Język skryptów poleceń Dial-Up
do obsługi programu Dial-Up Networking

Spis treści

1.0 Przegląd
2.0 Podstawowa struktura skryptów
3.0 Zmienne
    3.1 Zmienne systemowe
4.0 Ciągi literałów
5.0 Wyrażenia
6.0 Komentarze
7.0 Słowa kluczowe
8.0 Polecenia
9.0 Słowa zarezerwowane

1.0 Przegląd

Wielu dostawców usług sieci Internet oraz wiele usług online wymaga do uzyskania połączenia ręcznego wprowadzania informacji, takich jak nazwa użytkownika i hasło.
Korzystając z pomocy obsługi skryptów dla programu Dial-Up Networking można pisać skrypty w celu automatyzacji tego procesu.

Skrypt jest plikiem tekstowym zawierającym sekwencję poleceń, parametrów i wyrażeń wymaganych przez dostawcę usług w sieci Internet lub usług sieciowych w celu uzyskania połączenia i skorzystania z usługi. Do utworzenia pliku skryptu można wykorzystać dowolny edytor tekstowy, taki jak Notatnik Microsoft. Po utworzeniu pliku skryptu, można go przypisać do określonego połączenia wybieranego telefonicznie przez uruchomienie programu Dial-Up Scripting Tool.

2.0 Podstawowa struktura skryptu

Polecenie jest podstawową instrukcją zawartą w skrypcie. Niektóre polecenia wymagają parametrów, które definiują, co polecenie ma wykonać. Wyrażenie jest kombinacją operatorów i argumentów tworzących wynik. Wyrażenia mogą być używane jako wartości w dowolnym poleceniu. Przykłady wyrażeń zawierają działania arytmetyczne, porównania i konkatenację ciągów.

Podstawową formą skryptu dla programu Dial-Up Networking jest:

```
;
; Komentarz zaczynający się średnikiem
; i sięgający końca wiersza.
;

proc main
      ;  Skrypt może mieć dowolną liczbę zmiennych
      ;  i poleceń

      deklaracje zmiennych
```

```
        blok poleceń

endproc
```

Skrypt musi mieć procedurę główną określoną słowem kluczowym **proc** i odpowiednim słowem kluczowym **endproc**, wskazującym koniec procedury.

Należy zadeklarować zmienne przed wpisaniem poleceń. Najpierw jest wykonywane pierwsze polecenie w procedurze głównej, a następnie są wykonywane dalsze polecenia zgodnie z kolejnością, w jakiej występują w skrypcie. Skrypt kończy się po osiągnięciu końca procedury głównej.

3.0 Zmienne

Skrypty mogą zawierać zmienne. Nazwy zmiennych muszą zaczynać się literą lub znakiem podkreślenia ('_') i mogą zawierać dowolną kombinację wielkich i małych liter, cyfr i podkreśleń. Jako nazw zmiennych nie wolno używać słów zarezerwowanych. Dodatkowe informacje można znaleźć na liście słów zarezerwowanych na końcu niniejszego dokumentu.

Nazwy zmiennych należy zadeklarować przed ich użyciem. Deklarując zmienną należy także zdefiniować jej typ. Zmienna konkretnego typu może zawierać wartości tego samego typu. Obsługiwane są następujące trzy typy zmiennych:

Typ	Opis
integer	Liczba całkowita ujemna lub dodatnia, np. 7, -12 lub 5698.
string	Ciąg znaków zamkniętych w podwójnym cudzysłowie, np. "Halo, co słychać!" albo "Wprowadź hasło:".
boolean	Logiczna wartość TRUE (Prawda) lub FALSE (Fałsz).

Zmiennym przypisuje się wartości za pomocą poniższej instrukcji przypisania:

zmienna **=** *wyrażenie*

Zmienna przybiera wartość obliczonego wyrażenia.

Przykłady:

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

boolean bDone = FALSE

string szIP = "getip 2"

set ipaddr szIP
```

3.1 Zmienne systemowe

Zmienne systemowe są ustawiane przez polecenia skryptu lub są ustalane przez wprowadzane informacje podczas ustanawiania połączenia Dial-Up Networking. Zmienne systemowe są tylko do odczytu, co oznacza, że nie można ich zmienić w skrypcie. Zmiennymi systemowymi są:

Nazwa	Typ	Opis

$USERID String Identyfikacja użytkownika w bieżącym połączeniu. Zmienna ta jest wartością nazwy użytkownika określoną w oknie dialogowym Połącz z programu Dial-Up Networking.

$PASSWORD String Hasło dla bieżącego połączenia. Ta zmienna jest wartością hasła użytkownika określoną w oknie dialogowym Połącz z programu Dial- Up Networking.

$SUCCESS Boolean Ta zmienna jest ustawiana przez pewne polecenia w celu wskazania, czy polecenie zostało wykonane. W skrypcie mogą być podejmowane decyzje na podstawie wartości tych zmiennych.

$FAILURE Boolean Ta zmienna jest ustawiana przez pewne polecenia w celu wskazania, czy polecenie zakończyło się niepowodzeniem. W skrypcie mogą być podejmowane decyzje na podstawie wartości tych zmiennych.

Zmienne te mogą być używane wszędzie tam, gdzie jest używane wyrażenie podobnego typu. Na przykład,

