Langage des commandes d'un script d'accès au réseau pour le support de script d'accès au réseau

Table des matières

1.0 Présentation

De nombreux fournisseurs de services Internet et de services en ligne exigent que certaines informations soient entrées manuellement, telles que votre nom d'utilisateur et votre mot de passe, afin d'établir une connexion. A l'aide du support de script pour l'accès réseau à distance, vous pouvez écrire un fichier script pour automatiser ce processus.

Un script est un fichier-texte qui contient une série de commandes, de paramètres et d'expressions requises par votre fournisseur de services Internet ou un service en ligne pour établir la connexion et utiliser le service. Un fichier script peut être créé à l'aide de n'importe quel éditeur de texte tel que Bloc-notes de Microsoft. Une fois créé, vous pouvez l'affecter à une connexion spécifique de l'accès réseau à distance en exécutant l'outil de création du script d'accès au réseau.

2.0 Structure de base d'un script

Une commande est l'instruction élémentaire contenue dans un fichier script. Certaines commandes requièrent des paramètres qui définissent avec précision la fonction d'une commande. Une expression est une combinaison d'opérateurs et d'arguments qui produisent un résultat. Il est possible d'utiliser des expressions en tant que valeurs dans les commandes. Citons à titre d'exemple les comparaisons arithmétiques, relationnelles et les concaténations de chaînes.

La présentation de base d'un script d'accès réseau à distance est la suivante :

```
;
; Un commentaire commence par un point virgule et se termine à
; la fin de la ligne.
;

proc main
      ;  Un script peut inclure n'importe quel nombre de variables
      ;  et de commandes
```

```
        déclarations de variables

        bloc de commandes

endproc
```

Un script doit comporter une procédure principale MAIN, spécifiée par le mot clé **proc** et le mot clé **endproc** correspondant pour marquer la fin de la procédure.

Vous devez déclarer des variables avant d'ajouter des commandes. La première commande dans la procédure principale est exécutée, et toutes les commandes suivantes sont exécutées dans l'ordre où elles apparaissent dans le script. Le script se termine à la fin de la procédure principale.

3.0 Variables

Les scripts peuvent contenir des variables. Les noms de variables doivent commencer par une lettre ou par un souligné ('_') ; ils peuvent inclure toute séquence de lettres majuscules ou minuscules, chiffres et soulignés. Un mot réservé ne peut pas être pris comme nom de variable. Pour plus d'informations, consultez la liste des mots réservés à la fin de ce document.

Les variables doivent être déclarées avant d'être utilisées. Lorsque vous déclarez une variable, vous devez également définir son type. Une variable d'un type spécifique ne peut contenir que des valeurs de ce même type. Les trois types de variables suivants sont admis :

Type	Description
entier (integer)	nombre positif ou négatif, tel que 7, -12, ou 5698.
chaîne (string)	série de caractères mis entre guillemets. A titre d'exemple : "Bonjour la terre !" ou "Entrez votre mot de passe :".
booléen (boolean)	valeurs booléennes logiques de TRUE ou de FALSE.

Les variables sont des valeurs affectées à l'aide de l'instruction d'attribution suivante :

variable **=** *expression*

La variable trouve l'expression évaluée.

Exemples :

```
integer compteur = 5
integer délai = (4 * 3)
integer i

boolean fin = FALSE

string prendreIP = (getip 2)

set ipaddr prendreIP
```

3.1 Variables système

Les variables système sont définies à l'aide des commande de création de script ou elles sont déterminées par les informations que vous entrez lorsque vous configurez une connexion
d'accès au réseau. Les variables système sont accessibles en lecture seule uniquement, ce qui signifie qu'elles ne peuvent pas être changées à l'intérieur du script. Les variables système comprennent :

Nom	Type	Description
$USERID	Chaîne	Identification de l'utilisateur pour la connexion courante. Cette variable représente le nom de l'utilisateur qui est mentionné dans la boite de dialogue Accès réseau à distance - Connexion à.
$PASSWORD	Chaîne	Mot de passe pour la connexion courante. Cette variable représente la valeur du nom de l'utilisateur mentionné dans la boîte de dialogue Accès réseau à distance - Connexion à.
$SUCCESS	Booléen	Cette variable est définie par certaines commandes pour indiquer si la commande a pu ou non être exécutée normalement. Le script se base sur la valeur de cette variable pour prendre une décision.
$FAILURE	Booléen	Cette variable est définie par certaines commandes pour indiquer si la commande a échoué ou non. Le script se base sur la valeur de cette variable pour prendre des décisions.

