als **Kennwort** eingegeben wurde.
$SUCCESS	Boolean	Diese Variable wird von einigen Befehlen gesetzt, um zu signalisieren, daß der Befehl ausgeführt wurde. Die Ausführung eines Skripts kann, abhängig vom Wert dieser Variablen, in seinem Ablauf beeinflußt werden.
$FAILURE	Boolean	Diese Variable wird von einigen Befehlen gesetzt, um zu signalisieren, daß der Befehl nicht ausgeführt wurde. Die Ausführung eines Skripts kann, abhängig vom Wert dieser Variablen, in seinem Ablauf beeinflußt werden.

Die Systemvariablen können überall dort eingesetzt werden, wo der Typ der Variable erlaubt ist. Beispiel:

```
transmit $USERID
```

Dieser Befehl ist gültig, da $USERID eine Variable des Typs String (Zeichenfolge) ist.

4.0 Kürzel für Sonderzeichen

Die Skriptbefehlsprache des DFÜ-Netzwerks unterstützt Escape-Sequenzen und die Steuerzeichenumwandlung, die im folgenden beschrieben werden.

Kürzel	Beschreibung
^*Zeichen*	Steuerzeichenumwandlung Wenn *Zeichen* einem Wert zwischen '@' und '_' entspricht, wird die Steuerzeichenfolge in einen Einzelbytewert zwischen 0 und 31 übersetzt. Beispielsweise wird **^M** in ein Wagenrücklaufzeichen umgewandelt. Wenn *Zeichen* einem Wert zwischen ‘a’ und ‘z’ (Kleinbuchstabenalphabet) entspricht, wird die Steuerzeichenfolge in einen Einzelbytewert zwischen 1 und 26 umgewandelt Wenn *Zeichen* einem anderen Wert entspricht, wird die Zeichenfolge nicht umgewandelt.
<cr>	Wagenrücklauf
<lf>	Zeilenvorschub
\"	Anführungszeichen
\^	Einzelnes Caret
\<	Einzelnes '<'
****	Umgekehrter Schrägstrich

Beispiele:

```
transmit "^M"       ;Eingabetaste (Wagenrücklauf) senden
transmit "peter^M"        ;„peter“ und Eingabetaste senden
```

```
transmit "<cr><lf>"       ;Eingabetaste und Zeilenvorschub senden
waitfor "<cr><lf>"        ;Auf Eingabetaste und Zeilenvorschub warten
```

5.0 Ausdrücke

Ein Ausdruck ist eine Verknüpfung aus Operatoren und Argumenten, die ein Ergebnis hat. Ausdrücke können in Befehlen als Werte verwendet werden.

In einem Ausdruck werden Variablen oder Integer-, String- bzw. Boolean-Werte mit den unären oder binären Operatoren verknüpft, die in der folgenden Tabelle aufgeführt sind. Die unären Operatoren haben Vorrang. Die Rangfolge der binären Operatoren können Sie an der Reihenfolge in der Tabelle ersehen.

Unäre Operatoren:

Operator	Typ der Berechnung
-	Unäres Minus
!	Komplementär eines Wertes

In der folgenden Tabelle werden die binären Operatoren in der Reihenfolge ihres Rangs aufgeführt. Operatoren mit hohem Rang werden zuerst genannt.

Operatoren	Typ der Berechnung	Typenbeschränkung
* /	Multiplikation	Integer
+ -	Addition	Integer, (String nur +)
< > <= >=	Relation	Integer
== !=	Gleichheit	Integer, String, Boolean
and	Logisches UND	Boolean
or	Logisches ODER	Boolean

Beispiele:

```
count = 3 + 5 * 40
transmit "Hallo" + " Nachbar"
delay 24 / (7 - 1)
```

6.0 Kommentare

Text, der in einer Zeile steht, die mit einem Semikolon beginnt, wird nicht berücksichtigt.