```
transmit $USERID
```

jest poprawnym poleceniem, ponieważ $USERID jest zmienną typu string.

4.0 Ciągi literałów

Skrypty dla programu Dial-Up Networking obsługują sekwencje ESC i tłumaczenie sekwencji CTRL, jak opisano poniżej.

Ciąg literałów	Opis
^*znak*	Tłumaczenie znaku w sekwencji CTRL Jeśli *znak* jest wartością między '@' i '_', sekwencja znaków jest tłumaczona na wartość jednobajtową zawartą między 0 i 31. Na przykład ^M jest tłumaczone na powrót karetki. Jeśli *znak* jest wartością między a i z, sekwencja znaków jest tłumaczona na wartość jednobajtową zawartą między 1 i 26. Jeśli *znak* jest inną wartością, sekwencja znaków nie jest traktowana specjalnie.
<cr>	Powrót karetki
<lf>	Zmiana wiersza
\"	Podwójny cudzysłów
\^	Pojedynczy znak karetki
\<	Pojedynczy znak '<'
****	Kreska ukośna w lewo

Przykłady:

```
transmit "^M"
transmit "Joe^M"
transmit "<cr><lf>"
waitfor "<cr><lf>"
```

5.0 Wyrażenia

Wyrażenie jest kombinacją operatorów i argumentów tworzących wynik. Wyrażenia mogą być używane jako wartości w dowolnym poleceniu.

Wyrażenie może zawierać dowolną zmienną, wartości typu integer, string, boolean wraz z dowolnym operatorem jednoargumentowym lub dwuargumentowym, podanym w poniższych tablicach. Wszystkie operatory jednoargumentowe mają wyższy priorytet. Priorytet operatorów dwuargumentowych jest wskazany przez ich położenie w tablicy.

Operatorami jednoargumentowymi są:

Operator	Typ operacji
-	Minus jednoargumentowy
!	Dopełnienie do jedynki

Operatory dwuargumentowe są przedstawione w poniższej tablicy w kolejności priorytetu. Operatory o wyższym priorytecie są zgrupowane na początku:

Operatory	Typ operacji	Ograniczenia typu
* /	Multiplikacja	Całkowite
+ -	Addycja	Całkowite, ciągi (tylko +)
< > <= >=	Porównanie	Całkowite
== !=	Równość	Całkowite, ciągi, logiczne
and	Logiczna AND	Logiczne
or	Logiczna OR	Logiczne

Przykłady:

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

6.0 Komentarze

Każdy tekst w wierszu po średniku jest ignorowany.

Przykłady:

```
; to jest komentarz

transmit "halo"           ; wysyła ciąg "halo"
```

7.0 Słowa kluczowe

Słowa kluczowe określają strukturę skryptu. W przeciwieństwie do poleceń, nie wykonują one żadnych czynności. Poniżej przedstawiono listę słów kluczowych.

proc *nazwa*

Wskazuje początek procedury. Wszystkie skrypty muszą mieć procedurę główną (**proc** main). Wykonanie skryptu zaczyna się od procedury głównej i kończy się zakończeniem tej procedury.

endproc

Wskazuje koniec procedury. Gdy podczas wykonywania skryptu napotykane jest słowo kluczowe **endproc** w procedurze głównej, program Dial-Up Networking uruchamia protokół PPP lub SLIP.

integer *nazwa* [**=** *wartość*]

Deklaruje zmienną typu integer (liczba całkowita). Do inicjowania zmiennej można wykorzystać dowolne wyrażenie numeryczne lub zmienną.

string *nazwa* [**=** *wartość*]

Deklaruje zmienną typu string (ciąg). Do inicjowania zmiennej można wykorzystać dowolny ciąg literałów lub zmienną.

boolean *nazwa*[**=** *wartość*]

Deklaruje zmienną typu boolean (logiczna). Do inicjowania zmiennej można wykorzystać dowolne wyrażenie logiczne lub zmienną.

8.0 Polecenia

Wszystkie polecenia są słowami zarezerwowanymi, co oznacza, że nie wolno deklarować zmiennych o takich samych nazwach jak polecenia. Poniżej przedstawiono listę poleceń:

delay *nSekundy*

Przerywa pracę na liczbę sekund określoną parametrem *nSekundy* przed wykonaniem następnego polecenia w skrypcie.

Przykłady:

```
delay 2      ; przerywa na 2 sekundy
delay x * 3  ; przerywa na x * 3 sekundy
```

getip *wartość*

Czeka na adres IP, który ma być odebrany ze zdalnego komputera. Jeśli dostawca usługi Internet zwraca wiele adresów IP w ciągu, należy zastosować parametr *wartość* do określenia, który adres IP ma być użyty przez skrypt.

Przykłady:

