MADLib

Table des matières

ND-Faciliti MADLib

ND-Faciliti MADLib

MADLib est une bibliothèque de fonctions, pour PureBasic.
La librairie est une extension de la STL de Purebasic.
Elle comprend un résident avec structures et constantes, en plus du binaire des fonctions.

Toutes les fonctions, stuctures, constantes sont préfixées avec madlib_

La compiliation est effectué en multiLib (Thread,Unicode,threadSafe).

TailBite à été utilisé pour compiler la librairie. (http://www.tailbite.com/).

Cette UserLib est fournis par package avec l'enssemble.

L'enssemble en rêgle général est tournée vers une instrumentation sous windows.
Néamoins, il tout à fait possible de l'utiliser dans les autres environements. Se referer la plage des compatibilité des fonctions.

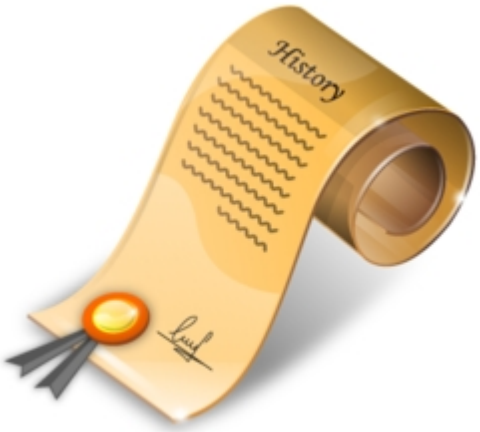
Version : 5
Révision : 10

Version compatible avec Purebasic : **5.x x86/x64**

 La version 4.X de Purebasic **n'est plus supportée**

Historique des versions

Historique des versions



Version actuelle :

- **5.10 :**
  - ListDirectory
  - ListFile
  - ExpandString
  - HotKeyToNum -> ShortKeyToNum
  - LoadLanguageFilePreBuilt
  - LoadLanguageFileWord

Historique :

1.0 :
- Journal

1.2 :
- ProgDir
- ProgName
- FileExists
- Const

1.3 :
- StringSplit

2.0 :
- Journal => Journal_old
- Journal
- chargeParametreCMD

2.2 :
- IniReadSectionNames
- IniReadSection
- Ping
- IPNum
- HostnameToIP

2.3 :
- chargeParametreCMD

2.4 :
- StrDate
- StrTime

2.5 :
- ConsoleDebug

3.0 :
- Suppression des éléments objets se trouvant dans cette librairie nécéssitant "oop"

3.1 :
- ChargeINI
- Structure ini

3.2 :
- chargeVariablesEnvironement

3.3 :
- codeNDF

3.4 :
- PingWin

3.7 :
- RandomChr
- StringCrypt
- StringDecrypt

3.8 :
- PingWin => PingWin_old
- PingWin
- Journal

3.11 :
- AjouteObjetFenetre
- IDObjetFenetre
- CommentairesObjetFenetre

3.12 :
- PingWin
- Journal

3.13 :
- ChargeINI

3.14 :
- Journal

3.15 :
- StringCrypt
- StringDecrypt

3.16 :
- CreateRecursiveDirectory

3.17 :
- Journal

3.18 :
- GFVI_GetElementName
- GFVI_GetInfo

3.19 :
- StringCrypt
- StringDecrypt

3.20 :
- RandomChr

3.21 :
- Console

3.22 :
- DetecteLangueWindows

3.24 :
- ListeFichiersDansRepertoire

3.27 :
- CodeNDF

3.28 :
- CreateDirectoryMulti
- VerifFichierChiffre
- ChiffreFichier
- DechiffreFichier

3.29 :
- ChiffreFichier
- DechiffreFichier

3.30 :
- Journal
- DeleteFileMulti
- Macro Const

3.31 :
- Journal

4.0 :
- Passage en UserLib (Compilation effectuée en PB 4.51).
- ChechMadLib
- PingWin
- GFVI_GetElementName
- GFVI_GetInfo

4.1 :
- Journal
- IniReadSectionNames

4.2 :
- ApplicationStart
- InitServiceStarting
- ServeurIP

- RecupereDonneesServeurIP
- AttenteCreationServeurIP
- InstallService
- DeleteService
- StartService
- ControlService
- StopService
- PauseService
- ContinueService

5.0 :

- Modification de la compilation pour Purebasic 4.6x
- RegParsePath
- RegConnect
- RegOpen
- RegCreate
- RegWriteMultiSZ
- RegReadMultiSZ
- RegWrite
- RegRead
- RegExist

5.1 :

- GetMADLibVersionDate
- RegEnumValue
- RegEnumKey
- RegDeleteValue
- RegDeleteTreeKeyValue
- CodeNDF

5.2 :

- HTMLEncodeText
- HTMLDecodeText
- DateTP
- WinGetFreeSpace
- DownloadFile
- DownloadFileList
- ExpandString (transformation codendf)
- InvertValueLong
- RequestIPServer
- ChargeVariablesEnvironement
- RegEnumValue
- RegEnumKey
- RegDeleteValue
- RegDeleteTreeKeyValue
- RegCreate
- RegWriteMultiSZ
- RegReadMultiSZ

- RegWrite
- RegRead
- RegExist

5.3 :

- NetJoinDomainWin
- GetIPByNameWin
- TextErrorWinWSA
- WrapDirectorySlash
- CheckCodeInCodei
- GetComputerName
- GetComputerGUID
- IsWin64
- LoadXML
- ParseXMLValue
- FindNodeXML
- TextErrorXML
- ReverseReturnCodei
- TextErrorWinIPStatus
- PingWin
- ApplicationStart
- ServerIP
- ExpandString

5.3.1 :

- DownloadFile
- DownloadFileList

5.4 :

- ListFile
- ListDirectory
- ListeFichierDansRepertoire
- InjectionDLL
- IsWin64 => IsWin64b
- RegExist
- ConfigDiskFreeSpace
- LoadCommandLineParameters
- ChargeParametreCMD
- Journal_old
- LogFile

5.5 :

- CreateFieldGrown
- CreatePlotGrown
- GetPlotFrowList
- SearchSeedByName
- DeletePlotGrown

- SizePlotGrownList
- GetTypeePlotGrown
- ResetPlot
- DeleteSeed
- AddSeedString
- AddSeedInt
- AddSeedFloat
- GetSeedString
- GetSeedInt
- ExistSeed
- AddSeedDouble
- GetSeedDouble
- GetAllSeedString
- GetAllSeedInt
- GetAllSeedFloat
- GetAllSeedDouble
- GetAllSeedQuad
- RandomChr
- LoadXML
- ParseXMLValue
- FindNodeXML
- TextErrorXML
- LoadFileXML
- GetXMLBaseNodeList
- GetXMLBaseNodeAttributes
- GetXMLBaseNodeID
- GetXMLBaseNodeName
- GetXMLBaseNodeInfos
- AddNodeAttribute
- AddNodeInfo
- DelNodeEntry
- DelNodeInfos
- DelNodeAttributes
- TextErrorXML
- CountXMLBaseSubNodes
- GetXMLBaseParentID
- ConfigExpandString
- ExpandString
- IsWin64

5.5.1 :

- RandomChr : Réinsertion de la fonction (erreur a la compilation).

5.6 :

- IAmUnicode

- ListFile
- ListDirectory
- GetDirectorySize
- ProtectProcessWin
- SplachScreen
- ChangeProcessPrivilege
- CenterWindow

5.7 :

- RegReadMultiSZ
- RegWriteMultiSZ

5.8 :

- DeleteFileMultiSZ
- RegWrite
- ListADStreamData
- ReadStringADStream
- AddStringADStream
- ReadADStream
- AddADStream
- DeleteADStream

5.9 :

- DeleteFileMulti
- RegWrite
- GetIPByNameWin
- Ini (structure)
- ListADStreamData
- ReadStringADStream
- AddStringADStream
- ReadADStream
- AddADStream
- DeleteADStream
- CompareVersion
- DirectoryCompress
- ConvertHexRGB
- HotKeyToNum
- WindowNotify
- CreateDirectoryMulti
- GetDirectorySize

Légende de notation

Légende de notation

X.Y : X est une version majeur en théorie non compatible avec les anciennes version de madlib.

- Version de la librairie
  - Ajout d'une fonction
  - Modification d'une fonction
  - Suppression d'une fonction

Fonctions

Description des fonctions

Cette partie décrit chaque fonction, comment utiliser les ressources.
Elle est divisé en plusieurs catégories pour mieux les repérer.
En général il existe un exemple. Les structures associées seront aussi expliquées.

Toute les fonctions sont préfixées de madlib_ afin d'éviter tout conflit avec d'autre "userlib".

 Il est important de la note relative au passage de paramètre disponible ICI

Standard

Fonctions standards

madlib_ConsoleDebug

madlib_ConsoleDebug()

Syntaxe :

```
Resultat$ = madlib_ConsoleDebug(Texte$="",NiveauEcriture$="",CouleurTexte=7,CouleurFond=0)
```

Description :

Ouvre une console

Le format retournée sera du style (00:00:00).

Exemple :

```
Debug madlib_ConsoleDebug("Coucou")
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_StrTime

madlib_StrTime()

Syntaxe :

Resultat$ = madlib_StrTime()

Description :

Retourne une heure en format texte.

Le format retournée sera du style (00:00:00).

Exemple :

```
Debug madlib_StrTime()
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_StrDate

madlib_StrDate()

Syntaxe :

Resultat$ = madlib_StrDate()

Description :

Retourne une date en format texte.

Le format retournée sera du style (jj/mm/aaaa).

Exemple :

```
Debug madlib_StrDate()
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_IniReadSection

madlib_IniReadSection()

Syntaxe :

Resultat = madlib_IniReadSection(CheminNomFichier$,nomSection$,Map

TAB_listeContenuSection.s())

Description :

Retourne le contenu d'une section d'un fichier INI.

La valeur du résultat contient le nombre de valeurs récupérées.

Exemple :

```
NewMap MonDictionnaire.s(1)

Debug madlib_IniReadSectionNames("c:
\MonFichier.ini","MaSection",MonDictionnaire())
```

OS Supportés (testé) :

madlib_IniReadSectionNames()

Syntaxe :

Resultat = madlib_IniReadSectionNames(CheminNomFichier$,Array TAB_SectionNames.s(1))

Description :

Retourne les noms de sections contenu dans un fichier INI.

La valeur du resultat contient le nombre de sections récupérés.

Exemple :

```
Dim MonTableau.s(1)

Debug madlib_IniReadSectionNames("c:
\MonFichier.ini",MonTableau())
```

OS Supportés (testé) :

madlib_LoadCommandLineParameters

madlib_LoadCommandLineParameters()

Syntaxe :

Resultat = madlib_LoadCommandLineParameters(Map HASH_CMDLINE.s())

Description :

Charge dans un dictionnaire les paramètres passer en ligne de commande.

Le nombre de paramètres sera placer dans le retour de la fonction.

Les commutateurs sont : / ou - ou +.

Les valeurs sans commutateurs seront interprêté avec l'éttiquette : *no_switch_x

Exemple :

```
NewMap LigneDeCommande.s()

Debug madlib_ChargeParametreCMD(LigneDeCommande())
```

OS Supportés (testé) :

madlib_Journal

madlib_Journal()

Syntaxe :

Resultat = madlib_Journal(*InfosJournal.madlib_ConfigJournal,Texte$,NiveauEcriture$="")

Description :

Journalise des événements données. Un réglage des paramètres du fichier de log peut être effectuée grace à la structure ConfigJournal.

Il est possible aussi de pouvoir utiliser plusieurs fois cette fonction avec plusieurs fichier journal.

Il existe une notion de niveau avec plusieurs sigles.

Description des structures associées :

Exemple :

```
Protected log.madlib_ConfigJournal

log\CheminFichier$ = "%windir%\temp\EssaiMadlibJournal.log

Journal(@log,"coucou","NIV1")
Journal(@log,"coucou","NIV+1")
```

OS Supportés (testé) :

madlib_StringSplit

madlib_StringSplit()

Syntaxe :

Resultat = madlib_StringSplit(szInput$,splitter$,Array A.s(1))

Description :

Sépare une chaîne de caractère en tableau. Le résultat sera le nombre de ligne dans le tableau

La tableau devra comporter une seule dimension.

Exemple :

```
Protected Dim tab.i
```

```
NombreDeLignes = madlib_StringSplit("Coucou|Voici|ma|
ligne","|",tab())

Debug NombreDeLignes
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_ProgName

madlib_ProgName()

Syntaxe :

Resultat$ = madlib_ProgName()

Description :

Récupere le nom courant de l'application.

Exemple :

```
Debug madlib_ProgName()
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_ProgDir

madlib_ProgDir()

Syntaxe :

Resultat$ = madlib_ProgDir()

Description :

Récupere le chemin courant de l'application.

Exemple :

```
Debug madlib_ProgDir()
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_RandomChr

madlib_RandomChr()

Syntaxe :

Resultat$ = madlib_RandomChr(AsciiMin.a=97,AsciiMax.a=122,AsciiExclusion$="",Length.i=1)

Description :

Retourne un caractère aléatoirement compris entre deux valeurs Ascii.

Il est possible aussi indiquer les caractères que l'on veut exclure lors de la génération aléatoire et la longueur de la chaine aléatoire souhaité.

Option :

AsciiMin.a => Valeur ASCII minimum generant les caractères.

AsciiMax.a => Valeur ASCII maximum generant les caractères.

AsciiExclusion$ => Chaine d'exclusion. Il est possible de spécifier des caractères qui ne seront écarté de la génération.

Length.i => Nombre de caractères à generer. (Par défaut 1 caractère).

Exemple :

```
Debug madlib_RandomChr(0,255,"abcdef")
```

OS Supportés (testé) :

madlib_CreateDirectoryMulti

madlib_CreateDirectoryMulti()

Syntaxe :

Resultat = madlib_CreateDirectoryMulti(Emplacement$)

Description :

Exécute l'opération de création de répertoire et de sous répertoire automatiquement.

Exemple :

```
Debug madlib_CreateDirectoryMulti("d:\monPremierRepertoire
\MonDeuxiemeRepertoire\etc")
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_ListDirectory()

Syntaxe :

```
Resultat$ = madlib_ListDirectory(Directory$,FilterDirectoryName$="",Recursive.b=#False,List
LST_DirectoryResult.s())
```

Description :

Remplis la liste chaînée du chemin ainsi que les nom de fichiers contenu dans le répertoire ou les sous-répertoires.

Il est possible de placer un filtre pour les fichiers.

En mode récursif il parcours aussi les sous-répertoires.

Exemple :

```
NewList LST_Directory.s()

madlib_ListDirectory("c:
\windows","",#True,LST_Directory())

Foreach LST_Directory()

    Debug LST_Directory()

Next
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_DeleteFileMulti

madlib_DeleteFileMulti()

Syntaxe :

Resultat = madlib_DeleteFileMulti(FileLocation$,RecursiveDirectory.b=#False)

Description :

Supprime plusieurs fichier en même temps.

Il est possible de spécifier plusieurs fichier dans chemin Fichier

L'option ExploreSousRepertoire, permet la récursivité de la fonction.

Le résultat sera le nombre de fichiers supprimées.

Exemple :

```
Debug madlib_DeleteFileMulti("c:\temp\*.tmp",#True)
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_FileExist

madlib_FileExist()

Syntaxe :

Resultat = madlib_FileExist(CheminFichier$)

Description :

Vérifie la présence d'un fichier.

Retourne la valeur "#True " si le fichier existe sinon #False

Exemple :

```
Debug madlib_FileExist("c:\windows\notepad.exe")
Debug madlib_FileExist("c:\windows\notep.exe")
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_CheckMadLib

madlib_CheckMadlib()

Syntaxe :

Resultat = madlib_CheckMadLib()

Description :

Le résultat sera booléen. Si la librairie n'est pas conforme il renvera ***#False***.

Exemple :

```
Debug madlib_CheckMadlib()
```

OS Supportés (testé) :

madlib_ApplicationStart

madlib_ApplicationStart()

Syntaxe :

Resultat$ = madlib_ApplicationStart(*Programme.madlib_ConfigProgram)

Description :

Gestion d'un lancement / suivi d'un programme.

Structure(s) associée(s) :

madlib_ConfigProgram

Constantes Associées :

#madlib_APPSTART_NO_FILE_PRESENT => Le fichier spécifier n'est pas présent.

#madlib_APPSTART_PB_WITH_START_PRG => Impossible de lancer le programme.

#madlib_APPSTART_TIMEOUT_PRG => Le programme dépasse le temp limit d'execution spécifié.

Exemple :

```
MonProgramme.madlib_ConfigProgram

MonProgramme\ProgramLocation$ = "%windir%\notepad.exe"
MonProgramme\ForceRunning = #True

madlib_ApplicationStart(@MonProgramme)
```

OS Supportés (testé) :

madlib_ChargeINI

madlib_ChargeINI()

Syntaxe :

Resultat = madlib_ChargeINI(CheminFichierINI$,Map DICT_INI.madlib_ini(),RechercheNomGroupe$="")

Structure(s) associée(s) :

madlib_ini

Description :

Ouvre le fichier ini spécifié et chargen dans le dictionnaire le fichier par section et valeur.

Exemple :

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_InvertValueLong

madlib_InvertValueLong()

Description :

Inverse une valeur de type "long"

Retourne la valeur inversée ou non suivant l'option

Syntaxe :

```
Resultat.l =
madlib_InvertValueLong(Value.l,NegativeTransform.i=#PB_Ignore)
```

Option :

NegativeTransform :

#True => la fonction sortira systématiquement une valeur négative quelques soit la valeur d'entrée.

#False => la fonction sortira systématiquement une valeur positive quelques soit la valeur d'entrée.

#PB_Ignore =>la fonction sortira l'inverse de la valeur systématiquement.

Exemple :

```
Debug madlib_InvertValueLong(-12)
Debug madlib_InvertValueLong(12)
Debug madlib_InvertValueLong(-12,#False)
Debug madlib_InvertValueLong(12,#True)
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_DateTP

madlib_DateTP()

Description :

Retourne la date en serie indiqué par le serveur de temp.


Le protocole utilisé est le TP et non le NTP. Ce protocole est défini dans la RFC 868.

http://tools.ietf.org/html/rfc868

Syntaxe :

Resultat.i = madlib_DateTP(ServerName$,GMT.b=0,WaitSignal.i=10,PortTP.i=37)

Option :

ServerName$ => Nom du serveur de temp

GMT.b => Inscrire la valeur de décalage en heure.

WaitSignal.i => Temp d'attente maximum du signale du serveur.

PortTP.i => Numéro de port sur la connexion au serveur. (Par défaut 37)

Exemple :

```
Debug FormatDate("%dd/%mm/%yyyy - %hh:%ii:%
ss",DateTP("ptbtime2.ptb.de"))
Debug FormatDate("%dd/%mm/%yyyy - %hh:%ii:%
ss",DateTP("utcnist2.colorado.edu",0,120))
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_HTMLDecodeText

madlib_HTMLDecodeText()

Description :

Décode le texte du format HTMLDecodeText

Syntaxe :

Resultat.s = madlib_HTMLDecodeText(Text$)

Option :

Text$ => Texte à encoder.

Exemple :

```
Debug madlib_HTMLDecodeText("Bonjour c'est super cool ! éé !")
```

OS Supportés (testé) :

madlib_HTMLEncodeText

madlib_HTMLEncodeText()

Description :

Encode le texte dans un format compatible HTML.

Syntaxe :

Resultat.s = madlib_HTMLEncodeText(Text$)

Option :

Text$ => Texte à encoder.

Exemple :

```
Debug madlib_HTMLEncodeText("Bonjour c'est super cool ! éé !")
```

OS Supportés (testé) :

madlib_SQLEncodeText

madlib_SQLEncodeText()

Description :

Encapsule le guillemet pour les requêtes SQL.

Syntaxe :

Resultat.b = madlib_SQLEncodeText(Text$)

Option :

Text$ => Texte à encoder.

Exemple :

```
Debug madlib_SQLEncodeText("Bonjour c'est super cool ! éé !")
```

OS Supportés (testé) :

madlib_SQLDecodeText

madlib_SQLDecodeText()

Description :

Decode le texte précédement encodé avec la fonction *SQLEncodeText*.

Syntaxe :

Resultat.b = madlib_SQLDecodeText(EncodeText$)

Option :

EncodeText$ => Texte à encoder.

Exemple :

```
Debug madlib_SQLDecodeText("Bonjour c'est super cool ! éé !")
```

OS Supportés (testé) :

madlib_DownloadFile()

Description :

Télécharge un fichier en HTTP.