Ces variables peuvent être utilisées pour chaque expression d'un même type. A titre d'exemple :

```
transmit $USERID
```

est une commande valable car $USERID est une variable de type "chaîne".

4.0 Constantes de chaîne

La création du script d'accès au réseau prend en charge les séquences d'échappement et les conversions des carets, tels que décrits ci-dessous.

Constante de chaîne	Description
^*char*	Conversion de caret Si *char* est une valeur entre '@' et '_', la séquence de caractère est convertie en une valeur d'un seul octet comprise entre 0 et 31. A titre d'exemple, ^M correspond à un retour-chariot. Si *char* est une valeur entre a et z, la séquence de caractère est convertie en une valeur d'un seul octet comprise entre 1 et 26. Si *char* est toute autre valeur, la séquence de caractères ne subit aucune conversion particulière.
<cr>	Retour chariot
<lf>	Saut de ligne
\"	Guillemet double
\^	Caret simple
\<	'<' simple
****	Barre oblique à gauche

Exemples :

```
transmit "^M"
```

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

5.0 Expressions

Une expression est une combinaison d'opérateurs et d'arguments qui produit un résultat. Les expressions peuvent être utilisées en tant que valeurs dans toute commande.

Une expression peut être n'importe quelle combinaison de variables, entiers, chaînes ou valeurs booléennes avec l'un des opérateurs unaires et binaires mentionnés les tableaux suivants. Tous les opérateurs unaires ont le niveau de priorité le plus élevé. La préséance des opérateurs binaires est indiquée d'après leur position dans le tableau.

Les opérateurs unaires sont les suivants :

Opérateur	Type d'opération
-	Soustraction unaire
!	Complément à un

Les opérateurs binaires sont recensés dans le tableau ci-dessous selon leur ordre de préséance. Les opérateurs bénéficiant d'un niveau de priorité supérieur sont recensés en premier :

Opérateurs	Type d'opération	Restrictions de types
* /	Multiplication	Entiers
+ -	Addition entiers chaînes (+ uniquement)	
< > <= >=	Relationnelle	Entiers
== !=	Egalité	Entiers, chaînes, booléens
AND	ET logique	Booléens
OR	OU logique	Booléens

Exemples :

```
compteur = 3 + 5 * 40
transmit "Bonjour" + " à tous"
delay 24 / (7 - 1)
```

6.0 Commentaires

Tout texte derrière un point virgule dans une ligne n'est pas pris en compte.

Exemples :

```
; ceci est un commentaire

transmit "bonjour"        ; transmettre la chaîne "bonjour"
```

7.0 Mots clé

Les mots clé caractérisent la structure du script. A la différence des commandes, ils n'exécutent aucune action. Les mots clé sont recensés dans la liste ci-après.

proc *nom*

Marque le début d'une procédure. Chaque script doit avoir une procédure principale (**proc** main). L'exécution du script commence au début de la procédure MAIN et se termine à la fin de celle-ci.

endproc

Marque la fin d'une procédure. Lorsque le script est exécuté jusqu'à l'instruction **endproc** de la procédure MAIN, l'accès réseau à distance démarre PPP ou SLIP.

integer *nom* [**=** *valeur*]

Déclare une variable de type entier. Vous pouvez utiliser toute expression ou variable numérique pour initialiser la variable.

string *nom* [**=** *valeur*]

Déclare une variable de type chaîne. Vous pouvez utiliser toute constante ou variable de chaîne pour initialiser la variable.

boolean *nom* [**=** *valeur*]

Déclare une variable de type booléen. Vous pouvez utiliser toute expression ou variable booléenne pour initialiser la variable.

8.0 Commandes

Toutes les commandes sont des mots réservés ; de ce fait, vous ne pouvez pas déclarer des variables qui comportent les mêmes noms que les commandes. Les commandes sont recensées ci-après :

delay *nSecondes*

S'arrête pendant le nombre de secondes spécifiées dans *nSecondes* avant d'exécuter la commande suivante dans le script.

Exemples :

```
delay 2      ; pause pendant 2 secondes
delay x * 3  ; pause pendant x * 3 secondes
```

getip *valeur*

Attend de recevoir une adresse IP à partir de l'ordinateur distant. Si votre fournisseur de services Internet renvoie plusieurs adresses IP dans une chaîne, utilisez le paramètre *valeur* pour préciser l'adresse IP que doit utiliser le script.

Exemples :

```
; obtenir la seconde adresse IP
set ipaddr getip 2

; affecter la première adresse IP reçue à la variable
PrendreAdrIP = getip
```

goto *label*

Va directement à l'emplacement désigné par *label* dans le script et continue l'exécution des commandes qui suivent.