Beispiel:

```
; Dies ist ein Kommentar

transmit "Hallo"          ; Überträgt die Zeichenfolge "Hallo"
```

7.0 Schlüsselwörter

Schlüsselwörter bestimmen den Aufbau eines Skripts. Im Gegensatz zu Befehlen lösen sie keine Aktion aus. Im folgenden werden die Schlüsselwörter aufgelistet.

proc *Name*

Kennzeichnet den Anfang einer Prozedur. Jedes Skript muß eine Hauptprozedur (**proc** main) haben. Die Ausführung des Skripts beginnt am Anfang der Hauptprozedur und endet am Schluß der Hauptprozedur.

endproc

Kennzeichnet das Ende einer Prozedur. Sobald ein Skript bis zur Anweisung **endproc** der Hauptprozedur ausgeführt wurde, startet das DFÜ-Netzwerk PPP oder SLIP.

integer *Name* [**=** *Wert*]

Deklariert eine Variable des Typs Integer. Sie können einen beliebigen numerischen Ausdruck oder eine Variable verwenden, um den Anfangswert der Variablen festzulegen.

string *Name* [**=** *Wert*]

Deklariert eine Variable des Typs String (Zeichenfolge). Sie können eine beliebige Zeichenfolge oder String-Variable verwenden, um den Anfangswert der Variablen festzulegen.

boolean *Name* [**=** *Wert*]

Deklariert eine Variable des Typs Boolean. Sie können einen beliebigen boolschen Ausdruck oder eine boolsche Variable verwenden, um den Anfangswert der Variablen festzulegen.

8.0 Befehle

Alle Befehle sind zugleich reservierte Wörter. Sie dürfen also keine Variable mit dem Namen eines Befehls deklarieren. Die Befehle werden im folgenden aufgeführt:

delay *nSekunden*

Wartet für die Dauer des mit *nSekunden* angegebenen Zeitraums, bis der nächste Skriptbefehl ausgeführt wird.

Beispiele:

```
delay 2       ; Wartet 2 Sekunden
delay x * 3   ; Wartet x * 3 Sekunden
```

getip *Wert*

Fordert eine IP-Adresse vom Remote-Computer an. Wenn der Internet-Dienstanbieter mehrere IP-Adressen in einer Zeichenfolge zurücksendet, können Sie mit dem Parameter *Wert* näher bestimmen, welche IP-Adresse das Skript verwenden soll.

Beispiele:

```
; Liest die zweite IP-Adresse:
set ipaddr getip 2

; Weist einer Variable die erste empfangene IP-Adresse zu:
szAddress = getip
```

goto *Sprungmarke*

Springt an die Stelle des Skripts, die mit *Sprungmarke* gekennzeichnet wurde, und führt das Skript ab dieser Stelle weiter aus.

Beispiel:

```
waitfor "Prompt>" until 10
if !$SUCCESS then
     goto Ausstieg; Springt zu Ausstieg und führt Befehle
                          ; aus, die dieser Marke folgen
endif

transmit "bbs^M"
goto End

Ausstieg:
  transmit "^M"
```

halt

Hält das Skript an. Mit diesem Befehl wird nicht das Terminalfenster geschlossen. Sie müssen auf die Schaltfläche **Weiter (F7)** klicken, um die Verbindung herzustellen. Sie können das Skript nicht neu starten.

if *Bedingung* **then**
Befehle
endif

Führt die *Befehle* aus, wenn der Ausdruck in *Bedingung* den Wert TRUE (WAHR) annimmt.

Beispiel:

```
if $USERID == "Paula" then
       transmit "Paulinchen^M"
endif
```

Sprungmarke:

Bestimmt eine Stelle, an die Sie im Skript verzweigen möchten. Eine Sprungmarke muß eindeutig sein. Für sie gelten außerdem die Benennungsregeln von Variablen.

set port databits 5 | 6 | 7 | 8

Ändert die Anzahl der Bits je übertragenem oder empfangenem Byte. Die Anzahl der Bits kann zwischen 5 und 8 liegen. Wenn Sie diesen Befehl nicht verwenden, gelten die für die DFÜ-Verbindung eingestellten Eigenschaften.

Beispiel:

```
set port databits 7
```

set port parity none | odd | even | mark | space

Ändert die Parität der Schnittstelle für die DFÜ-Verbindung. Wenn Sie diesen Befehl nicht verwenden, gelten die für die DFÜ-Verbindung eingestellten Eigenschaften.