Syntaxe :

Resultat.b = madlib_DownloadFile(*Config.madlib_ConfigDownloadFile)

Structure(s) associée(s) :

- madlib_ConfigDownloadFile

Exemple :

```
MonTelechargement.madlib_ConfigDownloadFile

MonTelechargement\URLLocation$ = "http://debian.netcologne.de/
debian-cd/6.0.3/amd64/iso-cd/debian-6.0.3-amd64-CD-"+Str(ind)
+".iso"

MonPanierDeTelechargement()\FileNameLocationReceive$ = "D:\temp
\downl\"

Debug madlib_DownloadFile()
```

OS Supportés (testé) :

madlib_DownloadFileList

madlib_DownloadFileList()

Description :

Télécharge une liste de fichiers.

Syntaxe :

```
Resultat.b = madlib_DownloadFileList(List
Config.madlib_ConfigDownloadFile(),NBSimultaneousDownloads.b=1)
```

Structure(s) associée(s) :

- madlib_ConfigDownloadFile

Options :

NBSimultaneousDownloads.b => Nombre de téléchargement simultané autorisé.

Exemple :

```
NewList MonPanierDeTelechargement.madlib_ConfigDownloadFile()

For ind = 1 To 23
   AddElement(MonPanierDeTelechargement())
   MonPanierDeTelechargement()\URLLocation$ = "http://
debian.netcologne.de/debian-cd/6.0.3/amd64/iso-cd/debian-6.0.3-
amd64-CD-"+Str(ind)+".iso"
   MonPanierDeTelechargement()\FileNameLocationReceive$ = "D:
\temp\downl\"
Next
```

```
ForEach MonPanierDeTelechargement()
   Debug MonPanierDeTelechargement()\URLLocation$
Next

madlib_DownloadFileList(MonPanierDeTelechargement(),2)
```

OS Supportés (testé) :

madlib_CheckCodeInCodei()

Description :

Vérifie une valeur encapsulé dans une autre.

Pratique pour les codes retours.

Nessecite d'avoir des pas minimum de valeur de 2.

Syntaxe :

Resultat.b = madlib_CheckCodeInCodei(Code.i,CodeToCheck.i)

Exemple :

```
Debug madlib_CheckCodeInCodei(12,4)
Debug madlib_CheckCodeInCodei(10,2)
```

OS Supportés (testé) :

madlib_WrapDirectorySlash

madlib_WrapDirectorySlash()

Description :

S'assure du bon formatage du chemin donné. Ajoute un "\" ou un "/" en fonction de la détection ou enlève le "\" ou "/" en fonction du réglage de AddSlash

Syntaxe :

```
Resultat.s =
madlib_WrapDirectorySlash(Directory$="",AddSlash.a=#True)
```

Paramètres :

Directory$: Chemin du répertoire
AddSlash.a : Par défaut ajoute le slash ou l'antislash. Passé à *#False* il enlève.

Résultat :

Le chemin modifié.

Exemple :

```
Debug madlib_WrapDirectorySlash("file://toto/a")
Debug madlib_WrapDirectorySlash("file://toto/b/")
Debug madlib_WrapDirectorySlash("file:\\toto\c")
Debug madlib_WrapDirectorySlash("file:\\toto\d\")
Debug madlib_WrapDirectorySlash("file:\\toto\d\",#False)
```

OS Supportés (testé) :

madlib_ReverseReturnCodei

madlib_ReverseReturnCodei()

Description :

Inverse une valeur en négatif ou positif.

Cette fonction est utile pour les fonctions qui retourne les codes API en valeur négative.

Syntaxe :

Resultat.i = madlib_ReverseReturnCodei(ReturnCode.i,NegativeTransformation.i=#PB_Ignore)

Resultat :

Valeur de ReturnCode inversé.

Exemple :

```
Debug madlib_ReverseReturnCodei(123,#True)
Debug madlib_ReverseReturnCodei(123)
Debug madlib_ReverseReturnCodei(123,#False)
```

OS Supportés (testé) :

madlib_ExpandString

madlib_ExpandString()

Description :

Remplacement les variables de substitution contenu dans le texte, par les valeurs associées.

Cette fonction va rechercher les variables d'environements ainsi qu'un certains nombre de variables de base.

Remplace l'ancienne fonction CodeNDF.

Syntaxe :

```
Resultat.s = madlib_ExpandString(Texte$,*Config.madlib_ConfigExpandString=0,MapDICT_CustomVariables.s())
```

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

#True

Paramètre(s) & Option(s) :

IndClef$ => Sert a reconnaitre les indices variables d'environment. (% MaVariable%).

***Config** => Pointeur sur la structure madlib_ConfigExpandString.

DICT_CustomVariables.s() => Permet de spécifier des variables spécifiques.

Variables interne :

Variables	Désignation
madlib_version	Retourne la version de la userlib madlib
version	version de la fonction ExpandString
h	Heure sous format 24h (ex: 14)
mi	Minutes
s	Secondes
yyyy	Année sous format quatre chiffre (ex: 2010)
y	Année sous format deux chiffre (ex: 10)
m	Mois à deux chiffres.
d	Date à deux chiffres (01)
currentdir	Répertoire courant du programme en execution.
programname	Nom du programme lancé sans l'extension.
home	Chemin du répertoire de l'utilisateur courant.

Exemple :

```
Debug madlib_ExpandString("%home%")
Debug madlib_ExpandString("%programmename%")
Debug madlib_ExpandString("%currentdir%")
Debug madlib_ExpandString("%madlib_version%")
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_LogFile

madlib_LogFile()

Syntaxe :

Resultat.b = madlib_LogFile(*InfosJournal.madlib_ConfigLogFile,Text$,Level$="")

Description :

Journalise des événements données. Un réglage des paramètres du fichier de log peut être effectuée grace à la structure ConfigLogFile.

Il est possible aussi de pouvoir utiliser plusieurs fois cette fonction avec plusieurs fichier journal.

Il existe une notion de niveau avec plusieurs sigles, avec une relation de niveau relatif ou absolue.

Par exemple si le niveau global est 3 et qu'un commentaire est "LVL+3" (relatif), alors cette inscription se fera au niveau 6. En revanche si le commentaire et "LVL3" (absolue), alors l'inscription sera au niveau 3.

Paramètre(s) & Option(s) :

Options pour les sigles	Désignation	Sigle dans le fichier
LVL	LEVEL	->
WRN	WARNING	!->
ERR	ERROR	x->

Options pour les niveaux	Désignation
LVL+3	Relatif au niveau global ajouté de 3

	niveaux
LVL3	Absolue, même si le niveau global est au dessus de 0, le commentaire sera au niveau 3.
ERR	ERROR

Structures associées :

madlib_ConfigLogFile

Exemple :

```
Protected log.madlib_ConfigLogFile

log\FileLocation = "%windir%\temp\EssaiMadlibJournal.log"

Journal(@log,"coucou","NIV1")
Journal(@log,"coucou","NIV+1")
```

OS Supportés (testé) :

madlib_ListFile

madlib_ListFile()

Syntaxe :

```
Resultat$ = madlib_ListFile(Directory$,List
LST_FileDirectory.s(),FilterFile$="*.*",Recursif.b=#False,Priority.a=0)
```

Description :

Remplis la liste chaînée du chemin ainsi que les nom de fichiers contenu dans le répertoire ou les sous-répertoires.

Il est possible de placer un filtre pour les fichiers.

En mode récursif il parcours aussi les sous-répertoires.

L'option "priority" va de 0 à 255, par défaut la fonction est au plus haut et donc lors de l'execution sera au maximum de sa vitesse.

Exemple :

```
NewList LST_Fichiers.s()
madlib_ListFile("d:\",LST_Fichiers(),"",#True)
Foreach LST_Fichiers()
    Debug LST_Fichiers()
Next
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_ListDirectory()

Syntaxe :

```
Resultat$ = madlib_ListDirectoryDirectory$,FilterDirectoryName$="",Recursive.b=#False,List
LST_DirectoryResult.s(),Priority.a=0)
```

Description :

Remplis la liste chaînée des chemins des répertoires contenu dans le répertoire ou les sous-répertoires spécifié.

Il est possible de placer un filtre pour les nom des répertoires.

En mode récursif il parcours aussi les sous-répertoires.

L'option "priority" va de 0 à 255, par défaut la fonction est au plus haut et donc lors de l'execution sera au maximum de sa vitesse.

Exemple :

```
NewList LST_Repertoires.s()
madlib_ListDirectory("d:\",LST_Repertoires(),"",#True)
Foreach LST_Repertoires()
    Debug LST_Repertoires()
Next
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_GetDirectorySize()

Syntaxe :

```
Resultat$ =
madlib_GetDirectorySize(DirectoryLocation$,RecursiveMode.a=#False,FilterDirAndFiles.s="*",Priority.a=0)
```

Description :

Permet de récupérer la taille en octets du répertoire demandé. En mode récursif il permettra de d'additionner les tailles de répertoire enfant.

L'option "priority" va de 0 à 255, par défaut la fonction est au plus haut et donc lors de l'execution sera au maximum de sa vitesse.

Exemple :

```
Debug GetDirectorySize("c:/",#True,"*")
Debug GetDirectorySize("d:/",#True)
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_IAmUnicode

madlib_IAmUnicode()

Syntaxe :

```
Resultat.a = madlib_IAmUnicode()
```

Description :

Indique si l'executable est en unicode

Resultat :

- #False : l'executable n'est pas en Unicode
- #True : l'executale est en Unicode.

Exemple :

```
Debug madlib_IAmUnicode()
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_PathExist

madlib_PathExist()

Syntaxe :

Resultat .b= madlib_PathExist(Chemin$)

Description :

Vérifie la présence d'un chemin.

Retourne la valeur "#True " si le fichier existe sinon #False

Exemple :

```
Debug madlib_PathExist("c:\windows")
Debug madlib_FileExist("c:\windows\toto")
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_ShortKeyToNum

madlib_HotKeyToNum()

Syntaxe :

Resultat.i = madlib_ShortKeyToNum(ShortKey$)

Description :

convertis une touche en valeur numérique pour être utilisé dans le code.

Ce code pourras être utilisé dans la fonction **AddKeyboardShortcut()** par exemple.

Liste des touches :

Touche ou fonction	Code AlphaNumérique
Retour Arriere	{back}
Tabulation	{tab}
Supp	{clear}
Retour chariot	{return}
Pause	{pause}
Impression écran	{print}
Majuscule	{capital}
Echap	{escape}
Espace	{space}
Priorité	{prior}
Suivant	{next}
Fin	{end}
Debut	{home}
Gauche	{left}
Haut	{up}
Bas	{down}
Page UP	{pageup}
Page DOWN	{pagedown}
Selection	{select}
Executer	{execute}
Capture	{snapshot}
Insertion	{insert}
Supprimer	{delete}
Aide	{help}
Touche Windows gauche	{leftwindows}
Touche Windows droite	{rightwindows}
Application	{apps}
Pavé numérique de 0 à 9	{pad0} à {pad9}
Multiplié	{multiply}
Addition	{add}
Soustraction	{substrac}

Division	{divide}
Séparateur	{separator}
Décimal	{decimal}
Touche de fonctions de F1 à F24	{f1} à {f24}
Verouillage numérique	{numlock}
Arrêt défilement	{scroll}
Shift	{shift}
Touche de contrôle	{ctrl}
Touche alternative	{alt}
Commande	{command}
Touche de a à z	{a} à {z}
Touche de 0 à 9	{0} à {9}

Exemple :

```
Debug madlib_HotKey("{f3}")
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_ConvertHexRGB

madlib_ConvertHexRGB()

Syntaxe :

Resultat.i= madlib_ConvertHexRGB(ColorHexString$)

Description :

Transforme le format héxadécimal d'une couleur en valeur numérique.

Peut être utilisé avec le caractère # ou $

Exemple :

```
Debug madlib_ConvertHexRGB("#F4024B")
Debug madlib_ConvertHexRGB("$F4024B")
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_CompareVersion()

Description

Compare plusieurs notation de version en texte et indique en résultat si la premierre (Va) est égale ou inférieur ou supérieur à la deuxième (Vb).

Il est possible de modifier le séparateur par défaut par un autre caratère ou autre série de caractère.

Syntaxe :

Resultat.b = madlib_CompareVersion(Va$,Vb$,Sep.s=".")

Resultat :

- **0** : Va et Vb sont egalent.

- **1** : Va est supérieur à Vb.
- **-1** : Vb est supérieur à Va.

Exemple :

```
Debug madlib_CompareVersion("3.2.1","3.002.120")
Debug madlib_CompareVersion("3.0021.12","3.002.120")
Debug madlib_CompareVersion("5.2","5.3.1")
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_DirectoryCompress()

Syntaxe :

Resultat.a = madlib_DirectoryCompress(File$,PackageType.q,Directory$,Recursive.a=#True)

Description :

Compresse un ou plusieurs répertoire enfant et l'enregistre dans un fichier. Il suffit d'indiquer e répertoire a compresser, puis le type de compression. Pour les différents types, se referer à l'aide de Purebasic.

Resultat :

- **#False** : Erreur lors de la compression
- **#True** : Opération terminée.

Exemple :

```
UseZipPacker()

Debug madlib_DirectoryCompress("d:\toto.zip",#PB_Packer_Zip,"d:
\toto",#True)
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_LogOnError()

Description :

Utilisé pour des conditions spécifique, cette fonction permet, en cas de plantage de l'executable lancé en mémoire, d'enregistrer les différentes adresses mémoires, les instructions en assembleur, pour plus facilement "débugger" par la suite. Il enregistre le tout dans un fichier.

Cette fonction par défaut doit être appellée avec la fonction native de la STL Purebasic

OnErrorCall().

Se fichier s'enregistre automatiquement dans le répertoire temporaire de celui qui execute le programme.

Se fichier s'apellera _ERROR_STACK_*NOMDUPROGRAMME*.log

A chaque fois qu'une erreur survient, il enregistre dans le même fichier, sans écraser les informations précédente.

Syntaxe :

```
OnErrorCall(@madlib_LogOnError())
```

Resultat :

Aucun résulat.

Exemple :

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_LoadINIFile()

Syntaxe :

```
Resultat = madlib_LoadINIFile(INIFile$,Map
DICT_INI.madlib_ini(),NameGroupFilter$="",KeyMapCaseMode.i=#PB_Ignore,ReadFormat.i=#PB_Ignore)
```

Description :

Ouvre le fichier ini spécifié et charge dans le dictionnaire le fichier par section et valeur.

Structure(s) associée(s) :

madlib_ini

Parametres :

INIFile$: Chemin du fichier INI.

DICT_INI.madlib_ini() : Dictionnaire du fichier qui contiendra les données.

NameGroupFilters$: Filtre où il est possible de spécifier juste une section.

KeyMapCaseMode.i : #PB_String_LowerCase ou **#PB_String_UpperCase** Passe tout les noms de section et de valeur en minuscule ou majuscule.

ReadFormat.i : #PB_Ascii ou **#PB_Unicode. Lit**

Resultat :

- < 0 : Erreur windows, utiliser la fonction madlib_ReverseReturnCodei().
- #True : Opération effectué correctement.

Exemple :

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_LoadLanguageFilePreBuilt()

Syntaxe :

```
Resultat = madlib_LoadLanguageFilePreBuilt(File$,Array ARR_PreBuilt.s(1),CheckNumberPrebuilt.i=#PB_Ignore)
```

Description :

Permet de charger un tableau de phrase. Cette fonction récupere depuis un fichier le texte et l'index dans un tableau. Il suffit ensuite de l'utiliser avec l'index voulu pour le texte.

Dans le fichier l'éttiquette [PREBUILT] doit être inscrite.

Structure(s) associée(s) :

Parametres :

File$: Chemin du fichier de language (format INI).

Array ARR_rebuilt.s(1) : Tableau de phrase préconstruite

CheckNumberPrebuilt.i : Fixe le nombre de récuperation maximale dans le fichier.

Resultat :

- < 0 : Erreur windows, utiliser la fonction madlib_ReverseReturnCodei().
- > 0 : Opération effectuée correctement, donne le nombre d'entrées récuperées.
- -1 : Récupération effectuée, mais il y a moins de ligne que spécifier dans les paramètres.

Exemple :

```
Contenu du fichier INI :

[PREBUILT]
#1=Chemin du fichier ->
#2=ID de l'ouverture du fichier XML ->
#3=Racine de base pour les réglages
#4=Partie connexion base de données
#5=Nom de la base ->
#6=Partie modèle de requètes
#7=Partie Export
#8=Démarrage des exportations
#9=Connexion non trouvée
#10=Pas de requète de cycle !
#11=Traitement de la requête
#12=Démarrage des autoremplissages
#13=Erreur lors de la création du répertoire
```

Code :

```
Dim TAB_Lng.s(1)
Debug LoadLanguageFilePreBuilt("C:\temp\French.lng",TAB_Lng(),38)
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_LoadLanguageFileWord()

Syntaxe :

```
Resultat = madlib_LoadLanguageFileWord(File$,Map DICT_Word.s(),CheckNumberWord.i=#PB_Ignore)
```

Description :

Permet de de charger des mots ou phrase et des les indexer dans un dictionnaire.
Le but est d'enlever la gestion du texte dans le code.

Structure(s) associée(s) :

Parametres :

File$: Chemin du fichier de language (format INI).

Map DICT_Word.s() : Dictionnaire contenant les mots ou les phrases.

CheckNumberPrebuilt.i : Fixe le nombre de récuperation maximale dans le

fichier.

Resultat :

- < 0 : Erreur windows, utiliser la fonction madlib_ReverseReturnCodei().
- > 0 : Opération effectuée correctement, donne le nombre d'entrées récuperées.
- -1 : Récupération effectuée, mais il y a moins de ligne que spécifier dans les paramètres.

Exemple :

Contenu du fichier INI :

```
[WORD]
file=fichier
location=emplacement
size=taille
repository=dépôt
query=requête(s)
delete=suppression
ok=ok
```

Code :

```
NewMap DICT_Mots.s()
Debug LoadLanguageFileWord("C:\temp\French.lng",DICT_Mots.s())
```

OS Supportés (testé) :

OS Compatibles (non testé) :

Fonctions pour interface graphique

madlib_AjouteObjetFenetre()

Syntaxe :

Resultat$ = madlib_AjoutObjetFenetre(NomObjet$,ID_Fenetre,ID_Objet,CommentairesObjets="")

Description :

Sorte d'annuaire qui permet de mémoriser les id d'objets de plusieurs fenêtres.

Pratique lorsque que l'on génère des fenêtres a la volée, surtout lorsque que l'on utilise des MDI's.

Le résultat sera sous format texte, il représentera le nom de l'étiquette (NomObjets$)

La valeur NomObjet$ est libre. Cepandant il fortement conseillé d'utiliser le nom de la variable qui sera inclue dedans. Si le Handle du widget est contenu dans *MonWidget* alors mettre "*MonWidget*".

La valeur ID_Fenetre doit être soit 0 (correspondant au root de l'annuaire). Sinon il peut être placé le handle de la fenêtre pour monter d'un niveau.

La valeur ID_Objet doit être le handle d'un widget par exemple.

Voici une représentation du système d'annuaire.