Exemple :

```
  waitfor "Prompt>" until 10
  if !$SUCCESS then
       goto Sortie  ; aller directement vers la Sortie et
                    ; exécuter les commandes qui suivent
  endif

  transmit "bbs^M"
  goto Fin

Sortie :
  transmit "^M"
```

halt

Arrête le script. Cette commande ne supprime pas la fenêtre de dialogue du terminal. Vous devez cliquer sur Continuer pour établir la connexion. Vous ne pouvez pas redémarrer le script.

if *condition* **then**
commandes
endif

Exécute la série de *commandes* si la *condition* est vraie (TRUE).

Exemple :

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

label **:**

Indique l'endroit auquel accéder dans le script. Un label doit être un nom unique et respecter les conventions de noms des variables.

set port databits 5 | 6 | 7 | 8

Modifie le nombre de bits dans les octets qui sont transmis et reçus au cours de la session. Ce nombre peut être entre 5 et 8. Si vous n'incluez pas cette commande, l'accès réseau à distance utilise les paramètres de propriétés spécifiés pour la connexion.

Exemple :

```
set port databits 7
```

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

Modifie le mode de parité du port au cours de la session. Si vous n'incluez pas cette commande, l'accès réseau à distance utilise les paramètres de propriétés spécifiés pour la connexion.

Exemple :

```
set port parity even
```

set port stopbits 1 | 2

Modifie le nombre de bits d'arrêt pour le port au cours de la session. Ce nombre peut être 1 ou 2. Si vous n'incluez pas cette commande, l'accès réseau à distance utilise les paramètres de propriétés spécifiés pour la connexion.

Exemple :

```
set port stopbits 2
```

set screen keyboard on | off

Active ou désactive l'entrée au clavier dans la fenêtre du terminal de création de script.

Exemple :

```
set screen keyboard on
```

set ipaddr *chaîne*

Indique l'adresse IP du poste de travail pour cette session. *Chaîne* doit être au format d'une adresse IP.

Exemples :

```
PrendreAdrIP = "11.543.23.13"
set ipaddr PrendreAdrIP

set ipaddr "11.543.23.13"

set ipaddr getip
```

transmit *chaîne* [**, raw**]

Transmet les caractères spécifiés dans la *chaîne* vers l'ordinateur distant.

L'ordinateur distant reconnaît les séquences d'échappement et les conversions de carets, sauf si le paramètre **raw** est inclus dans la commande. Ce paramètre permet de transmettre les variables système $USERID et $PASSWORD lorsque le nom ou le mot de passe de l'utilisateur contient des séquences de caractères qui seraient interprétées comme des séquences d'échappement ou des carets si le paramètre **raw** n'était pas précisé.

Exemples :

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

waitfor *chaîne* [**, matchcase**] [**then** *label*
{ **,** *chaîne* [**, matchcase**] **then** *label* }]
[**until** *délai*]

Attend que l'ordinateur reçoive la/les chaîne(s) spécifiées à partir de l'ordinateur distant. Le paramètre *chaîne* distingue les majuscules des minuscules, sauf si le paramètre **matchcase** est précisé.

Si une chaîne correspondante est reçue et si le paramètre **then** *label* est utilisé, cette commande va directement à l'endroit désigné par *label* dans le fichier script.

Le paramètre facultatif **until** *délai* définit le nombre de secondes maximum pendant lesquelles votre ordinateur doit attendre de recevoir la chaîne avant d'exécuter la commande suivante. Si ce paramètre n'est pas précisé, votre ordinateur attend indéfiniment.

Si votre ordinateur reçoit l'une des chaînes spécifiées, la variable système $SUCCESS est définie à TRUE (vrai). Sinon, elle est définie à FALSE (faux) si le nombre de secondes indiqué dans *délai* s'écoule avant que la chaîne ne soit reçue.

Exemples :

```
waitfor "Accès :"

waitfor "Mot de passe ?", matchcase

waitfor "invite>" until 10

waitfor
       "Accès :"    then ExécuterAccès,
       "Mot de passe :"    then ExécuterMotDePpasse,
       "BBS :"             then ExécuterBBS,
       "Autre :"    then ExécuterAutre
  until 10
```

while *condition* **do**
commandes
endwhile

Exécute la série de *commandes* jusqu'à ce que la *condition* soit fausse (FALSE).

Exemple :

```
integer compteur = 0

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

9.0 Mots réservés

Les mots suivants sont réservés et ne peuvent être utilisés en tant que noms de variables.

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	