Beispiel:

```
set port parity even
```

set port stopbits 1 | 2

Ändert die Anzahl der Stopbits für die Sitzung. Der Wert kann entweder 1 oder 2 betragen. Wenn Sie diesen Befehl nicht verwenden, gelten die für die DFÜ-Verbindung eingestellten Eigenschaften.

Beispiel:

```
set port stopbits 2
```

set screen keyboard on | off

Ermöglicht oder unterbindet Tastatureingaben im Terminalfenster während der Ausführung des Skripts.

Beispiel:

```
set screen keyboard on
```

set ipaddr *Zeichenfolge*

Gibt die IP-Adresse der Arbeitsstation für die Sitzung an. Die *Zeichenfolge* muß dem Format einer IP-Adresse entsprechen.

Beispiel:

```
szIPAddress = "194.97.97.103"
set ipaddr szIPAddress

set ipaddr "194.97.97.103"

set ipaddr getip
```

transmit *Zeichenfolge* [**, raw**]

Sendet den mit *Zeichenfolge* angegebenen Text an den Remote-Computer.

Normalerweise erkennt der Remote-Computer Steuerzeichenfolgen und Escape-Sequenzen, es sei denn, Sie geben den Parameter **raw** an. Der Parameter **raw** ist zweckmäßig, wenn die Systemvariablen $USERID und $PASSWORD Zeichenfolgen enthalten, die ohne den **raw** Parameter als Escape-Sequenzen oder Steuerzeichenfolgen gedeutet würden.

Beispiele:

```
transmit "slip-103" + "^M"
transmit $USERID, raw
```

waitfor *Zeichenfolge* [**, matchcase**] [**then** *Sprungmarke*
{ **,** *Zeichenfolge* [**, matchcase**] **then** *Sprungmarke* }]
[**until** *Zeit*]

Wartet, bis der Computer eine oder mehrere *Zeichenfolge*n vom Remote-Computer empfängt. Der Parameter *Zeichenfolge* berücksichtigt die Groß-/Kleinschreibung, es sei denn, Sie verwenden den Parameter **matchcase**.

Wenn eine entsprechende Zeichenfolge empfangen wurde, wird gegebenenfalls der Parameter **then** *Sprungmarke* ausgeführt. Dieser bewirkt, daß das Skript zur *Sprungmarke* verzweigt und dort die nachfolgenden Befehle abgearbeitet werden.

Der optionale Parameter **until** *Zeit* legt die maximale Zeit in Sekunden fest, die der Computer auf den Empfang der *Zeichenfolge* wartet, bevor das Skript fortgesetzt wird. Ohne diesen Parameter würde der Computer unbegrenzt warten.

Wenn der Computer eine der angegebenen Zeichenfolgen empfängt, wird die Systemvariable $SUCCESS auf TRUE (WAHR) gesetzt, andernfalls (falls nach Verstreichen der *Zeit* die angegebene Zeichenfolge nicht empfangen wurde) auf FALSE (FALSCH).

Beispiele:

```
waitfor "Login:"

waitfor "Password?", matchcase

waitfor "prompt>" until 10

waitfor
       "Login:"     then DoLogin,
       "Password:"  then DoPassword,
       "BBS:"       then DoBBS,
       "Other:"     then DoOther
  until 10
```

while *Bedingung* **do**
Befehle
endwhile

Führt die *Befehle* aus, bis der Ausdruck in *Bedingung* den Wert FALSE (FALSCH) annimmt.

Beispiel:

```
integer count = 0

while count < 4 do
       transmit "^M"
       waitfor "Login:" until 10
       if $SUCCESS then
              goto DoLogin
       endif
       count = count + 1
endwhile
...
```

9.0 Reservierte Wörter

Im folgenden werden die reservierten Wörter aufgeführt, die nicht als Variablennamen verwendet werden dürfen:

and	boolean	databits	delay
do	endif	endproc	endwhile
even	FALSE	getip	goto

halt	if	integer	ipaddr
keyboard	mark	matchcase	none
odd	off	on	or
parity	port	proc	raw
screen	set	space	stopbits
string	then	transmitTRUE	
until	waitfor	while	