```
| |
|  |---> handle_widget_champ_texte
| |---> handle_widget_bouton
|
|
|---> handle_deuxieme_fenetre
| |
| |---> handle_widget_champ_texte
| |---> handle_widget_bouton
|
|
|---> handle_troisieme_fenetre
| |
| |---> handle_widget_champ_texte
| |---> handle_widget_bouton
|
etc...
```

Exemple :

```
IDFen_Pci_Ndfaciliti = OpenWindow(#PB_Any, 472, 156, 400, 400, infosApp
\titreApplicationGlobal, #PB_Window_SystemMenu|
#PB_Window_MinimizeGadget|#PB_Window_TitleBar)
  If IDFen_Pci_Ndfaciliti
    ID_StatusBar_Window = CreateStatusBar(#PB_Any,
WindowID(IDFen_Pci_Ndfaciliti))
    If ID_StatusBar_Window
      AddStatusBarField(20)
      StatusBarText(ID_StatusBar_Window, 0, "-", #PB_StatusBar_Center|
#PB_StatusBar_BorderLess)
      AddStatusBarField(350)
      StatusBarText(ID_StatusBar_Window, 1, "-")
      AddStatusBarField(25)
      StatusBarText(ID_StatusBar_Window, 2, "-")
    EndIf
    If CreateGadgetList(WindowID(IDFen_Pci_Ndfaciliti))
      BT_Connexion = ButtonImageGadget(#PB_Any, 311, 333, 40, 40,
CatchImage(#Image_ButtonImage_7, ?Image_ButtonImage_7))
      BT_Deconnexion = ButtonImageGadget(#PB_Any, 270, 333, 40, 40,
CatchImage(#Image_ButtonImage_8, ?Image_ButtonImage_8))
      BT_ForceConnexion = ButtonImageGadget(#PB_Any, 10, 335, 35, 35,
CatchImage(#Image_ButtonImage_13, ?
Image_ButtonImage_13),#PB_Button_Toggle)
      SetWindowLong_(GadgetID(BT_ForceConnexion), #GWL_STYLE,
GetWindowLong_(GadgetID(BT_ForceConnexion), #GWL_STYLE) |
#PB_Button_Toggle)
      PAN_Menu = PanelGadget(#PB_Any, 1, 1, 400, 330)
      ; Liste des actions
        AddGadgetItem(PAN_Menu, -1, "Liste des actions",
CatchImage(#Image_Panel_9_1, ?Image_Panel_9_1))
        LST_ActionsApplications = ListIconGadget(#PB_Any, 0, 1, 394,
302, "", 300)
        SendMessage_(GadgetID(LST_ActionsApplications),
#LVM_SETCOLUMNWIDTH, 0, #LVSCW_AUTOSIZE_USEHEADER)
        SetGadgetAttribute(LST_ActionsApplications,#PB_ListIcon_DisplayM
ode,#PB_ListIcon_List)
```

```
        ; Activités
        AddGadgetItem(PAN_Menu, -1, "Activités",
CatchImage(#Image_Panel_9_2, ?Image_Panel_9_2))
        LST_Activite = ListIconGadget(#PB_Any, 0, 0, 394, 304, "ico",
20, #PB_ListIcon_AlwaysShowSelection|#PB_ListIcon_FullRowSelect|
#LVS_NOCOLUMNHEADER)
        AddGadgetColumn(LST_Activite, 1, "Texte", 100)
        SendMessage_(GadgetID(LST_Activite), #LVM_SETCOLUMNWIDTH, 1,
#LVSCW_AUTOSIZE_USEHEADER)
        ; Console
        AddGadgetItem(PAN_Menu, -1, "Console",
CatchImage(#Image_Panel_9_3, ?Image_Panel_9_3))
        LST_Console = ListViewGadget(#PB_Any, 0, 0, 394, 304)
      CloseGadgetList()
      ; Gadget Fonts
      SetGadgetFont(LST_Activite, FontID(LoadFont(#PB_Any, "DejaVu
Sans", 8, #PB_Font_HighQuality)))
      SetGadgetFont(LST_ActionsApplications, FontID(LoadFont(#PB_Any,
"DejaVu Sans", 8, #PB_Font_HighQuality)))
      SetGadgetFont(PAN_Menu, FontID(LoadFont(#PB_Any, "DejaVu Sans", 8,
#PB_Font_HighQuality)))
      SetGadgetFont(LST_Console, FontID(LoadFont(#PB_Any, "Consolas", 8,
#PB_Font_HighQuality)))
      ; Gadget Colors
      SetGadgetColor(LST_Console, #PB_Gadget_FrontColor, RGB(0, 255, 0))
      SetGadgetColor(LST_Console, #PB_Gadget_BackColor, 0)

      ;Désactivation initial
      DisableGadget(BT_Deconnexion,1)

      ;Ajout des items dans l'annuaire


      ; ID de la fenêtre principale
      AjouteObjetFenetre("IDFen_Pci_Ndfaciliti",0,IDFen_Pci_Ndfaciliti)

      ;Il faut remarquer que les ID's des widget sont contenu dans le
"répertoire" de l'id de la fenêtre principale.
      AjouteObjetFenetre("ID_StatusBar_Window",IDFen_Pci_Ndfaciliti,ID_S
tatusBar_Window)
      AjouteObjetFenetre("BT_Connexion",IDFen_Pci_Ndfaciliti,BT_Connexio
n)
      AjouteObjetFenetre("BT_Deconnexion",IDFen_Pci_Ndfaciliti,BT_Deconn
exion)
      AjouteObjetFenetre("BT_ForceConnexion",IDFen_Pci_Ndfaciliti,BT_For
ceConnexion)
      AjouteObjetFenetre("PAN_Menu",IDFen_Pci_Ndfaciliti,PAN_Menu)
      AjouteObjetFenetre("LST_ActionsApplications",IDFen_Pci_Ndfaciliti,
LST_ActionsApplications)
      AjouteObjetFenetre("LST_Activite",IDFen_Pci_Ndfaciliti,LST_Activit
e)
      AjouteObjetFenetre("LST_Console",IDFen_Pci_Ndfaciliti,LST_Console)
```

OS Supportés (testé) :

madlib_IDObjetFenetre()

Syntaxe :

Resultat = madlib_IDObjetFenetre(NomObjet$,ID_Fenetre)

Description :

Ressort de l'annuaire le Handle de l'objet ou widget que l'on à placé avec la fonction "AjouteObjetFenetre".

Exemple :

```
IDFen_Pci_Ndfaciliti = OpenWindow(#PB_Any, 472, 156, 400, 400, infosApp
\titreApplicationGlobal, #PB_Window_SystemMenu|
#PB_Window_MinimizeGadget|#PB_Window_TitleBar)
  If IDFen_Pci_Ndfaciliti
    ID_StatusBar_Window = CreateStatusBar(#PB_Any,
WindowID(IDFen_Pci_Ndfaciliti))
    If ID_StatusBar_Window
      AddStatusBarField(20)
      StatusBarText(ID_StatusBar_Window, 0, "-", #PB_StatusBar_Center|
#PB_StatusBar_BorderLess)
      AddStatusBarField(350)
      StatusBarText(ID_StatusBar_Window, 1, "-")
      AddStatusBarField(25)
      StatusBarText(ID_StatusBar_Window, 2, "-")
    EndIf
    If CreateGadgetList(WindowID(IDFen_Pci_Ndfaciliti))
      BT_Connexion = ButtonImageGadget(#PB_Any, 311, 333, 40, 40,
CatchImage(#Image_ButtonImage_7, ?Image_ButtonImage_7))
      BT_Deconnexion = ButtonImageGadget(#PB_Any, 270, 333, 40, 40,
CatchImage(#Image_ButtonImage_8, ?Image_ButtonImage_8))
      BT_ForceConnexion = ButtonImageGadget(#PB_Any, 10, 335, 35, 35,
CatchImage(#Image_ButtonImage_13, ?
Image_ButtonImage_13),#PB_Button_Toggle)
      SetWindowLong_(GadgetID(BT_ForceConnexion), #GWL_STYLE,
GetWindowLong_(GadgetID(BT_ForceConnexion), #GWL_STYLE) |
#PB_Button_Toggle)
```

```
      PAN_Menu = PanelGadget(#PB_Any, 1, 1, 400, 330)
      ; Liste des actions
        AddGadgetItem(PAN_Menu, -1, "Liste des actions",
CatchImage(#Image_Panel_9_1, ?Image_Panel_9_1))
        LST_ActionsApplications = ListIconGadget(#PB_Any, 0, 1, 394,
302, "", 300)
        SendMessage_(GadgetID(LST_ActionsApplications),
#LVM_SETCOLUMNWIDTH, 0, #LVSCW_AUTOSIZE_USEHEADER)
        SetGadgetAttribute(LST_ActionsApplications,#PB_ListIcon_DisplayM
ode,#PB_ListIcon_List)
        ; Activités
        AddGadgetItem(PAN_Menu, -1, "Activités",
CatchImage(#Image_Panel_9_2, ?Image_Panel_9_2))
        LST_Activite = ListIconGadget(#PB_Any, 0, 0, 394, 304, "ico",
20, #PB_ListIcon_AlwaysShowSelection|#PB_ListIcon_FullRowSelect|
#LVS_NOCOLUMNHEADER)
        AddGadgetColumn(LST_Activite, 1, "Texte", 100)
        SendMessage_(GadgetID(LST_Activite), #LVM_SETCOLUMNWIDTH, 1,
#LVSCW_AUTOSIZE_USEHEADER)
        ; Console
        AddGadgetItem(PAN_Menu, -1, "Console",
CatchImage(#Image_Panel_9_3, ?Image_Panel_9_3))
        LST_Console = ListViewGadget(#PB_Any, 0, 0, 394, 304)
      CloseGadgetList()
      ; Gadget Fonts
      SetGadgetFont(LST_Activite, FontID(LoadFont(#PB_Any, "DejaVu
Sans", 8, #PB_Font_HighQuality)))
      SetGadgetFont(LST_ActionsApplications, FontID(LoadFont(#PB_Any,
"DejaVu Sans", 8, #PB_Font_HighQuality)))
      SetGadgetFont(PAN_Menu, FontID(LoadFont(#PB_Any, "DejaVu Sans", 8,
#PB_Font_HighQuality)))
      SetGadgetFont(LST_Console, FontID(LoadFont(#PB_Any, "Consolas", 8,
#PB_Font_HighQuality)))
      ; Gadget Colors
      SetGadgetColor(LST_Console, #PB_Gadget_FrontColor, RGB(0, 255, 0))
      SetGadgetColor(LST_Console, #PB_Gadget_BackColor, 0)

      ;Désactivation initial
      DisableGadget(BT_Deconnexion,1)

      ;Ajout des items dans l'annuaire


      ; ID de la fenêtre principale
      AjouteObjetFenetre("IDFen_Pci_Ndfaciliti",0,IDFen_Pci_Ndfaciliti)

      ;Il faut remarquer que les ID's des widget sont contenu dans le
"répertoire" de l'id de la fenêtre principale.
      madlib_AjouteObjetFenetre("ID_StatusBar_Window",IDFen_Pci_Ndfacili
ti,ID_StatusBar_Window)
      madlib_AjouteObjetFenetre("BT_Connexion",IDFen_Pci_Ndfaciliti,BT_C
onnexion)
      madlib_AjouteObjetFenetre("BT_Deconnexion",IDFen_Pci_Ndfaciliti,BT
_Deconnexion)
      madlib_AjouteObjetFenetre("BT_ForceConnexion",IDFen_Pci_Ndfaciliti
,BT_ForceConnexion)
      madlib_AjouteObjetFenetre("PAN_Menu",IDFen_Pci_Ndfaciliti,PAN_Menu
)
      madlib_AjouteObjetFenetre("LST_ActionsApplications",IDFen_Pci_Ndfa
ciliti,LST_ActionsApplications)
      madlib_AjouteObjetFenetre("LST_Activite",IDFen_Pci_Ndfaciliti,LST_
Activite)
      madlib_AjouteObjetFenetre("LST_Console",IDFen_Pci_Ndfaciliti,LST_C
```

```
onsole)


Debug
madlib_IDObjetFenetre("BT_Connexion",IDObjetfenetre("IDFen_Pci_Ndfacilit
i",0))
```

OS Supportés (testé) :

madlib_CommentaireObjetFenetre

madlib_CommentaireObjetFenetre()

Syntaxe :

Resultat = madlib_CommentairesObjetFenetre(NomObjet$,ID_Fenetre)

Description :

Ressort le commentaire associé au nom de l'objet stocké avec la fonction "AjouteObjetFenetre"..

Exemple :

```
IDFen_Pci_Ndfaciliti = OpenWindow(#PB_Any, 472, 156, 400, 400, infosApp
\titreApplicationGlobal, #PB_Window_SystemMenu|
#PB_Window_MinimizeGadget|#PB_Window_TitleBar)
  If IDFen_Pci_Ndfaciliti
    ID_StatusBar_Window = CreateStatusBar(#PB_Any,
WindowID(IDFen_Pci_Ndfaciliti))
    If ID_StatusBar_Window
      AddStatusBarField(20)
      StatusBarText(ID_StatusBar_Window, 0, "-", #PB_StatusBar_Center|
#PB_StatusBar_BorderLess)
      AddStatusBarField(350)
      StatusBarText(ID_StatusBar_Window, 1, "-")
      AddStatusBarField(25)
      StatusBarText(ID_StatusBar_Window, 2, "-")
    EndIf
```

```
    If CreateGadgetList(WindowID(IDFen_Pci_Ndfaciliti))
      BT_Connexion = ButtonImageGadget(#PB_Any, 311, 333, 40, 40,
CatchImage(#Image_ButtonImage_7, ?Image_ButtonImage_7))
      BT_Deconnexion = ButtonImageGadget(#PB_Any, 270, 333, 40, 40,
CatchImage(#Image_ButtonImage_8, ?Image_ButtonImage_8))
      BT_ForceConnexion = ButtonImageGadget(#PB_Any, 10, 335, 35, 35,
CatchImage(#Image_ButtonImage_13, ?
Image_ButtonImage_13),#PB_Button_Toggle)
      SetWindowLong_(GadgetID(BT_ForceConnexion), #GWL_STYLE,
GetWindowLong_(GadgetID(BT_ForceConnexion), #GWL_STYLE) |
#PB_Button_Toggle)
      PAN_Menu = PanelGadget(#PB_Any, 1, 1, 400, 330)
      ; Liste des actions
        AddGadgetItem(PAN_Menu, -1, "Liste des actions",
CatchImage(#Image_Panel_9_1, ?Image_Panel_9_1))
        LST_ActionsApplications = ListIconGadget(#PB_Any, 0, 1, 394,
302, "", 300)
        SendMessage_(GadgetID(LST_ActionsApplications),
#LVM_SETCOLUMNWIDTH, 0, #LVSCW_AUTOSIZE_USEHEADER)
        SetGadgetAttribute(LST_ActionsApplications,#PB_ListIcon_DisplayM
ode,#PB_ListIcon_List)
        ; Activités
        AddGadgetItem(PAN_Menu, -1, "Activités",
CatchImage(#Image_Panel_9_2, ?Image_Panel_9_2))
        LST_Activite = ListIconGadget(#PB_Any, 0, 0, 394, 304, "ico",
20, #PB_ListIcon_AlwaysShowSelection|#PB_ListIcon_FullRowSelect|
#LVS_NOCOLUMNHEADER)
        AddGadgetColumn(LST_Activite, 1, "Texte", 100)
        SendMessage_(GadgetID(LST_Activite), #LVM_SETCOLUMNWIDTH, 1,
#LVSCW_AUTOSIZE_USEHEADER)
        ; Console
        AddGadgetItem(PAN_Menu, -1, "Console",
CatchImage(#Image_Panel_9_3, ?Image_Panel_9_3))
        LST_Console = ListViewGadget(#PB_Any, 0, 0, 394, 304)
      CloseGadgetList()
      ; Gadget Fonts
      SetGadgetFont(LST_Activite, FontID(LoadFont(#PB_Any, "DejaVu
Sans", 8, #PB_Font_HighQuality)))
      SetGadgetFont(LST_ActionsApplications, FontID(LoadFont(#PB_Any,
"DejaVu Sans", 8, #PB_Font_HighQuality)))
      SetGadgetFont(PAN_Menu, FontID(LoadFont(#PB_Any, "DejaVu Sans", 8,
#PB_Font_HighQuality)))
      SetGadgetFont(LST_Console, FontID(LoadFont(#PB_Any, "Consolas", 8,
#PB_Font_HighQuality)))
      ; Gadget Colors
      SetGadgetColor(LST_Console, #PB_Gadget_FrontColor, RGB(0, 255, 0))
      SetGadgetColor(LST_Console, #PB_Gadget_BackColor, 0)

      ;Désactivation initial
      DisableGadget(BT_Deconnexion,1)

      ;Ajout des items dans l'annuaire


      ; ID de la fenêtre principale
      AjouteObjetFenetre("IDFen_Pci_Ndfaciliti",0,IDFen_Pci_Ndfaciliti)

      ;Il faut remarquer que les ID's des widget sont contenu dans le
"répertoire" de l'id de la fenêtre principale.
      madlib_AjouteObjetFenetre("ID_StatusBar_Window",IDFen_Pci_Ndfacili
ti,ID_StatusBar_Window)
      madlib_AjouteObjetFenetre("BT_Connexion",IDFen_Pci_Ndfaciliti,BT_C
onnexion)
```

```
      madlib_AjouteObjetFenetre("BT_Deconnexion",IDFen_Pci_Ndfaciliti,BT
_Deconnexion)
       madlib_AjouteObjetFenetre("BT_ForceConnexion",IDFen_Pci_Ndfacilit
i,BT_ForceConnexion)
       madlib_AjouteObjetFenetre("PAN_Menu",IDFen_Pci_Ndfaciliti,PAN_Men
u)
madlib_AjouteObjetFenetre("LST_ActionsApplications",IDFen_Pci_Ndfaciliti
,LST_ActionsApplications)
      madlib_AjouteObjetFenetre("LST_Activite",IDFen_Pci_Ndfaciliti,LST_
Activite)
      madlib_AjouteObjetFenetre("LST_Console",IDFen_Pci_Ndfaciliti,LST_C
onsole)


Debug
madlib_CommentaireObjetFenetre("BT_Connexion",IDObjetfenetre("IDFen_Pci_
Ndfaciliti",0))
```

OS Supportés (testé) :

madlib_SystrayBalloon

madlib_SystrayBalloon()

Syntaxe :

```
Resultat =
madlib_SystrayBalloon_(Titre$,Texte$,Icone=#NIIF_NONE,TempAffichageSec=10,IDSystray=#PB_Ignore,IDF
enetre=#PB_Ignore)
```

Description :

Permet d'afficher un message de type "Balloon" dans la barre des tâches

Exemple :

```
madlib_SystrayBalloon("Mon Titre","Mon Texte",#NIIF_WARNING)

Delay(10000)
```

OS Supportés (testé) :

madlib_RechercheElement

madlib_RechercheElement()

Syntaxe :

Resultat =
madlib_RechercheElement(IDObjetTraiter,TexteRecherche$="",TraitementColonne=0,TraitementElementDATA=#False,RespectMinusculeMajuscule=#True)

Description :

Permet de rechercher une chaine de caractère dans un Widget de type list ou listicon.

Cette fonction ne renvoi rien en retour.

Paramètres :

Paramètre	Désignation	Valeur
IDObjetTraiter	Handle du widget	
TexteRecherche$	Chaine de caractère recherché	
TraitementColonne	Numéro de colonne a utiliser pour la recherche, par défaut 0.	
TraitementElementDATA	Option pour ajouter directement	
RespectMinusculeMajuscule	Comme son nom l'indique Case sensitive	booléen (#True / #False)

	sur la recherche.	

Exemple :

OS Supportés (testé) :

madlib_SplachScreen()

Syntaxe :

```
Resultat.a = madlib_SplachScreen(*Config.madlib_ConfigSplachScreen)
```

Description :

Affiche un logo, du texte et une barre de progression, automatiquement. Cette fonction peut être utilisé pour le démarrage d'une application ou une fenêtre d'attente. Une structure permet de configurer les options.

Les formats d'images pris en charge : JPG, PNG, BMP.

Paramètres & options :

-

Structure(s) associée(s) :

- madlib_ConfigSplachScreen

Resultat :

- #True : Arrêt correcte de la fenêtre.
- #False : Erreur dans les paramètres, au chargement et à l'initialisation de la fenêtre.

Exemple :

```
Define splscr.madlib_ConfigSplachScreen

splscr\ImageFile = "d:\NumberSign.jpg"
splscr\TimeOut = 10
splscr\Ypos = #PB_Ignore
splscr\Xpos = #PB_Ignore
splscr\Alpha = 0
splscr\ProgressBar = #True
splscr\Text = #False
splscr\BackColorProgressBar = "0025aa"
splscr\BackGroundTextColor = "0025aa"


Procedure.a SplachScreen(*splscr.madlib_ConfigSplachScreen)
   madlib_SplachScreen(*splscr)
EndProcedure

CreateThread(@SplachScreen(),@splscr)

Delay (1050)
SetGadgetState(splscr\_IDGadgetProgressBar,40)
Delay (50)
SetGadgetState(splscr\_IDGadgetProgressBar,60)
Delay (1000)
SetGadgetState(splscr\_IDGadgetProgressBar,100)

Delay (1000)
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_CenterWindow

madlib_CenterWindow()

Syntaxe :

```
Resultat.a =
madlib_CenterWindow(HandleWindow.i,PositionX.i=#PB_Ignore,PositionY.i=#PB_Ignore,Desktop.a=0
)
```

Description :

Centre une fenêtre en hauteur ou en largeur par rapport à une taille d'écran.

Paramètres & options :

HandleWindow.i => ID de la fenêtre à modifier

PositionX.i => Position en hauteur de la fenêtre. Si PB_Ignore est spécifié par défaut ça sera au centre

PositionY.i => Position en Largeur de la fenêtre. Si PB_Ignore est spécifié par défaut ça sera au centre

Desktop.a => Ecran permettant le centrage de la fenêtre. Par défaut (0 > écran par défaut)

Resultat :

- #True : Centrage OK
- #False : Erreur fenêtre non trouvé ou impossible de charger les écrans.