```
; weź drugi adres IP
set ipaddr getip 2

; przypisz pierwszy odebrany adres IP do zmiennej
szAddress = getip
```

goto *etykieta*

Wykonuje skok do miejsca w skrypcie określonego przez *etykietę* i kontynuuje wykonywanie poleceń po tej etykiecie.

Przykład:

```
waitfor "Prompt>" until 10
if !$SUCCESS then
```

```
        goto BailOut ; wykonuje skok do BailOut i wykonuje
                     ; polecenia po tej etykiecie
    endif

    transmit "bbs^M"
    goto End

BailOut:
    transmit "^M"
```

halt

Wstrzymuje wykonanie skryptu. Polecenie to nie usuwa okna dialogowego terminala. Aby ustanowić połączenie, należy kliknąć polecenie Continue (Kontynuuj). Skryptu nie można ponownie uruchomić.

if *warunek* **then**
polecenia
endif

Wykonuje sekwencję *poleceń*, jeśli *warunek* jest TRUE (Prawda).

Przykład:

```
if $USERID == "John" then
    transmit "Johnny^M"
endif
```

etykieta **:**

Określa miejsce w skrypcie, do którego ma nastąpić skok. Etykieta musi być nazwą unikatową oraz być zgodna z konwencjami nazewnictwa zmiennych.

set port databits 5 | 6 | 7 | 8

Zmienia liczbę bitów w bajtach nadawanych i odbieranych podczas sesji. Liczba bitów może być równa od 5 do 8. Jeśli to polecenie nie zostanie dołączone, program Dial-Up Networking zastosuje ustawienia właściwości określone dla połączenia.

Przykład:

```
set port databits 7
```

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

Zmienia schemat parzystości dla portu podczas sesji. Jeśli to polecenie nie zostanie dołączone, program Dial-Up Networking zastosuje ustawienia właściwości określone dla połączenia.

Przykład:

```
set port parity even
```

set port stopbits 1 | 2

Zmienia liczbę bitów stop dla portu podczas sesji. Liczba może być równa 1 lub 2. Jeśli to polecenie nie zostanie dołączone, program Dial-Up Networking zastosuje ustawienia właściwości określone dla połączenia.

Przykład:

```
set port stopbits 2
```

set screen keyboard on | off

Włącza lub wyłącza klawiaturę w oknie terminala skryptu.

Przykład:

```
set screen keyboard on
```

set ipaddr *ciąg*

Określa adres IP stacji roboczej dla sesji. *Ciąg* musi mieć postać adresu IP.

Przykłady:

```
szIPAddress = "11.543.23.13"
set ipaddr szIPAddress

set ipaddr "11.543.23.13"

set ipaddr getip
```

transmit *ciąg* [**, raw**]

Wysyła znaki określone przez *ciąg* do zdalnego komputera.

Zdalny komputer będzie rozpoznawać sekwencje ESC i CTRL, jeśli w poleceniu nie zostanie dołączony parametr **raw**. Parametr **raw** jest użyteczny podczas nadawania zmiennych systemowych $USERID i $PASSWORD, gdy nazwa użytkownika lub hasło zawiera sekwencje znaków, które bez parametru **raw** byłyby interpretowane jako sekwencje CTRL lub ESC.

Przykłady:

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

waitfor *ciąg* [**, matchcase**] [**then** *etykieta*
{ **,** *ciąg* [**, matchcase**] **then** *etykieta* }]
[**until** *czas*]

Czeka, aż komputer odbierze jeden lub kilka określonych ciągów ze zdalnego komputera. W parametrze *ciąg* nie jest rozróżniana wielkości liter, jeśli nie zostanie dołączony parametr **matchcase**.

Jeśli zostanie odebrany pasujący ciąg, a w poleceniu użyto parametru **then** *etykieta*, to nastąpi skok do miejsca w skrypcie oznaczonego przez parametr *etykieta*.

Opcjonalny parametr **until** *czas* definiuje maksymalną liczbę sekund, przez które komputer będzie czekać na odebranie ciągu przed wykonaniem następnego polecenia. Bez podania tego parametru komputer będzie czekać nieskończenie długo.

Jeśli komputer odbierze jeden z określonych ciągów, zmienna systemowa $SUCCESS jest ustawiana na wartość TRUE (Prawda). W przeciwnym razie jest ona ustawiana na wartość FALSE (Fałsz), jeśli liczba sekund określona przez parametr *czas* upłynie, zanim zostanie odebrany ciąg.

Przykłady:

```
waitfor "Login:"

waitfor "Hasło?", matchcase

waitfor "prompt>" until 10

waitfor
       "Login:"     then DoLogin,
       "Hasło:"     then DoPassword,
       "BBS:"       then DoBBS,
       "Inne:"      then DoOther
  until 10
```

while *warunek* **do**
polecenia
endwhile

Wykonuje sekwencję *poleceń* aż do chwili, gdy *warunek* przyjmie wartość FALSE (Fałsz).

Przykład:

```
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 Słowa zarezerwowane

Następujące słowa są zarezerwowane i nie mogą być używane jako nazwy zmiennych.

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	transmit	TRUE	
until	waitfor	while		