ダイヤルアップ ネットワーク スクリプト用
ダイヤルアップ スクリプト コマンド言語

目次

1.0 概要

通常、インターネット アクセス プロバイダやオンライン サービスでは、接続を確立するために、ユーザー名、パスワードなどの情報を入力する必要があります。ダイヤルアップ ネットワーク スクリプトを使うと、このような操作を自動化するスクリプトを作成できます。

スクリプトは、コマンド、パラメータ、および式が記述されたテキスト ファイルです。このコマンド、パラメータ、および式はインターネット アクセス プロバイダやオンライン サービスとの接続の確立、およびサービスを使うために必要な情報を自動的に入力するために必要です。スクリプト ファイルは、メモ帳などの任意のテキスト エディタを使って作成できます。作成したスクリプト ファイルは、ダイヤルアップ スクリプト ツールで、ダイヤルアップ ネットワークの接続に割り当てて使います。

2.0 スクリプトの基本構造

コマンドとは、スクリプト ファイルに記述する基本的な命令のことです。コマンドによっては、動作を細かく指定するためにパラメータが必要なものもあります。式は、演算子とパラメータを組み合わせたもので、その結果は 1 つの値になります。式はコマンドの中で値として使用できます。算術式、比較式、文字列の連結などはいずれも式です。

次に示すのは、ダイヤルアップ ネットワーク スクリプトの基本的な構造です。

```
;
; セミコロン (;) から行末まではコメントとして扱われます。
;

proc main
        ;  スクリプトでは、任意の数の変数とコマンドを使用できます。

        変数の宣言
```

```
        コマンド ブロック

    endproc
```

スクリプトには、**proc** キーワードによって指定された 1 つのメイン プロシージャが必要です。対応する **endproc** キーワードは、プロシージャの終わりを示します。

変数は、コマンドを記述する前に宣言する必要があります。メイン プロシージャのコマンドは、プロシージャに記述された順序に従って、先頭から順に実行されます。メイン プロシージャが終わると、スクリプトは終了します。

3.0 変数

スクリプトでは変数を使用できます。変数名に使う文字には、大文字または小文字の英字、数字、およびアンダースコア (_) を任意に組み合わせることができます。ただし、先頭の文字は英字またはアンダースコアである必要があります。また、予約語は変数名として使用できません。予約語の詳細については、このドキュメントの最後の一覧を参照してください。

変数は、使用する前に宣言する必要があります。変数を宣言するときは、変数の型も定義する必要があります。ある型の変数には、同じ型の値だけを格納できます。変数の型には、次の 3 種類があります。

型	説明
integer	7、-12、5698 などの正の数または負の数です。
string	"Hello world!"、"Enter password:" など、二重引用符で囲まれた文字列です。
boolean	ブール型 (二者択一) の値で、TRUE または FALSE のどちらかの値をとります。

変数には、次のような代入ステートメントを使って値を代入します。

    変数 = 式

変数には、式の評価結果が代入されます。

例:

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

boolean bDone = FALSE

string szIP = (getip 2)

set ipaddr szIP
```

3.1 システム変数

システム変数は、スクリプト コマンドによって値が設定されるか、またはダイヤルアップ ネットワークの接続をセットアップしたときに入力した情報によって値が決まります。システム変

数は読み取り専用で、スクリプト内で値を変えることはできません。システム変数には、次のものがあります。

名前	型	説明
$USERID	string	現在の接続で使われるユーザー名です。設定される値は、ダイヤルアップ ネットワークの [接続] ダイアログ ボックスで指定したユーザー名です。
$PASSWORD	string	現在の接続で使われるパスワードです。設定される値は、ダイヤルアップ ネットワークの [接続] ダイアログ ボックスで指定したパスワードです。
$SUCCESS	boolean	この変数は一部のコマンドによって設定され、コマンドが成功したかどうかを示します。この変数を使うと、値に応じてスクリプトの動作を変更できます。
$FAILURE	boolean	この変数は一部のコマンドによって設定され、コマンドが失敗したかどうかを示します。この変数を使うと、値に応じてスクリプトの動作を変更できます。

システム変数は、同じ型の式が使われるところでは常に使用できます。たとえば、次のようなコマンドを記述できます。