Exemple :

```
If OpenWindow(0, 10, 500, 1020, 26, "PureBasic Window",
#PB_Window_SystemMenu | #PB_Window_MinimizeGadget |
```

```
    #PB_Window_MaximizeGadget)

       madlib_CenterWindow(0)


      Repeat
        Event = WaitWindowEvent()

        If Event = #PB_Event_CloseWindow
          Quit = 1
        EndIf

      Until Quit = 1

    EndIf

    End
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_WindowNotify()

Syntaxe :

Resultat.a=
madlib_WindowNotify(*Config.madlib_ConfigWindowNotify)

Description :

Affiche une fenêtre de notification, sur le même comportement que l'infobulle de Windows par exemple. Il est possible de configurer le temp d'affichage du message, le titre les couleurs, les polices.

Par défaut, la taille de la fenêtre (hauteur et largeur) sont automatiquement redéfini par la taille de l'écran.

Il est possible aussi de charger un icône pour identifier la fenêtre plus facilement. La dimension maximale est de 20x20 pixels.

 Cette fonction ne convient pas au programme qui ont déja une boucle de message événementiel. (WindowEvent, etc...)

Structure(s) associée(s) :

- madlib_ConfigWindowNotify

Resultat :

- #False : Erreur lors de la préparation de la fenêtre ou de l'affichage
- #True : Opération terminé

Exemple :

```
Define ConfigNotif.madlib_ConfigWindowNotify

ConfigNotif\Text = "C'est un texte de
démonstration..."+#CRLF$+#CRLF$+"Sur plusieurs lignes !"
ConfigNotif\Title = "Ceci est un titre"
ConfigNotif\WindowColor = ConvertHexRGB("$B4E9FF")
ConfigNotif\BorderColor = ConvertHexRGB("#114da7")
ConfigNotif\TitleColor = RGB($fe,$fe,$fe)
ConfigNotif\TextColor = RGB($6,$c02,$77)
;ConfigNotif\TimeOut = 0 ; aucun arrêt, la fenêtre glisse
ConfigNotif\TimeOut = #PB_Ignore
ConfigNotif\SpeedScrollDown = 1
ConfigNotif\TypeFontText = "Consolas"
ConfigNotif\TypeFontTitle = "Verdana"
ConfigNotif\ForeFront = #True
;ConfigNotif\NoCloseWindow = #True
;ConfigNotif\Icon = ImageID(LoadImage(#PB_Any,"D:
```

```
    \MonIcone.ico"))

    WindowNotify(@ConfigNotif)
```

OS Supportés (testé) :

OS Compatibles (non testé) :

Chiffrement

Fonctions de chiffrement

madlib_StringDecrypt

madlib_StringDecrypt()

Syntaxe :

Resultat$ = madlib_StringDecrypt(Texte$,Clef$,NiveauForce=1)

Description :

Déchiffre une chaîne de caractère avec une clef.

Le niveau de force est règle la complexité.

Le résultat sera sous format texte.

Exemple :

```
MaChaineChiffre$ = madlib_StringCrypt("coucou","maclef",4)
Debug madlib_StringDecrypt(MaChaineChiffre$,"maclef",4)
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_StringCrypt

madlib_StringCrypt()

Syntaxe :

Resultat$ = madlib_StringCrypt(Texte$,Clef$,NiveauForce=1)

Description :

Chiffre une chaîne de caractère avec une clef.

Le niveau de force est règle la complexité.

Le résultat sera sous format texte.

Exemple :

```
Debug madlib_StringCrypt("coucou","maclef",4)
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_VerifFichierChiffre

madlib_VerifFichierChiffre()

Syntaxe :

Resultat.b = madlib_VerifFichierChiffre(EmplacementFichier$,ClefDeChiffrement$)

Description :

Vérifie si le fichier est chiffré avec la clef demandé.

Si le fichier est chiffrée avec cet clef de chiffrement, le retour se #True (vrai), sinon #False (Faux).

Exemple :

```
Debug madlib_VerifFichierChiffre("c:\toto.txt","MaClef")
```

OS Supportés (testé) :

madlib_ChiffreFichier

madlib_ChiffreFichier()

Syntaxe :

Resultat.b = madlib_ChiffreFichier(EmplacementFichier$,ClefDeChiffrement$,NiveauChiffrement=1)

Description :

Chiffre un fichier complètement.

Il faut spécifier un fichier et une clef de chiffrement au minimum.

Si l'opération s'effectue correctement, il retournera #True (vrai). Sinon #False (Faux).

Exemple :

```
Debug madlib_ChiffreFichier("c:\toto.txt","MaClef")
```

OS Supportés (testé) :

madlib_DechiffreFichier

madlib_DechiffreFichier()

Syntaxe :

```
Resultat.b = madlib_DechiffreFichier(EmplacementFichier$,List
ContenuFichier.s(),ClefDeChiffrement$="",NiveauChiffrement=1)
```

Description :

Déchiffre le fichier dans une liste chaîné.

Il faut spécifier un fichier et une clef de chiffrement au minimum.

Si l'opération s'effectue correctement, il retournera #True (vrai). Sinon #False (Faux).

Exemple :

```
NewList MonFichierAuClair.s()

Debug madlib_DechiffreFichier("c:
\toto.txt",MonFichierAuClair(),"MaClef")
```

OS Supportés (testé) :

Serveur IP

Serveur IP

Dans cette partie, les fonctions pour la création et gestion de serveur IP.

Les différentes fonctions utilisent des structures contenant les différentes informations sur l'implémentation de la configuration du serveur IP en écoute.

 Il est conseillé d'utiliser la gestion des "Multi-Thread". En effet ces fonctions sont intègrées en plusieurs processus.

madlib_ServeurIP

madlib_ServeurIP()

Syntaxe :

Resultat = madlib_ServeurIP(*config.madlib_ConfigServeurIP)

Structure(s) associée(s) :

madlib_ConfigServeurIP

Description :

Serveur d'écoute IP

 Il est impératif de lancer *InitNetwork()* avant toutes opérations.

Constantes associées :

#MADLIB_SERVEURIP_ENCOURS_CREATION

#MADLIB_SERVEURIP_ENCOURS_EXECUTION

#MADLIB_SERVEURIP_ERREUR_CREATION_SERVEUR

#MADLIB_SERVEURIP_ERREUR_ALLOCATION_MEMOIRE

#MADLIB_SERVEURIP_ERREUR_RECUPERATION_DONNEES

#MADLIB_SERVEURIP_NOMBRE_MACHINES_MAX

#MADLIB_SERVEURIP_ATTENTE_RETOUR

#MADLIB_SERVEURIP_TYPE_RECEPTION_DONNEES

#MADLIB_SERVEURIP_TYPE_RECEPTION_FICHIER

Exemple :

```
Procedure MonServeurTCP(*SuperIP.madlib_ConfigServeurIP)
    madlib_ServeurIP(*SuperIP)
EndProcedure



SuperIP.madlib_ConfigServeurIP

SuperIP\PortEcoute = 6060
SuperIP\TailleBuffer = 1024 ; En octets
SuperIP\InviteALaConnexion$ = "bonjour >"
superip\RafraEvenements = 10 ; Millisecondes
SuperIP\SEM_AttenteRetourReception = -1
SuperIP\ModeUDP = #False ; Ne sert à rien juste pour la démo, #False par
défaut.
SuperIP\TypeServeur = 0



InitNetwork()


CreateThread(@MonServeurTCP(),@SuperIP)
```

```
If madlib_AttenteCreationServeurIP(@SuperIP) = #True
    Repeat
              ;Attente du signal du sémaphore
              WaitSemaphore(SuperIP\SEM_ReceptionDonnees)

              Donnees$ = madlib_RecupereDonneesServeurIP(@SuperIP,-
1,0,Chr(13)+Chr(10))

              Debug Donnees$

              ;En fonction de la définition du fichier de conf TELESCPRO
              SuperIP\RetourReception$ = "Reçu merci bien !"

              SignalSemaphore(SuperIP\SEM_AttenteRetourReception)

              Donnees$ =  ""
    ForEver
EndIf
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_RecupereDonneesServeurIP

madlib_RecupereDonneesServeurIP ()

Description :

Pendant l'execution d'un serveur IP en écoute, le processus intègre un déclancheur (sémaphore), pour récupérer les données. Une fois que le sémaphore à signalé une donnée, cette fonction permet de récupérer les données.

Il est possible de supprimer des caractères non voulu par le formatage

Si la longueur n'est connu, ni voulu, pour une taille illimité utiliser la valeur : *-1*

L'option format texte est utilisé pour le type d'encodage (UTF8, ANSI).

Syntaxe :

Resultat = madlib_RecupereDonneesServeurIP(*config.madlib_ConfigServeurIP,LongueurBuffer.i=-1,FormatTexte.i=0,SupprimeRetourFinDeLigne$="")

Structure(s) associée(s) :

madlib_ConfigServeurIP

Exemple :

```
Procedure MonServeurTCP(*SuperIP.madlib_ConfigServeurIP)
    madlib_ServeurIP(*SuperIP)
EndProcedure



SuperIP.madlib_ConfigServeurIP

SuperIP\PortEcoute = 6060
SuperIP\TailleBuffer = 1024 ; En octets
SuperIP\InviteALaConnexion$ = "bonjour >"
superip\RafraEvenements = 10 ; Millisecondes
SuperIP\SEM_AttenteRetourReception = -1
SuperIP\ModeUDP = #False ; Ne sert à rien juste pour la démo, #False par
défaut.
SuperIP\TypeServeur = 0




InitNetwork()


CreateThread(@MonServeurTCP(),@SuperIP)

If madlib_AttenteCreationServeurIP(@SuperIP) = #True
    Repeat
              ;Attente du signal du sémaphore
              WaitSemaphore(SuperIP\SEM_ReceptionDonnees)

              Donnees$ = madlib_RecupereDonneesServeurIP(@SuperIP,-
1,0,Chr(13)+Chr(10))
```

```
                Debug Donnees$

                ;En fonction de la définition du fichier de conf TELESCPRO
                SuperIP\RetourReception$ = "Reçu merci bien !"

                SignalSemaphore(SuperIP\SEM_AttenteRetourReception)

                Donnees$ =  ""
        ForEver
    EndIf
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_RequestIPServer

madlib_RequestIPServer()

Description :

Système

Syntaxe :

Resultat = madlib_RecupereDonneesServeurIP(*config.madlib_ConfigServeurIP,LongueurBuffer.i=-1,FormatTexte.i=0,SupprimeRetourFinDeLigne$="")

Structure(s) associée(s) :

madlib_ConfigServeurIP

Exemple :

```
Procedure MonServeurTCP(*SuperIP.madlib_ConfigServeurIP)
    madlib_ServeurIP(*SuperIP)
EndProcedure



SuperIP.madlib_ConfigServeurIP

SuperIP\PortEcoute = 6060
SuperIP\TailleBuffer = 1024 ; En octets
SuperIP\InviteALaConnexion$ = "bonjour >"
superip\RafraEvenements = 10 ; Millisecondes
SuperIP\SEM_AttenteRetourReception = -1
SuperIP\ModeUDP = #False ; Ne sert à rien juste pour la démo, #False par
défaut.
SuperIP\TypeServeur = 0




InitNetwork()


CreateThread(@MonServeurTCP(),@SuperIP)

If madlib_AttenteCreationServeurIP(@SuperIP) = #True
    Repeat
              ;Attente du signal du sémaphore
              WaitSemaphore(SuperIP\SEM_ReceptionDonnees)

              Donnees$ = madlib_RecupereDonneesServeurIP(@SuperIP,-
1,0,Chr(13)+Chr(10))

              Debug Donnees$

              ;En fonction de la définition du fichier de conf TELESCPRO
              SuperIP\RetourReception$ = "Reçu merci bien !"

              SignalSemaphore(SuperIP\SEM_AttenteRetourReception)

              Donnees$ =  ""
    ForEver
EndIf
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_AttenteCreationServeurIP

madlib_AttenteCreationServeurIP()

Description :

Lors du lancement de la fonction, madlib_ServeurIP, des initialisations de variables et sous processus s'effectuent. La fonction d'attente permet de surveiller et attendre le démarrage et de rendre la main une fois le serveur IP démarré correctement.

Syntaxe :

Resultat.b = madlib_AttenteCreationServeurIP(*config.madlib_ConfigServeurIP)

Structure(s) associée(s) :

madlib_ConfigServeurIP

Exemple :

```
Procedure MonServeurTCP(*SuperIP.madlib_ConfigServeurIP)
    madlib_ServeurIP(*SuperIP)
EndProcedure



SuperIP.madlib_ConfigServeurIP

SuperIP\PortEcoute = 6060
SuperIP\TailleBuffer = 1024 ; En octets
SuperIP\InviteALaConnexion$ = "bonjour >"
superip\RafraEvenements = 10 ; Millisecondes
SuperIP\SEM_AttenteRetourReception = -1
```

```
SuperIP\ModeUDP = #False ; Ne sert à rien juste pour la démo, #False par
défaut.
SuperIP\TypeServeur = 0



InitNetwork()


CreateThread(@MonServeurTCP(),@SuperIP)

If madlib_AttenteCreationServeurIP(@SuperIP) = #True
    Repeat
              ;Attente du signal du sémaphore
              WaitSemaphore(SuperIP\SEM_ReceptionDonnees)

              Donnees$ = madlib_RecupereDonneesServeurIP(@SuperIP,-
1,0,Chr(13)+Chr(10))

              Debug Donnees$

              ;En fonction de la définition du fichier de conf TELESCPRO
              SuperIP\RetourReception$ = "Reçu merci bien !"

              SignalSemaphore(SuperIP\SEM_AttenteRetourReception)

              Donnees$ =  ""
    ForEver
EndIf
```

OS Supportés (testé) :

OS Compatibles (non testé) :

XML

Fonctions pour le XML

Cette rubrique concerne les fonctions XML. Ce sont des fonctions qui utilisent les librairies basées sur Expat.

madlib_LoadFileXML

madlib_LoadFileXML()

Description :

Charge un fichier XML en mémoire, transformé sous format en base de données.

Les noeuds sont séparé des attributs. Il serat possible de retrouver avec les fonctions les noeud et ses attributs associés.

Syntaxe :

```
Resultat.l = madlib_LoadFileXML(FileLocationXML$,StartBrowsingAt.s="",FormatXML.i=#PB_Ignore,TypeEncodeFile.i=#PB_Ignore)
```

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

Identifiant qui sera utilisé et demandé par les autres fonctions.

Il sera possible de requeter manuellement dans la base.

Paramètre(s) & Option(s) :

FileLocation$:

Chemin du fichier à charger.

StartBrowsingAt.s :

Emplacement de départ du chargement. Il est possible de charger qu'une partie du fichier.

FormatXML.i :

Permet de spécifier des options tel que suppression des retour chariots, etc ...

TypeEncodeFile.i :

Permet de spécifier l'encodage du fichier (UTF-8, Ansi, etc...).

Structure(s) associée(s) :

-

Exemple :

```
madlib_LoadFileXML("c:\test.xml",LST_InstructionsXML())

ForEach LST_InstructionsXML()
   Debug LST_InstructionsXML()\Names$
Next
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_GetXMLBaseNodeList

madlib_GetXMLBaseNodeList()

Description :

Remplis un dictionnaire de hashage de structure madlib_XMLNodes(), contenant les noeuds demandé.

Il est possible de filtrer les noeuds ou l'emplacement.

Syntaxe :

```
Resultat.i = madlib_GetXMLBaseNodeList(*IDBaseXML,Map
NodeList.madlib_XMLNodes(),NodeName.s="",StartBrowsingAt.s="/")
```

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

Taille de la liste

Paramètre(s) & Option(s) :

*IDBaseXML :

Identifiant provenant de la fonction madlib_FileXML.

NodeList.madlib_XMLNodes() :

Dictionnaire de hashage qui contient le résultat de la recherche demandé.

NodeName.s :

Nom de recherche souhaité. Il est possible de spécifier un nom afin de filtrer la recherche.

StartBrowsingAt.s :

Par défaut la recherche s'éffectue dans l'abre complet du fichie XML chargé, depuis le root, (**/**), mais il est aussi possible de spécifier un emplacement de départ.

Structure(s) associée(s) :

madlib_XMLNodes

Exemple :

```
NewMap titi.madlib_XMLNodes()

GetXMLBaseNodeList(id,titi())

ForEach titi()
   Debug "Nom => " + titi()\Name$
   Debug "Contenu =>" + titi()\Content$
Next
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_GetXMLBaseNodeAttributes ()

Description :

Remplis le dictionnaire avec les attributs du noeud.

La clef du dictionnaire est le nom de l'attribut, son contenu, la valeur de celui-ci.

Syntaxe :

Resultat.b = madlib_GetXMLBaseNodeAttributes(*IDBaseXML,NodeID.l,MapNodeAttributes.s())

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

Nombre d'attributs listés.

Paramètre(s) & Option(s) :

*IDBaseXML :

Identifiant provenant de la fonction madlib_FileXML.

NodeID.l :

Identifiant du noeud. Cette valeur peut être récuperé depuis la fonction madlib_GetXMLBaseNodeList.

NodeAttributes.s() :

Liste chaîné qui contient les attributs du noeud demandé.

Structure(s) associée(s) :

-

Exemple :

```
NewMap toto.s()

GetXMLBaseNodeAttributes(Id,idnoeud,toto())

ForEach toto()
   Debug MapKey(toto()) +" => "+toto()
Next
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_GetXMLBaseNodeID

madlib_GetXMLBaseNodeID()

Description :

Remplis le dictionnaire avec les attributs du noeud.

La clef du dictionnaire est le nom de l'attribut, son contenu, la valeur de celui-ci.

Syntaxe :

```
Resultat.b = madlib_GetXMLBaseNodeID(*IDBaseXML,Location$)
```

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

#True

Paramètre(s) & Option(s) :

*IDBaseXML :

Identifiant provenant de la fonction madlib_FileXML.

NodeID.l :

Identifiant du noeud. Cette valeur peut être récuperé depuis la fonction madlib_GetXMLBaseNodeList.

NodeAttributes.s() :

Liste chaîné qui contient les attributs du noeud demandé.

Structure(s) associée(s) :

-

Exemple :

```
idnoeud = GetXMLBaseNodeID(id,"/")
Debug idnoeud
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_GetXMLBaseNodeName()

Description :

Récupere le nom du noeud.

Syntaxe :

Resultat.s = madlib_GetXMLBaseNodeName(*IDBaseXML,NodeID.l)

Résultat :

Si l'opération à échouée ou n'existe pas :

Retour vide

Si l'opération est réussi :

Si un nom est spécifié alors retour du nom.

Paramètre(s) & Option(s) :

*IDBaseXML :

Identifiant provenant de la fonction madlib_FileXML.

NodeID.l :

Identifiant du noeud. Cette valeur peut être récuperé depuis la fonction madlib_GetXMLBaseNodeList.

Structure(s) associée(s) :

-

Exemple :

```
Debug GetXMLBaseNodeName(Id,idnoeud)
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_GetXMLBaseNodeInfos

madlib_GetXMLBaseNodeInfos()

Description :

Donne des information sur le noeud (le nombre de sous-noeud, son ID, le type de noeud, et le parent).

Syntaxe :

Resultat.b =

madlib_GetXMLBaseNodeInfos(*IDBaseXML,NodeID.l,*NodeInfos.madlib_XMLNodesInfos)

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

#True

Paramètre(s) & Option(s) :

*IDBaseXML :

Identifiant provenant de la fonction madlib_FileXML.

NodeID.l :

Identifiant du noeud. Cette valeur peut être récuperé depuis la fonction madlib_GetXMLBaseNodeList.

*NodeInfos.madlib_XMLNodesInfos :

Structure XML contenant les résultats.

Structure(s) associée(s) :

madlib_XMLNodesInfos

Exemple :

```
GetXMLBaseNodeInfos(id,titi()\NodeID,@t)
   Debug "Count :"
   Debug t\CountSubNode
   Debug "Parent Node ID :"
   Debug t\ParentNodeID
   Debug "Type :"
   Debug t\TypeContent
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_AddNodeEntry()

Description :

Ajoute un noeud XML dans la base.

Syntaxe :

```
Resultat.b =
madlib_AddNodeEntry(*IDXMLCatalog,NodeID.l,Location$,NodeName.s="",Co
ntent.s="",EntryType.b=#PB_XML_Normal)
```

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

#True

Paramètre(s) & Option(s) :

*IDXMLCatalog :

Identifiant provenant de la fonction madlib_LoadFileXML ou madlib_Create.

NodeID.l :

Identifiant du noeud. Cette valeur peut être récuperé depuis la fonction

madlib_GetXMLBaseNodeList.

Location$:

Emplacement de la balise.

NodeName.s :

Nom du noeud.

Content.s :

Nom du noeud.

EntryType.b :

Type de noeud, peut avoir comme valeur :

#PB_XML_Normal (Par défaut)

#PB_XML_CData

#PB_XML_Comment

#PB_XML_Instruction

Structure(s) associée(s) :

-

Exemple :

```
Debug
AddNodeEntry(*IDBaseXML,*NoeudCourant,XMLNodePath(*NoeudCourant
),GetXMLNodeName(*NoeudCourant),GetXMLNodeText(*NoeudCourant),#
PB_XML_Instruction)
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_AddNodeAttribute

madlib_AddNodeAttribute()

Description :

Ajoute un attribut à un noeud

Syntaxe :

```
Resultat.b =
madlib_AddNodeAttribute(*IDXMLCatalog,NodeID.l,Name$,Value.s="")
```

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

#True

Paramètre(s) & Option(s) :

*IDXMLCatalog :

Identifiant provenant de la fonction madlib_LoadFileXML ou madlib_CreateXMLCatalog.

NodeID.l :

Identifiant du noeud. Cette valeur peut être récuperé depuis la fonction madlib_GetXMLBaseNodeList.

Name.s :

Nom de l'attribut.

Value.s :

Valeur à associer à l'attribut.

Structure(s) associée(s) :

-

Exemple :

```
Debug
AddNodeAttribute(*IDBaseXML,*NoeudCourant,"MonNouveauAttribut",
"MaValeur")
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_AddNodeInfo()

Description :

Ajoute des informations sur un noeud.

Syntaxe :

Resultat.b = madlib_AddNodeInfo(*IDXMLCatalog,NodeID.l,CountSubNode.l,Type.b=#PB_XML_Normal,*ParentNodeID=0)

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

#True

Paramètre(s) & Option(s) :

*IDXMLCatalog :

Identifiant provenant de la fonction madlib_LoadFileXML ou madlib_CreateXMLCatalog.

NodeID.l :

Identifiant du noeud. Cette valeur peut être récuperé depuis la fonction madlib_GetXMLBaseNodeList.

CountSubNode.l :

Nombre de sous-noeud.

Type.b :

Type de noeud, peut avoir comme valeur :

#PB_XML_Normal (Par défaut)

#PB_XML_CData

#PB_XML_Comment

#PB_XML_Instruction

Structure(s) associée(s) :

-

Exemple :

```
-
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_DelNodeEntry

madlib_DelNodeEntry()

Description :

Supprime un noeud.

Syntaxe :

Resultat.b = madlib_DelNodeEntry(*IDXMLCatalog,NodeID.l)

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

#True

Paramètre(s) & Option(s) :

*IDXMLCatalog :

Identifiant provenant de la fonction madlib_LoadFileXML ou madlib_CreateXMLCatalog.

NodeID.l :

Identifiant du noeud. Cette valeur peut être récuperé depuis la fonction madlib_GetXMLBaseNodeList.

Structure(s) associée(s) :

-

Exemple :

```
Debug
AddNodeAttribute(*IDBaseXML,*NoeudCourant,"MonNouveauAttribut",
"MaValeur")
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_DelNodeInfos()

Description :

Supprime les information du noeud spécifié.

Syntaxe :

```
Resultat.b = madlib_DelNodeInfos(*IDXMLCatalog,NodeID.l)
```

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

#True

Paramètre(s) & Option(s) :

*IDXMLCatalog :

Identifiant provenant de la fonction madlib_LoadFileXML ou madlib_CreateXMLCatalog.

NodeID.l :

Identifiant du noeud. Cette valeur peut être récuperé depuis la fonction madlib_GetXMLBaseNodeList.

Structure(s) associée(s) :

-

Exemple :

```
-
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_DelNodeAttributes

madlib_DelNodeAttributes()

Description :

Supprime tout les attributs du noeud spécifié.

Syntaxe :

Resultat.b = madlib_DelNodeAttributes(*IDXMLCatalog,NodeID.l)

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

#True

Paramètre(s) & Option(s) :

*IDXMLCatalog :

Identifiant provenant de la fonction madlib_LoadFileXML ou madlib_CreateXMLCatalog.

NodeID.l :

Identifiant du noeud. Cette valeur peut être récuperé depuis la fonction madlib_GetXMLBaseNodeList.

Structure(s) associée(s) :

-

Exemple :

```
-
```

OS Supportés (testé) : **OS Compatibles (non testé) :**

madlib_TextErrorXML

madlib_TextErrorXML()

Description :

Interprète le code erreur retourné par les fonctions de chargement XML en une explication texte.

Syntaxe :

Resultat.s = madlib_TextErrorXML(IDFichierXML.i)

Résultat :

Texte indiquant l'erreur.

Paramètre(s) & Option(s) :

IDFichierXML.i :

Identifiant provenant de la fonction madlib_LoadFileXML ou madlib_CreateXMLCatalog.

Structure(s) associée(s) :

-

Exemple :

-

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_CountXMLBaseSubNodes()

Description :

Indique le nombre de sous-noeud d'un noeud parent.

Syntaxe :

Resultat.i = madlib_CountXMLBaseSubNodes(*IDBaseXML,NodeID.l)

Résultat :

Nombre de sous noeud

Paramètre(s) & Option(s) :

*IDXMLCatalog :

Identifiant provenant de la fonction madlib_LoadFileXML ou madlib_CreateXMLCatalog.

NodeID.l :

Identifiant du noeud. Cette valeur peut être récuperé depuis la fonction madlib_GetXMLBaseNodeList.

Structure(s) associée(s) :

-

Exemple :

```
-
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_GetXMLBaseParentNodeID

madlib_GetXMLBaseParentNodeID()

Description :

Retourne le numéro du noeud parent a celui spécifié.

Syntaxe :

Resultat.l = madlib_GetXMLBaseParentNodeID(*IDBaseXML,NodeID.l)

Résultat :

Numéro du noeud parent, ou 0 si pas de noeud parent.

Paramètre(s) & Option(s) :

*IDBaseXML :

Identifiant provenant de la fonction madlib_LoadFileXML ou madlib_CreateXMLCatalog.

NodeID.l :

Identifiant du noeud. Cette valeur peut être récuperé depuis la fonction madlib_GetXMLBaseNodeList.

Structure(s) associée(s) :

-

Exemple :

```
-
```

OS Supportés (testé) :

OS Compatibles (non testé) :

Windows

Fonctions pour Windows

Toute les fonctions que vous verrez sous cette catégorie ne sont utilisable que sous Microsoft Windows.

La majeur des fonctions décrites sont implémenté avec les WinAPI de Microsoft.

Création / Gestions des services Windows

Dans cette partie, sont rangés les outils pour créer, supprimer, contrôler les services Windows.

Voici un exemple générale d'implémentation et d'utilisation d'un programme PureBasic en service Windows.

 Il est vivement conseillé d'utiliser la gestion multi-thread lors de la compilation.

Exemple :

```
NewMap LDC.s()
Define ServiceApplication.madlib_ConfigService
Define stop


;
;Configuration du service.
;Différente options qui ne sont pas forcément obligatoire
;

   ServiceApplication\CommandLineParametre$ = "-service"
   ServiceApplication\Name$ = "" ; Si laissé vide alors dans
l'initialisation il sera automatiquement calculé
```

```
    ServiceApplication\DisplayName$ = "Test du service Purebasic"
    ServiceApplication\Description$ = "Ceci est la description du
service purebasic."
    ServiceApplication\Journal\CheminFichier$ = "Install_Service.log"
;Il est possible de spécifier un fichier journal pour l'arrêt
l'installation la suppression le démarrage du service.
    ServiceApplication\StartType = 0 ; Faire son choix entre =>
#SERVICE_AUTO_START, #SERVICE_BOOT_START, #SERVICE_DEMAND_START,
#SERVICE_DISABLED, #SERVICE_SYSTEM_START
    ServiceApplication\ServiceType = #SERVICE_WIN32_OWN_PROCESS|
#SERVICE_INTERACTIVE_PROCESS ;Faire son choix entre =>
#SERVICE_WIN32_OWN_PROCESS (avec l'option :
#SERVICE_INTERACTIVE_PROCESS)
    ;
                    #SERVICE_FILE_SYSTEM_DRIVER
    ;
                    #SERVICE_KERNEL_DRIVER
    ;
                    #SERVICE_WIN32_SHARE_PROCESS (avec l'option :
#SERVICE_INTERACTIVE_PROCESS)
    ServiceApplication\ErrorControlType = 0 ; Faire son choix entre =>
#SERVICE_ERROR_CRITICAL, #SERVICE_ERROR_IGNORE, #SERVICE_ERROR_NORMAL,
#SERVICE_ERROR_SEVERE
    ServiceApplication\ServiceStatus\dwControlsAccepted =
#SERVICE_ACCEPT_STOP|#SERVICE_ACCEPT_SHUTDOWN ; Faire son Choix =>
#SERVICE_ACCEPT_STOP | #SERVICE_ACCEPT_SHUTDOWN |
#SERVICE_ACCEPT_PAUSE_CONTINUE ;
    ServiceApplication\StartProgram\WorkingDirectory$ = madlib_ProgDir()

    ServiceApplication\StartProgram\ProgramLocation$ = ProgramFilename()
    ServiceApplication\StartProgram\Parameters$ = "-service ok"


;Récupération de la ligne de commmande
madlib_ChargeParametreCMD(LDC())


;Vérification si des paramètres sont demandés pour l'installation ou
suppression
If LDC("install") = Chr(2)

    madlib_InstallService(@ServiceApplication)

ElseIf LDC("remove") = Chr(2)
    madlib_DeleteService(@ServiceApplication)
    End
EndIf

;Vérification si demande de démarrage
If LDC("start") = Chr(2)
    madlib_StartService(@ServiceApplication)
    End
EndIf


;Arrêt du service.
If LDC("stop") = Chr(2)
    madlib_StopService(@ServiceApplication)
    End
EndIf


;Démarrage par défaut
```

```
If LDC("service") = Chr(2)
   MessageRequester("Test Service Purebasic","C'est parti ...")
   ServiceApplication\Journal\CheminFichier$ = "start_service.log"
   ServiceApplication\Journal\ArchiveMax = 5
   ServiceApplication\Journal\TailleFichierMax = 1024
   ServiceApplication\StartProgram\ForceRunning = #True
   madlib_Journal(@ServiceApplication\Journal,"Starting
service...","NIV1")
   madlib_InitServiceStarting(@ServiceApplication)
ElseIf LDC("service") = "ok"


   Repeat
            Delay(3000)
            If MessageRequester("Test Service Purebasic", "Le service
tourne et je suis lancé en mode startprogram"+#CRLF$+"Cliquer sur NON,
ne fera pas quitter l'application, car j'ai un système de protection,
que l'on peut désactiver.", #PB_MessageRequester_YesNo) =
#PB_MessageRequester_No
                     stop = #True
            EndIf
   Until stop = #True

   MessageRequester("Test Service Purebasic","Je quitte, aurevoir !")
   End 0
EndIf
```


madlib_InstallService

madlib_InstallService()

Description :

Installation du service avec la valeur du membre *name$*

Syntaxe :

```
Resultat = madlib_InstallService(*Service.madlib_ConfigService)
```

Structure(s) associée(s) :

- madlib_ConfigService
- madlib_ConfigJournal

- madlib_Thread
- madlib_ConfigProgram

Résultat :

Si l'opération à échouée :

#False : Erreur interne de la fonction, ou mauvais paramètres.

Valeur négative : Code retour système. Utiliser la fonction madllib_ReverseReturnCodei pour les interpréter.

Si l'opération est réussi :

#True

Exemple :

Voir exemple général.

OS Supportés (testé) :

madlib_DeleteService

madlib_DeleteService()

Description :

Supprime le service avec le nom contenu dans la valeur du membre Name$

Syntaxe :

```
Resultat = madlib_RemoveService(*Service.madlib_ConfigService)
```

Structure(s) associée(s) :

- madlib_ConfigService
- madlib_ConfigJournal
- madlib_Thread
- madlib_ConfigProgram

Résultat :

Si l'opération à échouée :

#False : Erreur interne de la fonction, ou mauvais paramètres.

Valeur négative : Code retour système. Utiliser la fonction madllib_ReverseReturnCodei pour les interprêter.

Si l'opération est réussi :

#True

Exemple :

Voir exemple général.

OS Supportés (testé) :

madlib_InitServiceStarting

madlib_InitServiceStarting()

Description :

Créer l'instance qu'attend en retour le gestionnaire de service Windows quand on lance l'application.

Syntaxe :

Resultat.i = madlib_InitServiceStarting(*Service.madlib_ConfigService)

Structure(s) associée(s) :

- madlib_ConfigService
- madlib_ConfigJournal
- madlib_Thread
- madlib_ConfigProgram

Résultat :

Si l'opération à échouée :

#False : Erreur interne de la fonction, ou mauvais paramètres.

Valeur négative : Code retour système. Utiliser la fonction madllib_ReverseReturnCodei pour les interprêter.

Si l'opération est réussi :

#True

Exemple :

Voir exemple général.

OS Supportés (testé) :

madlib_StartService

madlib_StartService()

Description :

Lance le démarrage du service nommée dans la valeur du membre Name$

Syntaxe :

Resultat.i = madlib_StartService(*Service.madlib_ConfigService)

Structure(s) associée(s) :

- madlib_ConfigService
- madlib_ConfigJournal
- madlib_Thread
- madlib_ConfigProgram

Résultat :

Si l'opération à échouée :

#False : Erreur interne de la fonction, ou mauvais paramètres.

Valeur négative : Code retour système. Utiliser la fonction madllib_ReverseReturnCodei pour les interpréter.

Si l'opération est réussi :

#True

Exemple :

Voir exemple général.

OS Supportés (testé) :

madlib_StopService

madlib_StopService()

Description :

Lance la procédure d'arrêt du service nommée dans la valeur du membre Name$

Syntaxe :

```
Resultat.i = madlib_StopService(*Service.madlib_ConfigService)
```

Structure(s) associée(s) :

- madlib_ConfigService
- madlib_ConfigJournal
- madlib_Thread
- madlib_ConfigProgram

Résultat :

Si l'opération à échouée :

#False : Erreur interne de la fonction, ou mauvais paramètres.

Valeur négative : Code retour système. Utiliser la fonction madllib_ReverseReturnCodei pour les interpréter.

Si l'opération est réussi :

#True

Exemple :

Voir exemple général.

OS Supportés (testé) :

madlib_StopService()

Description :

Initie au gestionnaire de contrôle de service, une demande

Les différentes options de l'option **InstructionControl** :

- #SERVICE_CONTROL_CONTINUE
- #SERVICE_CONTROL_STOP
- #SERVICE_CONTROL_INTERROGATE
- #SERVICE_CONTROL_PAUSE
- #SERVICE_CONTROL_SHUTDOWN

Syntaxe :

```
Resultat.i = madlib_ControlService(*Service.madlib_ConfigService,InstructionControl=#SERVICE_CONTROL_STOP)
```

Structure(s) associée(s) :

- madlib_ConfigService
- madlib_ConfigJournal
- madlib_Thread
- madlib_ConfigProgram

Résultat :

Si l'opération à échouée :

#False : Erreur interne de la fonction, ou mauvais paramètres.

Valeur négative : Code retour système. Utiliser la fonction madllib_ReverseReturnCodei pour les interpréter.

Si l'opération est réussi :

#True

Exemple :

```
madlib_ControlService(*Service,#SERVICE_CONTROL_STOP)
```

OS Supportés (testé) :

madlib_PauseService

madlib_PauseService()

Description :

Demande de pause auprès au gestionnaire de services.

Syntaxe :

Resultat.i = madlib_PauseService(*Service.madlib_ConfigService)

Structure(s) associée(s) :

- madlib_ConfigService
- madlib_ConfigJournal
- madlib_Thread
- madlib_ConfigProgram

Résultat :

Si l'opération à échouée :

#False : Erreur interne de la fonction, ou mauvais paramètres.

Valeur négative : Code retour système. Utiliser la fonction madllib_ReverseReturnCodei pour les interpréter.

Si l'opération est réussi :

#True

Exemple :

Voir exemple général.

OS Supportés (testé) :

madlib_ContinueService

madlib_ContinueService()

Description :

Demande de relance après une pause au gestionnaire de services.

Syntaxe :

```
Resultat.i = madlib_ContinueService(*Service.madlib_ConfigService)
```

Structure(s) associée(s) :

- madlib_ConfigService
- madlib_ConfigJournal
- madlib_Thread
- madlib_ConfigProgram

Résultat :

Si l'opération à échouée :

#False : Erreur interne de la fonction, ou mauvais paramètres.

Valeur négative : Code retour système. Utiliser la fonction madllib_ReverseReturnCodei pour les interprêter.

Si l'opération est réussi :

#True

Exemple :

Voir exemple général.

OS Supportés (testé) :

Gestion du registre Windows

Enssemble de fonction servant à manipuler l'environement base de registre.

Pour tout les fonctions de cette catégories, il est possible d'utiliser madlib_RegParsePath afin de récupérer les contexte hKey, SubKey$ et ComputerName$.

 Il est impossible de passer des variables de type chaines de caractère de longueur fixe. Utiliser MaChaine$ ou Machine.s

madlib_RegParsePath()

Description :

Analyse, découpe et interprête, la valeur contenu dans *Location$* afin de preparer les valeurs à donner pour les autres fonctions de cette catégorie.

Syntaxe :

Resultat.b = madlib_RegParsePath(Location$,*RegPath.madlib_RegPathValue)

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

#True

La structure *RegPath* sera remplis correctement.

Structure(s) associée(s) :

madlib_RegPathValue

Exemple :

```
Define essaiparse.madlib_RegPathValue

Debug RegParsePath("\\MAmachine\HKLM\Essai1\Toto",@essaiparse)

Debug essaiparse\ComputerName$
Debug essaiparse\hKey
```

```
Debug essaiparse\SubKey$

Debug RegParsePath("\\MAmachine\HKEY_LOCAL_MACHINE\loulou
\lili",@essaiparse)

Debug essaiparse\ComputerName$
Debug essaiparse\hKey
Debug essaiparse\SubKey$

Debug RegParsePath("\\MAmachine\HKCU\essai\essai",@essaiparse)

Debug essaiparse\ComputerName$
Debug essaiparse\hKey
Debug essaiparse\SubKey$

Debug RegParsePath(".\HKCU\essai\essai",@essaiparse)

Debug essaiparse\ComputerName$
Debug essaiparse\hKey
Debug essaiparse\SubKey$

Debug RegParsePath("HKCU\essai\essai",@essaiparse)

Debug essaiparse\ComputerName$
Debug essaiparse\hKey
Debug essaiparse\SubKey$
```

OS Supportés (testé) :

madlib_RegConnect

madlib_RegConnect()

Description :

Detecte la connexion de la base de registre local ou permet de se connecter à une base de registre distante.

Syntaxe :

```
Resultat.q = madlib_RegConnect(hKey.q,ComputerName$=".")
```

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

La connexion sera établis avec la machine ou le Hkey sera renvoyé dans la situation d'une connexion locale.

Structure(s) associée(s) :

madlib_RegPathValue

Paramètre(s) & Option(s) :

Exemple :

```
Debug RegConnect(#HKEY_CURRENT_USER,"MaMachineDistante")
Debug RegConnect(#HKEY_CURRENT_USER)
```

OS Supportés (testé) :

madlib_RegOpen

madlib_RegOpen()

Description :

Ouvre une clef dans la base de registre. *IDOuvertureBDR.q* Accepte des Retour de *madlib_RegConnect* ou les constantes de type *#HKEY_*.

Syntaxe :

```
Resultat.l = madlib_RegOpen(IDOuvertureBDR.l,SubKey$,Reg64.b=#False)
```

Paramètre(s) & Option(s) :

IDOuvertureBDR :

ID de retour de madlib_RegConnect.

SubKey$:

Chemin pour le chemin.

Reg64 :

IsWin64 permet d'indiquer si l'accès doit se faire avec l'abstraction de 32 à 64.

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

Handle de l'ouverture de la clef.

Fonction pouvant être associée(s) :

madlib_IsWin64

Exemple :

```
Debug madlib_RegOpen(#HKEY_CURRENT_USER,"Software\Test")

Debug
madlib_RegOpen(madlib_RegConnect(#HKEY_CURRENT_USER,"MaMachineD
istante"),"Software\Test")
```

OS Supportés (testé) :

madlib_RegCreate

madlib_RegCreate()

Description :

Créer une clef dans la base de registre. Le mode temporaire permet de créer des clefs qui ne seront par sauvegardé une fois reconnecté a la session utilisateur ou au redémarrage de la machine


Attention il sera necessaire de fermer la clef crée avec cette fonction.

Syntaxe :

```
Resultat.l = madlib_RegCreate(hKey.q,SubKey$,ComputerName$=".",Reg64.b=#False,Mode Temporaire=#REG_OPTION_NON_VOLATILE)
```

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

Handle de clef nouvellement crée.