```
transmit $USERID
```

$USERID は string 型なので、このコマンドは有効なコマンドです。

4.0 リテラル文字列

ダイヤルアップ ネットワーク スクリプトでは、次のようなエスケープ シーケンスとキャレット変換を使用できます。

リテラル文字列	説明
^*char*	キャレット変換 *char* が '@' から '_' までの値である場合、この文字列は 0 ～ 31 の 1 バイトの値に変換されます。たとえば、^M は 1 つのキャリッジ リターンに変換されます。 *char* が 'a' から 'z' までの値である場合、この文字列は 1 ～ 26 の 1 バイトの値に変換されます。 *char* が上に示した値以外の値である場合、この文字列に対してキャレット変換は行われません。
<cr>	キャリッジ リターン
<lf>	ライン フィード
\"	二重引用符
\^	キャレット
\<	山形かっこ (<)
****	円記号

例:

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

5.0 式

式は、演算子とパラメータを組み合わせたもので、その評価結果は 1 つの値になります。式はコマンドの中で値として使うことができます。

式では、任意の変数、および integer 型、string 型、または boolean 型の値と、次の表に示す単項演算子および 2 項演算子とを組み合わせて使用できます。単項演算子は最も優先順位が高く、2 項演算子は表の上にあるものほど優先順位が高くなります。

次の表は、単項演算子の一覧です。

演算子	演算の種類
-	符号の反転
!	1 の補数

次の表は、2 項演算子の一覧です。表の上にあるものほど優先順位が高くなります。また、同じ行の演算子では、左にあるものほど優先順位が高くなります。

演算子	演算の種類	型の制限
* /	乗算または除算	integer 型
+ -	加算または減算	integer 型、string 型 (+ 演算子のみ)
< > <= >=	比較	integer 型
== !=	等しい、または等しくない	integer 型、string 型、boolean 型
and	論理 AND	boolean 型
or	論理 OR	boolean 型

例:

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

6.0 コメント

セミコロンから行末までのすべてのテキストはコメントとして扱われ、スクリプト実行時には無視されます。

例:

```
; これはコメントです。

transmit "hello"          ; 文字列 "hello" を送信します。
```

7.0 キーワード

キーワードは、スクリプトの構造を指定するために使います。コマンドとは異なり、キーワードは何らかのアクションを実行するわけではありません。次に示すのは、キーワードの一覧です。

proc 名前

プロシージャの始まりを示します。スクリプトには、必ずメイン プロシージャ (**proc** main) が 1 つ必要です。スクリプトの実行は、メイン プロシージャから始まり、メイン プロシージャの終わりで終了します。

endproc

プロシージャの終わりを示します。スクリプトがメイン プロシージャの **endproc** ステートメントまで実行されると、ダイヤルアップ ネットワークで PPP または SLIP が起動されます。

integer 名前 [**=** 値]

integer 型の変数を宣言します。変数は、任意の数式または変数で初期化できます。

string 名前 [**=** 値]

string 型の変数を宣言します。変数は、任意のリテラル文字列または変数で初期化できます。

boolean 名前 [**=** 値]

boolean 型の変数を宣言します。変数は、任意の boolean 型の式または変数で初期化できます。

8.0 コマンド

すべてのコマンドは予約語です。したがって、コマンドと同じ名前の変数は宣言できません。次に示すのは、コマンドの一覧です。

delay *nSeconds*

nSeconds に指定された秒数だけ待ってから次のコマンドを実行します。

例:

```
delay 2       ; 2 秒間待機します。
delay x * 3   ; x * 3 秒間待機します。
```

getip *value*

リモート コンピュータから IP アドレスを受け取ります。接続先のインターネット アクセス プロバイダが 1 つの文字列内で複数の IP アドレスを返す場合は、*value* を指定すると、何番目の IP アドレスを使うかを指定できます。

例:

```
; 2 番目の IP アドレスを使います。
set ipaddr getip 2

; 最初の IP アドレスを変数に代入します。
szAddress = getip
```

goto *label*

スクリプトの中の *label* で指定された行に分岐し、*label* に続くコマンドが順次実行されます。

例:

```
  waitfor "Prompt>" until 10
  if !$SUCCESS then
       goto BailOut ; BailOut に分岐し、BailOut に続くコマンドを
                    ; 実行します。
  endif

  transmit "bbs^M"
  goto End

BailOut:
  transmit "^M"
```

halt

スクリプトの実行を停止します。このコマンドを指定しても、ターミナル ウィンドウは閉じません。接続を確立するには、[続行] をクリックする必要があります。スクリプトは再実行できません。

if *condition* **then**
commands
endif

condition が TRUE の場合に、*commands* を実行します。*commands* には、複数のコマンドを指定できます。

例:

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

label **:**

スクリプトの分岐先を示します。*label* には、変数の名前付け規則に従って固有の名前を指定する必要があります。

set port databits 5 | 6 | 7 | 8

セッション中に送受信するバイトのデータビットを変えます。データビットは、5 ～ 8 の値をとることができます。スクリプトにこのコマンドが指定されていない場合は、ダイヤルアップ ネットワークの接続で指定されたプロパティの設定が使われます。

例:

```
set port databits 7
```

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

セッション中のポートのパリティの設定を変えます。スクリプトにこのコマンドが指定されていない場合、ダイヤルアップ ネットワークの接続で指定されたプロパティの設定が使われます。

例:

```
set port parity even
```

set port stopbits 1 | 2

セッション中のポートのストップビットの設定を変えます。スクリプトにこのコマンドが指定されていない場合、ダイヤルアップ ネットワークの接続で指定されたプロパティの設定が使われます。

例:

```
set port stopbits 2
```

set screen keyboard on | off

スクリプト実行中のターミナル ウィンドウで、キーボードからの入力を有効または無効にします。

例:

```
set screen keyboard on
```

set ipaddr *string*

ワークステーションの IP アドレスを指定します。*string* には、IP アドレスの形式で値を指定する必要があります。

例:

```
szIPAddress = "11.543.23.13"
set ipaddr szIPAddress

set ipaddr "11.543.23.13"

set ipaddr getip
```

transmit *string* [**, raw**]

string に指定された文字列をリモート コンピュータに送信します。

raw パラメータを指定した場合を除いて、エスケープ シーケンスとキャレット変換が処理された後の文字列がリモート コンピュータに送信されます。キャレット変換またはエスケープ シーケンスとして解釈される文字列がユーザー名やパスワードに含まれている場合は、**raw** パラメータを指定して $USERID や $PASSWORD システム変数を送信します。

例:

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

waitfor *string* [**, matchcase**] [**then** *label*
{ **,** *string* [**, matchcase**] **then** *label* }]
[**until** *time*]

指定された 1 つまたは複数の文字列をリモート コンピュータから受け取るまで待機します。**matchcase** パラメータを指定した場合を除き、*string* に指定された文字列の大文字と小文字は区別されません。

一致する文字列を受け取った場合、 **then** *label* パラメータが指定されているときは、スクリプト ファイルの中の *label* で指定された行に分岐します。

until *time* パラメータを指定すると、一致する文字列を受け取るために *time* に指定された秒数だけ待機した後、次のコマンドを実行します。**until** *time* パラメータは省略できます。**until** *time* パラメータを指定しないと、一致する文字列を受け取るまで待機し続けます。

指定された文字列のいずれかを受け取ると、システム変数 $SUCCESS には TRUE が設定されます。文字列を受け取らないまま *time* に指定された秒数が経過すると、システム変数 $SUCCESS には FALSE が設定されます。

例:

```
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 *condition* **do**
commands
endwhile

condition が FALSE になるまで、*commands* を実行します。*commands* には、複数のコマンドを指定できます。

例:

```
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 予約語

以下の語は予約語です。変数名として使用することはできません。

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	