Paramètre(s) & Option(s) :

hKey :

#HKEY_LOCAL_MACHINE

#HKEY_CURRENT_USER

#HKEY_CLASSES_ROOT

#HKEY_USERS

#HKEY_CURRENT_CONFIG

SubKey$:

Chemin pour l'emplacement de la valeur.

Reg64 :

IsWin64 permet d'indiquer si l'accès doit se faire avec l'abstraction de 32 à 64.

Structure(s) associée(s) :

madlib_RegPathValue

Exemple :

```
Debug RegCreate(#HKEY_CURRENT_USER,"Software\Test\Test2")
```

OS Supportés (testé) :

madlib_RegWriteMultiSZ

madlib_RegWriteMultiSZ()

Description :

Utilise la liste *ItemsContentValue* pour le placer du contenu dans la valeur Reg_multi_sz.

Syntaxe :

Resultat.i = madlib_RegWriteMultiSZ(hKey.q,SubKey$,NameValue$,List ItemsContentValue.s(),ComputerName$=".",Reg64.b=#False)

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

#True

Paramètre(s) & Option(s) :

hKey :

#HKEY_LOCAL_MACHINE

#HKEY_CURRENT_USER

#HKEY_CLASSES_ROOT

#HKEY_USERS

#HKEY_CURRENT_CONFIG

SubKey$:

Chemin pour l'emplacement de la valeur.

NameValue$:

Nom de la valeur

ItemsContentValue.s() :

Liste chaîné qui contiendra les valeurs du multiSZ

Reg64 :

IsWin64 permet d'indiquer si l'accès doit se faire avec l'abstraction de 32 à 64.

Exemple :

```
NewList ListeRegSZ.s()

Debug madlib_RegWriteMultiSZ(#HKEY_CURRENT_USER,"SOFTWARE
\Test","TestSZ",ListeRegSZ())
```

OS Supportés (testé) :

madlib_RegReadMultiSZ

madlib_RegReadMultiSZ()

Description :

Charge le contenu de la Reg_mutli_sz, dans la liste (*ListeRegSz()*) fournis en paramètre.

Syntaxe :

```
Resultat.i = madlib_RegReadMultiSZ(hKey.q,SubKey$,NameValue$,List
ItemsContentValue.s(),ComputerName$=".",Reg64.b=#False)
```

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

#True

Paramètre(s) & Option(s) :

hKey :

#HKEY_LOCAL_MACHINE

#HKEY_CURRENT_USER

#HKEY_CLASSES_ROOT

#HKEY_USERS

#HKEY_CURRENT_CONFIG

SubKey$:

Chemin pour l'emplacement de la valeur.

NameValue$:

Nom de la valeur

ItemsContentValue.s() :

Liste chaîné qui contiendra les valeurs du multiSZ

Reg64 :

IsWin64 permet d'indiquer si l'accès doit se faire avec l'abstraction de 32 à 64.

Exemple :

```
NewList ListeRegSZ.s()

Debug madlib_RegReadMultiSZ(#HKEY_CURRENT_USER,"SOFTWARE
\Test","TestSZ",ListeRegSZ())
```

OS Supportés (testé) :

madlib_RegWrite

madlib_RegWrite()

Description :

Ecrit tout type de valeurs dans la base de registre.

- #REG_SZ

- #REG_EXPAND_SZ
- #REG_MULTI_SZ (Attention le contenu passé en paramètre ne peut être découpé en plusieurs item.
- #REG_DWORD
- #REG_QWORD
- #REG_BINARY

Syntaxe :

```
Resultat.i = madlib_RegWrite(hKey.q,SubKey$,TypeValue.i=#REG_SZ,NameValue$="",Value$="",ComputerName$=".",Reg64.b=#False)
```

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

#True

Paramètre(s) & Option(s) :

hKey :

#HKEY_LOCAL_MACHINE

#HKEY_CURRENT_USER

#HKEY_CLASSES_ROOT

#HKEY_USERS

#HKEY_CURRENT_CONFIG

SubKey$:

Chemin pour l'emplacement de la valeur.

NameValue$:

Nom de la valeur

ItemsContentValue.s() :

Liste chaîné qui contiendra les valeurs du multiSZ

Reg64 :

IsWin64 permet d'indiquer si l'accès doit se faire avec l'abstraction de 32 à 64.

Exemple :

```
Debug RegWrite(#HKEY_CURRENT_USER,"Software
\Test",#REG_SZ,"Coucou","Test")

Debug RegWrite(#HKEY_CURRENT_USER,"Software
\Test",#REG_BINARY,"Coucou2","0003022915")

Debug RegWrite(#HKEY_CURRENT_USER,"Software
\Test",#REG_DWORD,"Coucou3","15")

Debug RegWrite(#HKEY_CURRENT_USER,"Software
\Test",#REG_QWORD,"Coucou4","15")

Debug RegWrite(#HKEY_CURRENT_USER,"Software
\Test",#REG_MULTI_SZ,"Coucou5","Test")
```

OS Supportés (testé) :

madlib_RegReadType

madlib_RegReadType()

Description :

Retoune le type de la valeur :

- #REG_SZ
- #REG_EXPAND_SZ

- #REG_MULTI_SZ
- #REG_DWORD
- #REG_QWORD
- #REG_BINARY

Syntaxe :

```
Resultat.i = madlib_RegReadType(hKey.q,SubKey$,NameValue$,ComputerName$=".",Reg64.b=#False)
```

Paramètre(s) & Option(s) :

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

Le type de la valeur.

Paramètre(s) & Option(s) :

hKey :

#HKEY_LOCAL_MACHINE

#HKEY_CURRENT_USER

#HKEY_CLASSES_ROOT

#HKEY_USERS

#HKEY_CURRENT_CONFIG

SubKey$:

Chemin pour l'emplacement de la valeur.

NameValue$:

Nom de la valeur

ComputerName$:

Nom de la machine distante sinon "." par défaut.

Reg64 :

IsWin64 permet d'indiquer si l'accès doit se faire avec l'abstraction de 32 à 64.

Exemple :

```
Debug madlib_RegReadType(#HKEY_CURRENT_USER,"Software
\Test","Coucou1")
```

OS Supportés (testé) :

madlib_RegRead()

Description :

Retoune les types valeurs :

- #REG_SZ
- #REG_EXPAND_SZ
- #REG_MULTI_SZ (retour du contenu de la valeur séparé par un espace entre chaque ligne)
- #REG_DWORD
- #REG_QWORD
- #REG_BINARY

Syntaxe :

```
Resultat.s = madlib_RegRead(hKey.q,SubKey$,NameValue$="",ComputerName$=".",Reg64.b=#False)
```

Résultat :

Si l'opération à échouée :

Retour vide.

Si l'opération est réussi :

Le contenu de la valeur.

Paramètre(s) & Option(s) :

hKey :

#HKEY_LOCAL_MACHINE

#HKEY_CURRENT_USER

#HKEY_CLASSES_ROOT

#HKEY_USERS

#HKEY_CURRENT_CONFIG

SubKey$:

Chemin pour l'emplacement de la valeur.

NameValue$:

Nom de la valeur

ComputerName$:

Nom de la machine distante sinon "." par défaut.

Reg64 :

IsWin64 permet d'indiquer si l'accès doit se faire avec l'abstraction de 32 à 64.

Exemple :

```
Debug RegRead(#HKEY_CURRENT_USER,"Software\Test","Coucou")
Debug RegRead(#HKEY_CURRENT_USER,"Software\Test","Coucou2")
Debug RegRead(#HKEY_CURRENT_USER,"Software\Test","Coucou3")
Debug RegRead(#HKEY_CURRENT_USER,"Software\Test","Coucou4")
```

OS Supportés (testé) :

madlib_RegExist

madlib_RegExist()

Description :

Test les clefs ou les valeurs dans la base de registre.

Syntaxe :

Resultat.b = madlib_RegExist(hKey.q,SubKey$,NameValue$="",ComputerName$=".",Reg64.b=#False)

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

#True

Paramètre(s) & Option(s) :

hKey :

#HKEY_LOCAL_MACHINE

#HKEY_CURRENT_USER

#HKEY_CLASSES_ROOT

#HKEY_USERS

#HKEY_CURRENT_CONFIG

SubKey$:

Chemin pour l'emplacement de la valeur.

NameValue$:

Nom de la valeur

ComputerName$:

Nom de la machine distante sinon "." par défaut.

Reg64 :

IsWin64 permet d'indiquer si l'accès doit se faire avec l'abstraction de 32 à 64.

Exemple :

```
Debug RegExist(#HKEY_CURRENT_USER,"Software
\Test","testadistance")
```

OS Supportés (testé) :

madlib_RegEnumValue

madlib_RegParsePath()

Description :

Permet le parcours de valeur sous une clef de base de registre. Si il n'existe plus de valeur le retour sera vide.

Syntaxe :

```
Resultat.s = madlib_RegEnumValue(hKey.q,SubKey$,Index,
ComputerName$=".",Reg64.b=#False)
```

Résultat :

Si l'opération à échouée :

vide

Si l'opération est réussi :

retour du nom de la valeur.

Paramètre(s) & Option(s) :

hKey :

#HKEY_LOCAL_MACHINE

#HKEY_CURRENT_USER

#HKEY_CLASSES_ROOT

#HKEY_USERS

#HKEY_CURRENT_CONFIG

SubKey$:

Chemin pour l'emplacement de la valeur.

Index :

Permet le parcours de la branche des portes

ComputerName$:

Nom de la machine distante sinon "." par défaut.

Reg64 :

IsWin64 permet d'indiquer si l'accès doit se faire avec l'abstraction de 32 à 64.

Exemple :

```
Repeat

          Clef$ =
madlib_RegEnumKey(#HKEY_LOCAL_MACHINE,"SOFTWARE\Microsoft
\Windows\CurrentVersion\Uninstall",ind,".",1)

          If Clef$ <> ""
           Debug "---- "+Clef$

           EmplacementValeurClef$ = "SOFTWARE\Microsoft\Windows
\CurrentVersion\Uninstall\"+Clef$
           ind_contenu = 0
           Repeat
                   ValeurClef$ =
madlib_RegEnumValue(#HKEY_LOCAL_MACHINE,EmplacementValeurClef$,
ind_contenu,".",1)
                   Debug ValeurClef$+" = "+
madlib_RegRead(#HKEY_LOCAL_MACHINE,EmplacementValeurClef$,Valeu
rClef$,".",1)
                   ind_contenu +1
           Until ValeurClef$ = ""

         EndIf

         ind + 1
   Until Clef$ = ""
```

OS Supportés (testé) :

madlib_RegEnumKey

madlib_RegEnumKey()

Description :

Permet le parcours de clef sous une clef défini de base de registre. Si il n'existe plus de valeur le retour sera vide.

Syntaxe :

```
Resultat.s = madlib_RegEnumKey(hKey.q,SubKey$,Index,
ComputerName$=".",Reg64.b=#False)
```

Résultat :

Si l'opération à échouée :

vide

Si l'opération est réussi :

retour du nom de la valeur.

Paramètre(s) & Option(s) :

hKey :

#HKEY_LOCAL_MACHINE

#HKEY_CURRENT_USER

#HKEY_CLASSES_ROOT

#HKEY_USERS

#HKEY_CURRENT_CONFIG

SubKey$:

Chemin pour l'emplacement de la valeur.

Index :

Permet le parcours de la branche dans la base de registre.

ComputerName$:

Nom de la machine distante sinon "." par défaut.

Reg64 :

IsWin64 permet d'indiquer si l'accès doit se faire avec l'abstraction de 32 à 64.

Exemple :

```
Repeat
```

```
   ValeurClef$ =
madlib_RegEnumKey(#HKEY_LOCAL_MACHINE,"SOFTWARE",ind)
   Debug ValeurClef$
   ind + 1
Until ValeurClef$ = ""
```

OS Supportés (testé) :

madlib_RegDeleteValue

madlib_RegDeleteValue()

Description :

Supprime les valeurs dans la base de registre.

Syntaxe :

Resultat.b = madlib_RegDeleteValue(hKey.q,SubKey$,NameValue$="",ComputerName$="." ,Reg64.b=#False)

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

#True

Paramètre(s) & Option(s) :

hKey :

#HKEY_LOCAL_MACHINE

#HKEY_CURRENT_USER

#HKEY_CLASSES_ROOT

#HKEY_USERS

#HKEY_CURRENT_CONFIG

SubKey$:

Chemin pour l'emplacement de la valeur.

NameValue$:

Nom de la valeur

ComputerName$:

Nom de la machine distante sinon "." par défaut.

Reg64 :

IsWin64 permet d'indiquer si l'accès doit se faire avec l'abstraction de 32 à 64.

Exemple :

```
Debug RegDeleteValue(#HKEY_CURRENT_USER,"Software\Test\Nouvelle clé #1\Nouvelle clé #1","Valeur")
```

OS Supportés (testé) :

madlib_RegDeleteKey

madlib_RegDeleteKey()

Description :

Supprime les clefs dans la base de registre.

 La clef doit être impérativement vide (sans valeur).

Syntaxe :

```
Resultat.b =
madlib_RegDeleteKey(hKey.q,SubKey$,ComputerName$=".",Reg64.b=#False)
```

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

#True

Paramètre(s) & Option(s) :

hKey :

#HKEY_LOCAL_MACHINE

#HKEY_CURRENT_USER

#HKEY_CLASSES_ROOT

#HKEY_USERS

#HKEY_CURRENT_CONFIG

SubKey$:

Chemin pour l'emplacement de la valeur.

ComputerName$:

Nom de la machine distante sinon "." par défaut.

Reg64 :

IsWin64 permet d'indiquer si l'accès doit se faire avec l'abstraction de 32 à 64.

Exemple :

```
Debug RegDeleteKey(#HKEY_CURRENT_USER,"Software\Test\Nouvelle
clé #1\Nouvelle clé #1")
```

OS Supportés (testé) :

madlib_RegDeleteTreeKeyValue

madlib_RegDeleteTreeKeyValue()

Description :

Supprime toutes les valeurs et toutes les clefs.

Syntaxe :

Resultat.b = madlib_RegDeleteTreeKeyValue(hKey.q,SubKey$,ComputerName$,Reg64.b=#False)

Résultat :

Si l'opération à échouée :

#False

Si l'opération est réussi :

#True

Paramètre(s) & Option(s) :

hKey :

#HKEY_LOCAL_MACHINE

#HKEY_CURRENT_USER

#HKEY_CLASSES_ROOT

#HKEY_USERS

#HKEY_CURRENT_CONFIG

SubKey$:

Chemin pour l'emplacement de la valeur.

ComputerName$:

Nom de la machine distante sinon "." par défaut.

Reg64 :

IsWin64 permet d'indiquer si l'accès doit se faire avec l'abstraction de 32 à 64.

Exemple :

```
Debug RegDeleteTreeKeyValue(#HKEY_CURRENT_USER,"Software\Test
\Nouvelle clé #1\Nouvelle clé #1")
```

OS Supportés (testé) :

madlib_TextErrorWinWSA

madlib_TextErrorWinWSA()

Description :

Retourne la description en Français par rapport au code erreur.

Il n'est pas necessaire de retourner le code retour, il peut être inversé

(négatif).

Syntaxe :

Resultat.i = madlib_TextErrorWinWSA(Code.l)

Paramètres & options :

Code.l : Code retour de l'erreur, même négatif.

Resultat :

Texte expliquatif en Français du code erreur.

Exemple :

```
Debug madlib_TextErrorWinWSA (0)
Debug madlib_TextErrorWinWSA(11010)
Debug madlib_TextErrorWinWSA(1013)
```

OS Supportés (testé) :

OS non compatibles :

madlib_TextErrorWinIPStatus()

Description :

Retourne la description en Français par rapport au code erreur.

Il n'est pas necessaire de retourner le code retour, il peut être inversé (négatif).

Syntaxe :

Resultat.i = madlib_TextErrorWinIPStatus(Code.l)

Paramètres & options :

Code.l : Code retour de l'erreur, même négatif.

Resultat :

Texte expliquatif en Français du code erreur.

Exemple :

```
Debug madlib_TextErrorWinIPStatus(0)
Debug madlib_TextErrorWinIPStatus(11010)
Debug madlib_TextErrorWinIPStatus(1013)
```

OS Supportés (testé) :

OS non compatibles :

madlib_PingWin

madlib_PingWin()

Description :

Execute un ping Windows en utilisant les sockects Microsoft.

Il n'est possible de spécifier que des adresses IP. Pas de nom de machine.

Pour les erreurs, se referer sur le site Microsoft.

Syntaxe :

Resultat.i = madlib_PingWin(IP_Address$,TimeOutMs.i=5000,PacketSizeBytes.u=0)

Paramètres & options :

TimeOutMs.i : Délai maximum de la requête ICMP en millisecondes. Par défaut 5 secondes

PacketSizeBytes.u : Taille du paquet en octets qui sera envoyé à la machine ciblé. Par défaut un paquet de 32 octects.

Contantes associées :

Retour de la fonction :

#madlib_WINSTATUS_IP_BUF_TOO_SMALL

#madlib_WINSTATUS_IP_DEST_NET_UNREACHABLE

#madlib_WINSTATUS_IP_DEST_HOST_UNREACHABLE

#madlib_WINSTATUS_IP_DEST_PROT_UNREACHABLE

#madlib_WINSTATUS_IP_DEST_PORT_UNREACHABLE

#madlib_WINSTATUS_IP_NO_RESOURCES

#madlib_WINSTATUS_IP_BAD_OPTION

#madlib_WINSTATUS_IP_HW_ERROR

#madlib_WINSTATUS_IP_PACKET_TOO_BIG

#madlib_WINSTATUS_IP_REQ_TIMED_OUT

#madlib_WINSTATUS_IP_BAD_REQ

#madlib_WINSTATUS_IP_BAD_ROUTE

#madlib_WINSTATUS_IP_TTL_EXPIRED_TRANSIT

#madlib_WINSTATUS_IP_TTL_EXPIRED_REASSEM

#madlib_WINSTATUS_IP_PARAM_PROBLEM

#madlib_WINSTATUS_IP_SOURCE_QUENCH

#madlib_WINSTATUS_IP_OPTION_TOO_BIG

#madlib_WINSTATUS_IP_BAD_DESTINATION

#madlib_WINSTATUS_IP_GENERAL_FAILURE

Resultat :

En valeur positif est le retour en milliseconde du retour du ping.

En valeur négative est le retour en erreur (API Windows, erreur interne à la fonction, erreur de socket).

Exemple :

```
Debug madlib_PingWin(10.0.0.1)
```

OS Supportés (testé) :

OS non compatibles :

madlib_IsWin64b()

Description :

Indique sur l'environement est un 64bits ou non. Il peut, si les droits administrateur sont suffisant, intéroger la version d'une machine a traver le réseau.

Syntaxe :

Resultat.b = madlib_IsWin64(ComputerName$=".")

Resultat :

- #True si l'environement est en 64bits.
- #False si l'envirnoement n'est pas en 64bits.

Exemple :

```
Debug madlib_isWin64b()
```

OS Supportés (testé) :

OS non compatibles :

madlib_TextGetLastError()

Syntaxe :

Resultat$ = madlib_TextGetLastError(CodeErreur.l=#PB_Ignore)

Description :

Retourne le message associé à l'erreur Windows.

Si il n'y pas de code retour spécifié, la valeur de GetLasError, sera récuperé automatiquement.

Exemple :

```
Debug madlib_TextGetLastError()
```

```
Debug madlib_TextGetLastError(1313)
```

OS Supportés (testé) :

OS non compatibles :

madlib_WinGetFreeSpace()

Description :

Retourne les informations concernant l'espace disque, disponible et total.

Syntaxe :

Resultat.b = madlib_WinGetFreeSpace(*Config.madlib_ConfigDiskFreeSpace)

Structure(s) associée(s) :

- madlib_ConfigDiskFreeSpace

Exemple :

```
InfosDisque.madlib_ConfigDiskFreeSpace

toto\DiskLocation$ = "c:\"

If WinGetFreeSpace(@InfosDisque)
   Debug toto\FreeBytesAvailable
   Debug toto\TotalNumberOfBytes
   Debug toto\FreeSpaceGb
   Debug toto\FreeSpaceMb
   Debug toto\TotalSpaceGb
   Debug toto\TotalSpaceMb
```

```
EndIf
```

OS Supportés (testé) :

OS non compatibles :

madlib_ChargeVariablesEnvironement()

Syntaxe :

Resultat = madlib_ChargeVariablesEnvironement(Map DICT_VariablesEnvironement.s())

Description :

Charge les variables d'environement dans un dictionnaire.

Le résultat sera le nombre de variables récupérées.

Exemple :

```
Newmap DICT_MesVariablesEnvironement.s()

Debug madlib_ChargeVariablesEnvironement(DICT_MesVariablesEnvironement())
```

OS Supportés (testé) :

madlib_GetIPByNameWin

madlib_GetIPByNameWin()

Description :

Retourne l'adresse IP V4 à partir du nom de machine.

 La version du socket utilisé pour cette fonction est la version 1.1

Il est possible de la modifier, mais réservé aux initiés.

Pour les erreur il suffit de récuperer le code retour de la fonction et le traiter avec madlib_TextErrorWinWSA.

Syntaxe :

```
Resultat.i =
madlib_GetIPByNameWin(ComputerName$,*IPAddress,MajorVersionSoc
ket.b=1,MinorVersionSocket.b=1)
```

Paramètres & options :

ComputerName$: Nom de la machine cible.

***IPAddress :** Pointeur mémoire sur une variable de type chaîne de caractères. Celle-ci contiendra l'adresse IP de la machine cible.

MajorVersionSocket.b : Version majeur du socket Windows.

MinorVersionSocket.b : Version mineur du socket Windows.

Resultat :

L'adresse ip de la machine sera stocké à l'emplacement *IPAdresse.

En valeur négative est le retour en erreur (API Windows, erreur interne à la fonction, erreur de socket).

Exemple :

```
Define AdresseIP$
debug madlib_GetIPByName("NomMachine",@AdresseIP$)
```

OS Supportés (testé) :

OS non compatibles :

madlib_NetJoinDomainWin

madlib_NetJoinDomainWin()

Description :

Retourne l'adresse IP à partir du nom de machine.

La version du socket utilisé pour cette fonction est la version 1.1

Il est possible de la modifier, mais réservé aux initiés.

Pour les erreur il suffit de récuperer le code retour de la fonction et le traiter avec madlib_TextErrorWinWSA.

Syntaxe :

Resultat.i = NetJoinDomainWin(Domain$, Account$, Password$, OU$="",HostName$=#NULL$,JoinOptions.l =

#madlib_WIN_NETSETUP_JOIN_DOMAIN)

Paramètres & options :

Domain$ => Nom du domaine cible.

***Account$:** Compte utilisateur permettant l'insertion dans le domain cible.

Password$: Mot de passe du compte utilisateur.

OU$: Unité d'Organisation cible de l'activedirectory où sera placé la machine.

HostName$: Le nom machine a inserer. Par défaut il n'est pas necessaire de spécifier le nom de machine.

JoinOptions.l : Options d'insertion de la machine. Pour plus de renseignement sur le sujet, il faut se referer sur le site de Microsoft (MSDN).

Constantes associés :

Pour l'option "**JoinOptions.l**" :

#madlib_WIN_NETSETUP_JOIN_DOMAIN

#madlib_WIN_NETSETUP_ACCT_CREATE

#madlib_WIN_NETSETUP_WIN9X_UPGRADE

#madlib_WIN_NETSETUP_DOMAIN_JOIN_IF_JOINED

#madlib_WIN_NETSETUP_JOIN_UNSECURE

#madlib_WIN_NETSETUP_MACHINE_PWD_PASSED

#madlib_WIN_NETSETUP_DEFER_SPN_SET

#madlib_WIN_NETSETUP_JOIN_WITH_NEW_NAME

#madlib_WIN_NETSETUP_JOIN_READONLY

Resultat :

L'adresse ip de la machine sera stocké à l'emplacement *IPAdresse.

En valeur négative est le retour en erreur (API Windows, erreur interne à la fonction, erreur de socket).

Exemple :

OS Supportés (testé) :

OS non compatibles :

madlib_GetComputerName()

Description :

Retourne le nom de la machine Windows. L'avantage de cette fonction est principalement dans le faite de l'utiliser a distance.

Elle peut être utilisé aussi, pour les notions de "BADDNS" qui peuvent survenir sur un parc informatique.

Syntaxe :

Resultat.s = madlib_GetComputerName(IPAdressOrComputerName$="")

Paramètres & options :

IPAdressOrComputerName$: Nom de la machine cible.

Resultat :

Le nom de la machine

Exemple :

```
debug madlib_GetComputerName("NomMachine")
debug madlib_GetComputerName("10.0.0.1")
```

OS Supportés (testé) :

OS non compatibles :

madlib_GetComputerGUID

madlib_GetComputerGUID()

Description :

Retourne le GUID de la machine Windows. L'avantage de cette fonction est principalement dans le faite de l'utiliser à distance.

Syntaxe :

Resultat.s = madlib_GetComputerGUID(ComputerName$="")

Paramètres & options :

ComputerName$: Nom de la machine cible.

Resultat :

Retourne le GUID de machine

Exemple :

```
debug madlib_GetComputerGUID("NomMachine")
debug madlib_GetComputerGUID("10.0.0.1")
```

OS Supportés (testé) :

OS non compatibles :

madlib_InjectionDLL()

Description :

Permet de connecter dans un programme lancé, une librairie externe.

Syntaxe :

Resultat.i = madlib_InjectionDLL(ProcessID.l,DLLLocation$)

Resultat :

- -1 : Erreur sur les paramètres données, soit l'identifiant du processus n'est pas bon ou aucune librairie n'est spécifié.
- < -1 : Erreur windows, utiliser la fonction madlib_ReverseReturnCodei().
- #True : Opération effectué correctement.

Exemple :

```
NotePad = RunProgram("notepad", "", "", #PB_Program_Open|
#PB_Program_Read)

Debug(NotePad)

Debug "Attente de 5 secondes"
Delay(5000)

If NotePad
  ProcessID = ProgramID(NotePad)

  If ProcessID
    Debug DLLInjection(ProcessID, "e:\Programmation\PureBasic
\injector_dll\dll101.dll")
  EndIf
EndIf
```

OS Supportés (testé) :

OS non compatibles :

madlib_ProtectProcessWin()

Description :

Protège le processus ciblé par le vidage des sécurités DACL. Il sera donc impossible de supprimer le processus en mémoire par exemple.

Syntaxe :

Resultat.i = madlib_ProtectProcessWin(IDProcess.i=#PB_Ignore)

Paramètres & options :

IDProcess.i : ID du processus cible ou par défaut celui courant.

Resultat :

- < -1 : Erreur windows, utiliser la fonction madlib_ReverseReturnCodei().
- #True : Opération effectué correctement.

Exemple :

```
debug madlib_ProtectProcessWin()
```

OS Supportés (testé) :

OS non compatibles :

Flux de données alternatifs

Différentes fonctions, pour lister, écrire, lire, supprimer les Alternative Data Stream, disponible en format chaine de caractères ou utilisation de mémoire tampon.

Toutes les fonctions, y compris le système des "ADS" lui même se porte sur les

répertoires ou les fichiers. Ils est donc tout à fait possible de positionner un flux aternatif sur un répertoire.

 Ce système étant à l'origine dans la norme POSIX, n'est disponible que sur le système de fichiers dérivé, NTFS.

madlib_ListAdStreamData

madlib_ListADStreamData()

Description :

Permet de créer une liste des flux alternative de type $DATA.

Syntaxe :

```
Resultat.i = madlib_ListADStreamData(Location$,List
LST_Stream.madlib_ADStreamInfo())
```

Paramètres & options :

Location$: Chemin du fichier ou du répertoire

List LST_Stream : Liste chaîné contenant le résultat. Structure spécifique.

Structure(s) associée(s) :

- madlib_ADStreamInfo

Resultat :

- **Au dessus de 0** : Nombre de flux récupéré.
- **0** : Impossible d'ouvrir l'objet spécifié
- **En dessous de 0** : Codes d'erreur standard Windows.

Exemple :

```
Debug madlib_ListADStreamData("d:\toto",lst)
```

OS Supportés (testé) :

OS non compatibles :

madlib_AddStringADStream

madlib_AddStringADStream()

Description :

Créer ou modifie un flux et remplis sont contenu par la chaine de caractères.

Syntaxe :

Resultat.i = madlib_AddStringADStream(Location$,LabelName$,Text$,TextUnicode.a=#madlib_AutoChoice)

Paramètres & options :

Location$: Chemin du fichier ou du répertoire.

LabelName$: Nom du flux de données. Si aucun nom n'est spécifié le nom du répertoire ou du fichier sera utilisé.

Text$: Chaîne de caracctères.

TextUnicode.a : Permet d'enregistrer au format Unicode ou Ascii.

En mode "AutoChoice" il sera choisi automatiquement le mode en fonction de la compilation.

Resultat :

- **Au dessus de 0** : Enregistrement réussi.

- **0** : Impossible d'enregistrer le flux.
- **#madlib_ERROR_AT_CREATE** : Impossible d'ouvrir le flux sur l'objet spécifié.

Exemple :

```
Debug madlib_AddStringADStream("c:\temp,"ESSAI","Coucou
c'est moi !")
```

OS Supportés (testé) :

OS non compatibles :

madlib_AddADStream()

Description :

Créer ou modifie un flux et remplis sont contenu par la mémoire tampon.

Syntaxe :

```
Resultat.i =
madlib_AddADStream(Location$,LabelName$,*Buffer,BufferSize.i,UnicodeBuffer.a=#madlib_AutoCh
oice)
```

Paramètres & options :

Location$: Chemin du fichier ou du répertoire.

LabelName$: Nom du flux de données. Si aucun nom n'est spécifié le nom du répertoire ou du fichier sera utilisé.

***Buffer :** Nom du flux de données.

BufferSize : Nom du flux de données.

UnicodeBuffer.a : Permet de dimenssionner la mémoire tampon au format Unicode ou Ascii;

En mode "AutoChoice" il sera choisi automatiquement le mode en fonction de la compilation.

Resultat :

- **Au dessus de 0** : Nombre d'octets écrit dans le flux.
- **#madlib_ERROR_AT_CREATE** : Impossible d'ouvrir / créer le flux sur l'objet spécifié.
- **#madlib_NOT_EXIST** : Le fichier ou répertoire n'existe pas.

Exemple :

```
#FICHIER = "d:\a.txt"

Define toto$,titi.s{6}, *zz

toto$ = "coucou"
titi = "ciicaa"
*zz = AllocateMemory(200)

PokeS(*zz,"coucou")

Debug CreateStream(#FICHIER,"",@toto$,Len(toto$))
Debug CreateStream(#FICHIER,"zizi",@toto$,Len(toto$))
Debug CreateStream(#FICHIER,"tete",@toto$,Len(toto$))
```

OS Supportés (testé) :

OS non compatibles :

madlib_ReadADStream

madlib_ReadADStream()

Description :

Créer ou modifie un flux et remplis sont contenu par la mémoire tampon.

Syntaxe :

```
Resultat.i = madlib_ReadADStream(Location$,LabelName$,*Buffer,BufferSize.i)
```

Paramètres & options :

Location$: Chemin du fichier ou du répertoire.

LabelName$: Nom du flux de données. Si aucun nom n'est spécifié le nom du répertoire ou du fichier sera utilisé.

***Buffer :** Adresse mémoire tampon de réception.

BufferSize : Taille du tampon mémoire.

Resultat :

- **Au dessus de 0** : Nombre d'octets écrit dans le flux.
- **#madlib_BUFFER_TOO_SMALL** : La mémoire tampon est trop petit.
- **#madlib_NOT_EXIST** : Le fichier ou répertoire n'existe pas.

Exemple :

```
#FICHIER = "d:\a.txt"

Define *ii

*ii = AllocateMemory(200)

Debug madlib_ReadADStream(#FICHIER,"",@toto$,Len(toto$))
```

OS Supportés (testé) : **OS non compatibles :**

madlib_ReadStringADStream

madlib_ReadStringADStream()

Description :

Créer ou modifie un flux et remplis sont contenu par la mémoire tampon.

Syntaxe :

```
Resultat.s =
madlib_ReadStringADStream(Location$,LabelName$,TextUnicode.a=#madlib_AutoChoice)
```

Paramètres & options :

Location$: Chemin du fichier ou du répertoire.

LabelName$: Nom du flux de données. Si aucun nom n'est spécifié le nom du répertoire ou du fichier sera utilisé.

TextUnicode.a : Permet d'enregistrer au format Unicode ou Ascii.

En mode "AutoChoice" il sera choisi automatiquement le mode en fonction de la compilation.

Resultat :

- "" : Soit le flux est vide, soit il n'existe pas. (a vérifier avec la fonction ListAdStreamData())
- <> "" : Contenu du flux.

Exemple :

```
#FICHIER = "d:\a.txt"
```

```
Debug madlib_AddStringADStream(#FICHIER,"ESSAI","Coucou c'est
moi !")
Debug madlib_ReadStringADStream(#FICHIER,"ESSAI")
```

OS Supportés (testé) :

OS non compatibles :

madlib_DeleteADStream()

Description :

Supprime un flux.

Syntaxe :

Resultat.a = madlib_DeleteADStream(Location$,LabelName$)

Paramètres & options :

Location$: Chemin du fichier ou du répertoire.

LabelName$: Nom du flux de données. Si aucun nom n'est spécifié le nom du répertoire ou du fichier sera utilisé.

Resultat :

- **0** : Impossible de supprimer le flux spécifié.
- **#True** : Suppression effectuée.

Exemple :

```
#FICHIER = "d:\a.txt"

Debug madlib_DeleteADStream(#FICHIER,"")
```

OS Supportés (testé) :

OS non compatibles :

madlib_Subst()

Description :

Permet de de la substitution de répertoire. Créer une lettre de lecteur associé a un répertoire. Il est possible d'associté des répertoires réseau, cepandant il n'est pas conseillé de le faire.

Syntaxe :

Resultat.i = madlib_Subst(Letter$,Path$,RemoveLetter.a=#False)

Parametres :

Letter$: Lettre de lecteur qui sera associé au répertoire.

Path$: Répertoire.

RemoveLetter.a : Par défaut #False, passé à #True il supprimera le subst.

Resultat :

- < 0 : Erreur windows, utiliser la fonction madlib_ReverseReturnCodei().
- #True : Opération effectué correctement.

Exemple :

```
debug madlib_Subst("R:\","D:\Temp")
```

OS Supportés (testé) :

OS non compatibles :

Données mémoire

Cette partie, permet de stocker des informations simple en mémoire dans une table de base de données en mémoire.

il est possible de faire un Swap pour pouvoir utiliser des

La méthode de fonctionne et agit come un dictionnaire de hachage (Map).

Il existe une fonction de recherche afin de pouvoir lister toutes les entrées qui corresponde au critère.

Avec cet essemble de fonction, la méthode consiste à executer des informations en mémoire et/ou en fichier, afin de récupérer des informations si l'application plante ou faire du travail partagé entre deux applications.

 Tout cette mécanique utilise la librairie SQLLite de purebasic.

madlib_CreateFieldGrown

madlib_CreateFieldGrown()

Description :

Créer un emplacement en mémoire ou dans un fichier que l'on peut spécifier.

Syntaxe :

Resultat.i = madlib_CreateFieldGrown(File$="")

Option :

File$ => Chemin du fichier mémoire.

Resultat :

Si succès le retour sera différent de zéro. Cela sera l'identifiant de la zone mémoire alloué.

Si 0 alors un problème lors de la création (fichier ou mémoire), est survenue.

Exemple :

```
Debug madlib_CreateFieldGrown()
Debug madlib_CreateFieldGrown("D:\temp\test.mem")
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_CreatePlotGrown()

Description :

Créer un contenu de valeur.

Syntaxe :

Resultat.i = madlib_CreatePlotGrown(HandleFieldGrown.i,PlotName$,PlotType.l)

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

PlotName$ =>Nom du conteneur de valeurs.

PlotType.l => Type de conteneur de valeurs (voir les constantes ci-dessous).

Constantes associés :

#madlib_GROWN_PLOTTYPE_STRING => Chaine de caractère.

#madlib_GROWN_PLOTTYPE_LONG => Numériques de type "long".

#madlib_GROWN_PLOTTYPE_QUAD => Numériques de type "quad".

#madlib_GROWN_PLOTTYPE_FLOAT => Numériques de type "float".

#madlib_GROWN_PLOTTYPE_BLOB => Numériques de type "blob".

Resultat :

Si #True => conteneur créer correctement.

Si #False => Erreur de création de conteneur.

Exemple :

```
*HandleGrown = CreateFieldGrown()

Debug
CreatePlotGrown(*HandleGrown,"TestMaVariable",#madlib_GROWN_PLO
TTYPE_STRING)
Debug
CreatePlotGrown(*HandleGrown,"TMonSuperBloblie",#madlib_GROWN_P
LOTTYPE_BLOB)
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_GetPlotGrownList()

Description :

Récupère la listes des de conteneurs de valeurs et stocke les nom des "Plot" dans une liste.

Syntaxe :

Resultat.b = madlib_GetPlotGrownList(HandleFieldGrown.i,List

LST_PlotGrown.s())

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

List LST_PlotGrown.s() => Liste de type string qui contiendra le résultat.

Resultat :

Si #True => Les "Plot" ont été listé correctement.

Si #False => Erreur dans la récupération.

Exemple :

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_SearchSeedByName

madlib_SearchSeedByName()

Description :

Recherche un nom de valeur dans un conteneur. Le résultat sera retournée dans une liste. Cette fonction permet aussi de lister plusieurs valeurs. Il inclus une

notion de recherche de valeur.

Syntaxe :

Resultat.b = madlib_SearchSeedByName(HandleFieldGrown.i,PlotName$,List LST_SearchResult.s(),SeedName$="")

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

PlotName$ => Nom du conteneur de valeur.

List LST_SearchResult.s() => Liste de type string qui contiendra le résultat.

SeedName$ => Partie du nom ou complet, de la valeur.

Resultat :

Si #True => Les valeurs ont été listées correctement.

Si #False => Erreur dans la récupération/recherche.

Exemple :

```

```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_DeletePlotGrown

madlib_DeletePlotGrown()

Description :

Supprime un conteneur de valeur.

Syntaxe :

Resultat.b = madlib_DeletePlotGrown(HandleFieldGrown.i,PlotName$)

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

PlotName$ => Nom du conteneur de valeur.

Resultat :

Si #True => Les valeurs ont été listées correctement.

Si #False => Erreur dans la récupération/recherche.

Exemple :

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_SizePlotGrown

madlib_SizePlotGrown()

Description :

Renvois le nombre de valeur du conteur de valeur.

Syntaxe :

Resultat.b = madlib_SizePlotGrown(HandleFieldGrown.i,PlotName$)

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

PlotName$ => Nom du conteneur de valeur.

Resultat :

Si #True => Les valeurs ont été listées correctement.

Si #False => Erreur dans la récupération/recherche.

Exemple :

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_GetTypePlotGrown

madlib_GetTypePlotGrown()

Description :

Renvois le type de valeur du conteur de valeur.

Syntaxe :

Resultat.l = madlib_GetTypePlotGrown(HandleFieldGrown.i,PlotName$)

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

PlotName$ => Nom du conteneur de valeur.

Resultat :

Voir constantes associés.

Constantes associés :

#madlib_GROWN_PLOTTYPE_STRING => Chaine de caractère.

#madlib_GROWN_PLOTTYPE_LONG => Numériques de type "long".

#madlib_GROWN_PLOTTYPE_QUAD => Numériques de type "quad".

#madlib_GROWN_PLOTTYPE_FLOAT => Numériques de type "float".

#madlib_GROWN_PLOTTYPE_BLOB => Numériques de type "blob".

Exemple :

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_ResetPlot()

Description :

Supprime toutes les entrées d'un conteneur de valeur.

Syntaxe :

Resultat.b = madlib_ResetPlot(HandleFieldGrown.i,PlotName$)

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

PlotName$ => Nom du conteneur de valeur.

Resultat :

Si #True => Les valeurs ont été listées correctement.

Si #False => Erreur dans la récupération/recherche.

Exemple :

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_DeleteSeed()

Description :

Supprime une valeur contenu dans un "Plot".

Syntaxe :

Resultat.b = madlib_DeleteSeed(HandleFieldGrown.i,PlotName$,SeedName$)

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

PlotName$ => Nom du conteneur de valeur.

SeedName$ => Nom de la valeur.

Resultat :

Si #True => Les valeurs ont été listées correctement.

Si #False => Erreur dans la récupération/recherche.

Exemple :

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_AddSeedString()

Description :

Ajoute / enregistre une valeur avec nom et contenu dans le conteneur au format de caractères.

Syntaxe :

```
Resultat.b = madlib_AddSeedString(HandleFieldGrown.i,PlotName$,SeedName$,SeedValue$="")
```

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

PlotName$ => Nom du conteneur de valeur.

SeedName$ => Nom de la valeur.

SeedValue$ => Valeur.

Resultat :

Si #True => Les valeurs ont été listées correctement.

Si #False => Erreur dans la récupération/recherche.

Exemple :

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_AddSeedInt()

Description :

Ajoute / enregistre une valeur avec nom et contenu dans le conteneur au format numérique entier.

Syntaxe :

```
Resultat.b =
madlib_AddSeedInd(HandleFieldGrown.i,PlotName$,SeedName$,SeedVa
lue.i="")
```

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

PlotName$ => Nom du conteneur de valeur.

SeedName$ => Nom de la valeur.

SeedValue.i => Valeur.

Resultat :

Si #True => Les valeurs ont été listées correctement.

Si #False => Erreur dans la récupération/recherche.

Exemple :

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_AddSeedFloat

madlib_AddSeedFloat()

Description :

Ajoute / enregistre une valeur avec nom et contenu dans le conteneur au format numérique flotant.

Syntaxe :

Resultat.b = madlib_AddSeedFloat(HandleFieldGrown.i,PlotName$,SeedName$,SeedValue.f="")

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

PlotName$ => Nom du conteneur de valeur.

SeedName$ => Nom de la valeur.

SeedValue.f => Valeur.

Resultat :

Si #True => Les valeurs ont été listées correctement.

Si #False => Erreur dans la récupération/recherche.

Exemple :

```

```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_GetSeedString

madlib_GetSeedString()

Description :

Récupere, un contenu de valeur de format texte.

 Aucune erreur ne sera généré. Il faut donc utiliser, afin de vérifier si une valeur existe dans le conteneur.

Syntaxe :

Resultat$ = madlib_GetSeedString(HandleFieldGrown,PlotName$,SeedName$,*StringBuffer)

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

PlotName$ => Nom du conteneur de valeur.

SeedName$ => Nom de la valeur.

***StringBuffer** => Adresse de la zone mémoire pour le retour.

Resultat :

Si #True => Les valeurs ont été listées correctement.

Si #False => Erreur dans la récupération/recherche.

Exemple :

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_GetSeedInt

madlib_GetSeedInt()

Description :

Récupere, un contenu de valeur de format numérique.

Aucune erreur ne sera généré. Pour vérifier si une valeur existe dans le conteneur, utilisez

Syntaxe :

Resultat.i = madlib_GetSeedInt(HandleFieldGrown,PlotName$,SeedName$)

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

PlotName$ => Nom du conteneur de valeur.

SeedName$ => Nom de la valeur.

Resultat :

Si #True => Les valeurs ont été listées correctement.

Si #False => Erreur dans la récupération/recherche.

Exemple :

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_ExistSeed

madlib_ExistSeed()

Description :

Récupere le nombre d'occurance correspond a SeedName$.

Il est possible d'utiliser la fonction de recherche pour récuperer le nombre de valeurs correspondantes.

Syntaxe :

```
Resultat.i =
madlib_ExistSeed(HandleFieldGrown,PlotName$,SeedName$)
```

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

PlotName$ => Nom du conteneur de valeur.

SeedName$ => Nom de la valeur, ou partie. Dans le cas d'une recherche utilser le caractère %.

Resultat :

Si #False => Erreur dans la récupération/recherche.

Si au dessus de 0 => Nombre d'entrées correspondantes.

Exemple :

```
Debug SeedExist(*HandleGrown,"TestMaVariable","%ma%")
Debug SeedExist(*HandleGrown,"TestMaVariable","MonNomDeValeur")
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_AddSeedDouble()

Description :

Ajoute / enregistre une valeur avec nom et contenu dans le conteneur au format numérique flotant double précision.

Syntaxe :

Resultat.b = madlib_AddSeedDoube(HandleFieldGrown.i,PlotName$,SeedName$,SeedValue.d)

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

PlotName$ => Nom du conteneur de valeur.

SeedName$ => Nom de la valeur.

SeedValue.d => Valeur.

Resultat :

Si #True => Les valeurs ont été listées correctement.

Si #False => Erreur dans la récupération/recherche.

Exemple :

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_GetSeedDouble

madlib_GetSeedDouble()

Description :

Récupere, un contenu de valeur de format numérique.

 Aucune erreur ne sera généré. Pour vérifier si une valeur existe dans le conteneur, utilisez la fonction ExistSeed.

Syntaxe :

Resultat.d = madlib_GetSeedDouble(HandleFieldGrown,PlotName$,SeedName$)

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

PlotName$ => Nom du conteneur de valeur.

SeedName$ => Nom de la valeur.

Resultat :

Si #True => Les valeurs ont été listées correctement.

Si #False => Erreur dans la récupération/recherche.

Exemple :

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_GetAllSeedString()

Description :

Récupere, un enssemble de contenu au format chaine de caractère.

Il est possible de spécifier des filtres afin de sélectionner des valeurs dans la liste.

Aucune erreur ne sera généré. Pour vérifier si une valeur existe dans le conteneur, utilisez la fonction ExistSeed.

Syntaxe :

Resultat.s = madlib_GetAllSeedString(HandleFieldGrown.i,PlotName$,Map DictResultSeed.s(),FilterSeedName$="",FilterSeedValue$="")

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

PlotName$ => Nom du conteneur de valeur.

DictResultSeed.s() => Dictionnaire des résultats.

FilterSeedName$ => Filtrage sur le nom de la valeur.

FilterSeedValue$ => Filtrage sur la valeur.

Resultat :

Retourne le nombre de ligne contenu dans le dictionnaire.

Exemple :

```
NewMap essai.s()

Debug GetAllSeedString(*HandleGrown,"TestMaVariable",essai())

ForEach essai()
    Debug MapKey(essai())+" : "+essai()
Next
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_GetAllSeedInt

madlib_GetAllSeedInt()

Description :

Récupere, un enssemble de contenu au format chaine de caractère.

Il est possible de spécifier des filtres afin de sélectionner des valeurs dans la liste.

 Aucune erreur ne sera généré. Pour vérifier si une valeur existe dans le conteneur, utilisez la fonction ExistSeed.

Syntaxe :

Resultat.i = madlib_GetAllSeedInt(HandleFieldGrown.i,PlotName$,Map DictResultSeed.i(),FilterSeedName$="",FilterSeedValue$="")

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

PlotName$ => Nom du conteneur de valeur.

DictResultSeed.i() => Dictionnaire des résultats.

FilterSeedName$ => Filtrage sur le nom de la valeur.

FilterSeedValue$ => Filtrage sur la valeur.

Resultat :

Retourne le nombre de ligne contenu dans le dictionnaire.

Si #False => Erreur dans la récupération/recherche.

Exemple :

```
NewMap essai.i()

Debug GetAllSeedInt(*HandleGrown,"TestMaVariable",essai())

ForEach essai()
    Debug MapKey(essai())+" : "+essai()
Next
```

OS Supportés (testé) : **OS Compatibles (non testé) :**

madlib_GetAllSeedFloat

madlib_GetAllSeedFloat()

Description :

Récupere, un enssemble de contenu au format numérique flotant.

Il est possible de spécifier des filtres afin de sélectionner des valeurs dans la liste.

Aucune erreur ne sera généré. Pour vérifier si une valeur existe dans le conteneur, utilisez la fonction ExistSeed.

Syntaxe :

Resultat.f = madlib_GetAllSeedFloat(HandleFieldGrown.i,PlotName$,Map DictResultSeed.f(),FilterSeedName$="",FilterSeedValue$="")

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

PlotName$ => Nom du conteneur de valeur.

DictResultSeed.f() => Dictionnaire des résultats.

FilterSeedName$ => Filtrage sur le nom de la valeur.

FilterSeedValue$ => Filtrage sur la valeur.

Resultat :

Retourne le nombre de ligne contenu dans le dictionnaire.

Si #False => Erreur dans la récupération/recherche.

Exemple :

```
NewMap essai.f()

Debug GetAllSeedFloat(*HandleGrown,"TestMaVariable",essai())

ForEach essai()
    Debug MapKey(essai())+" : "+essai()
Next
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_GetAllSeedDouble()

Description :

Récupere, un enssemble de contenu au format numérique flotant double précision.

Il est possible de spécifier des filtres afin de sélectionner des valeurs dans la liste.

Aucune erreur ne sera généré. Pour vérifier si une valeur existe dans le

conteneur, utilisez la fonction ExistSeed.

Syntaxe :

```
Resultat.d =
madlib_GetAllSeedDouble(HandleFieldGrown.i,PlotName$,Map
DictResultSeed.d(),FilterSeedName$="",FilterSeedValue$="")
```

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

PlotName$ => Nom du conteneur de valeur.

DictResultSeed.d() => Dictionnaire des résultats.

FilterSeedName$ => Filtrage sur le nom de la valeur.

FilterSeedValue$ => Filtrage sur la valeur.

Resultat :

Retourne le nombre de ligne contenu dans le dictionnaire.

Si #False => Erreur dans la récupération/recherche.

Exemple :

```
NewMap essai.d()

Debug GetAllSeedDouble(*HandleGrown,"TestMaVariable",essai())

ForEach essai()
    Debug MapKey(essai())+" : "+essai()
Next
```

OS Supportés (testé) :

OS Compatibles (non testé) :

madlib_GetAllSeedQuad

madlib_GetAllSeedQuad()

Description :

Récupere, un enssemble de contenu au format numérique.

Il est possible de spécifier des filtres afin de sélectionner des valeurs dans la liste.


Aucune erreur ne sera généré. Pour vérifier si une valeur existe dans le conteneur, utilisez la fonction ExistSeed.

Syntaxe :

```
Resultat.q =
madlib_GetAllSeedQuad(HandleFieldGrown.i,PlotName$,Map
DictResultSeed.q(),FilterSeedName$="",FilterSeedValue$="")
```

Option :

HandleFieldGrown.i => ID de la zone mémoire ouverte par la fonction CreateFieldGrown.

PlotName$ => Nom du conteneur de valeur.

DictResultSeed.q() => Dictionnaire des résultats.

FilterSeedName$ => Filtrage sur le nom de la valeur.

FilterSeedValue$ => Filtrage sur la valeur.

Resultat :

Retourne le nombre de ligne contenu dans le dictionnaire.

Si #False => Erreur dans la récupération/recherche.

Exemple :

```
NewMap essai.q()

Debug GetAllSeedQuad(*HandleGrown,"TestMaVariable",essai())

ForEach essai()
    Debug MapKey(essai())+" : "+essai()
Next
```

OS Supportés (testé) :

OS Compatibles (non testé) :

Structures

Description des structures

Les structures sont utilisé par les fonctions de la MadLib.
Elles sont implanté en tant que ressource de Purebasic.
Elles ne sont donc pas encore compilé à cet étape d'utilisation.

Toute les structures sont préfixées de madlib_ afin d'éviter tout conflit avec d'autre "structure userlib".

madlib_ConfigService

Utilisée dans les fonctions :

Toutes fonctions de dans la catégorie Services Windows.

Description :

Membres	Désignation
ComputerName$	EXPERIMENTALE. Spécifie un nom de machine sur le réseau.
Name$	Nom du service. Il sera utilisé pour créer ou supprimé ou gerer le service. Il doit être unique.
DisplayName$	Nom d'affichage. C'est celui que l'on trouve singulièrrement dans le gestionnaire des services.
Description$	Description du service. Généralement associé avec le displayname$. Visible aussi sous le gestionnaire des services.
CommandLine$	Ligne de commande de démarrage du service.
CommandLineParametre$	Ajout de paramètre pour la ligne de commande.
StartType.i	Comportement du service au démarrage machine. Plusieurs modes sont possible : ▪ #SERVICE_AUTO_START ▪ #SERVICE_BOOT_START ▪ #SERVICE_DEMAND_START ▪ #SERVICE_DISABLED ▪ #SERVICE_SYSTEM_START
ServiceType.i	Type de service, permettant de prioriser le niveau de criticité : ▪ #SERVICE_WIN32_OWN_PROCESS (avec l'option : #SERVICE_INTERACTIVE_PROCESS

	▪ #SERVICE_FILE_SYSTEM_DRIVER ▪ #SERVICE_KERNEL_DRIVER ▪ #SERVICE_WIN32_SHARE_PROCESS (avec l'option : #SERVICE_INTERACTIVE_PROCESS)
ErrorControlType.i	Typage et gestion du comportement des erreurs de services : ▪ #SERVICE_ERROR_CRITICAL ▪ #SERVICE_ERROR_IGNORE ▪ #SERVICE_ERROR_NORMAL ▪ #SERVICE_ERROR_SEVERE
ServiceStatus.SERVICE_STATUS	Structure interne au fonctionnement des services. Indique l'état des services.
AccesControl.l	Controle d'accès pour le SCManager et l'instanciation des services. Plusieurs choix sont possible :
Journal.madlib_ConfigJournal	Journal pour l'execution, créatio, suppression, etc, du service. Si aucun besoin n'est demandé laisser à 0.
ind_verif.b	Marqueur de vérification de la conformité de la structure. Si pour des raisons de prudence une nouvelle vérification devrait être executé, mettre le membre à la valeur 0.
ServiceTable.SERVICE_TABLE_ENTRY	Table d'entrée pour indiquer l'état du service et lister les processus de démarrage que va gerer le gestionnaire des services. Ne pas toucher.
StatusMemory.MEMORYSTATUS	Structure permettant le contrôle mémoire du service au niveau de la couche service, toujours effectué par le gestionnaire de service.
ForceActions.b	Même si des erreurs surviennent dans les différent processus, ce membre permet d'outrepassé les erreurs d'executions. Non recommandé.
MainThread.madlib_Thread	Connecteur de fonctions Purebasic.
StopService.b	Indiquateur d'une demande d'arrêt de service. Utile pour les processus qui gravitent autour et qui n'ont pas de visu sur les demande du gestionnaire de service. Lorque q'une demande de stop est demandé, ce membre passe à la valeur #True
StartProgram.madlib_ConfigProgram	Structure de démarrage d'un programme lors du lancement du service.

madlib_ConfigJournal

madlib_ConfigJournal

Utilisée dans les fonctions :

- madlib_Journal
- Services Windows
- madlib_ServeurIP
- madlib_ApplicationStart

Description :

Membres	Désignation
ind_init.i	Indiquateur pour la fonction de journalisation si la structure a été déjà une première fois configuré, il sert aussi à compter le nombre de fois que la structure est passé dans la fonction journal.
CheminFichier$	Chemin du fichier à incrémenter Le chemin peut être calculé. (%exemple% \toto.log)
TailleFichierMax.i	Taille maximum du fichier en octet.
Type$	Type de fichier (html,csv,log).
FormatDate$	Formatage de la date. Par défaut la date sera complète (jour mois année heure minute seconde).
NiveauGlobal.b	De 0 à 127 maximum (donc 127 niveaux)
IndiquateurNiveauGlobal$	Indiquateur du niveau global si besoin
Map IndiquateurEcriture.s()	Dictionnaire des indicateurs des écritures
ArchiveMax.i	Taille maximale du fichier de log ou htm
EnteteFichier$	Permet de déposer dans l'entête du fichier
SuffixeNomFichierArchive$	Permet de placer un suffixe quand le fichier est archivée

madlib_ConfigProgram

madlib_ConfigProgram

Utilisé dans les fonctions :

madlib_ApplicationStart

Description :

Membres	Désignation
ProgramLocation$	Chemin de l'executable.
Parameters$	Paramètres de l'executable.
WorkingDirectory$	Répertoire de travail.
WaitTime.i	Temp d'attente maximum de l'executable une fois lancée.
DemandeStop.b	Demande d'arrêt
ReturnCode.i	Code retour du programme une fois terminée est a PB_Ignore si pas programme non terminée.
TextFormat.i	Format du texte des retours des deux canaux.
ReturnText$	Retour du canal de texte de l'application.
ReturnTextError$	Retour du canal de texte de l'application pour les erreurs.
HandleExeWindows.i	Identifiant Windows pour l'application lancé
ExecutionMode.i	Paramètre en plus pour le lancement de l'application. Se referer au lancement de l'application avec RunProgram de la STL de Purebasic
smph_StopProgram.i	Sémaphore pour la gestion de l'arrêt du programme.Il pourrat être utilisé lors de la demande d'arrêt du programme, afin d'attendre la fin de l'execution. Par défaut il faut l'utiliser avec la fonction de la STL "WaitSemaphore"
ForceRunning.b	Par défaut est #False. Si L'option est modifié alors il sera retenté de lancer l'application indéfiniment.

madlib_Thread

madlib_Thread

Utilisé dans les fonctions :

Services Windows

Description :

Membres	Désignation
*Procedure	Pointeur vers la procédure (@MaProcedure)
*Parametre	Pointeur de paramètre. Sinon placer 0
WaitThread.a	Attend le retour du sous processus.
TimeOut.i	Permet de spécifier un temp d'attente maximum.

madlib_ConfigServeurIP

madlib_ConfigServeurIP

Utilisé dans les fonctions :

- madlib_ServeurIP
- madlib_RecupereDonneesServeurIP

Description :

Membres	Désignation
PortEcoute.u	Port d'écoute pour le serveur TCP
TailleBuffer.i	Taille du buffer en octet.
SortieErreur.b	Si le membre la valeur est à #True,
TempEcouteMaxi.i	Exprimé en seconde, sert à limiter le temps d'écoute, ensuite on quitte.
CptConnexionMax.i	Compteur de connexion si 0 illimité, si supérieur alors une fois le nombre atteint déconnexion.
Buffer.l	Pointeur pour le buffer.
ModeUDP.b	Si #True, alors le serveur sera ouvert en mode UDP
ID_ServeurEcoute.i	Handle du serveur en écoute.
SEM_ReceptionDonnees.i	Sémaphore automatiquement créé au démarrage du serveur. Il est utiliser pour notifier
MUT_Buffer.i	Mutex pour indiquer si la mémoire tampon utilisé pour le transfert de texte est en cours d'utilisation.
AccesControl.l	Controle d'accès pour le SCManager et l'instanciation des services. Plusieurs choix sont possible :
*Journal.madlib_ConfigJournal	Pointeur d'un journal pour l'execution et l'activité du
REF_ServeurIP.i	Retour sur l'execution du serveur.
SEM_AttenteRetourReception.i	Sémpahore, si différent de zéro alors le serveur attendra un signal de celui-ci pour aller chercher dans la variable membre RetourReception$. Si on spécifie avant de démarrer -1 dans le membre, il fera automatiquement la création du sémaphore.
RetourReception$	Chaine de caractère à envoyer pour la bonne réception des données.
TypeServeur.b	Par défaut le serveur IP sera en écoute en mode TCP. Si la constante est marqué en #PB_Network_UDP
InviteALaConnexion$	Chaine de caractère, qui ne sert q'une seule fois à la premièrre connexion du client.
InviteCourante$	Invite courante affichée une fois après le retour.
RafraEvenements.u	Temp de rafraichissement des événements du serveur IP. Attention, les valeurs sont comprise entre 1 et xxxxx. millisecondeS. Plus le temps sera cout, plus la charge sera importante.

madlib_ini

madlib_ini

Utilisée dans les fonctions :

madlib_LoadIniFile

Description :

Membres	Désignation
GroupeName.s{512}	Remplis automatiquement, par la fonction, indique le nom de la section du fichier ini.
KeyName.s{512}	Nom de la clef contenu dans la section.
Value.s	Valeur de la clef.

madlib_ObjetsFenetresMDI

madlib_ObjetsFenetresMDI

Utilisée dans les fonctions :

- madlib_AjouteObjetFenetre
- madlib_IDObjetFenetre
- madlib_CommentaireObjetFenetre

Description :

Membres	Désignation
NomGroupe.s	Remplis automatiquement, par la fonction, indique le nom de la section du fichier ini.
NomClef.s	Nom de la clef contenu dans la section.
ValeurClef.s	Valeur de la clef.

madlib_RegPathValue

madlib_RegPathValue

Utilisé dans les fonctions :

madlib_RegParsePath

Description :

Membres	Désignation
hKey.q	Valeur du contexte de la clef de base de registre. Exemple : #HKEY_LOCAL_MACHINE
SubKey$	Chemin de la clef de base de registre
ComputerName$	Nom de la machine cible (par défaut c'est >> . <<)

madlib_TabChr

madlib_TabChr

Utilisé dans les fonctions :

madlib_RegWriteMultiSZ

madlib_RegReadMultiSZ

Description :

Membres	Désignation
Chr.c[0]	Tableau de caractère

madlib_ConfigDiskFreeSpace

madlib_ConfigDiskFreeSpace

Utilisée dans les fonctions :

- madlib_WinGetFreeSpace

Description :

Membres	Désignation
DiskLocation.s{255}	Désigne le disque à vérifier
FreeBytesAvaible.q	Espace libre en octets
TotalNumberOfBytes.q	Taille totale en octets
TotalNumberOfFreeBytes.q	Taille total en espace libre
FreeSpaceMb.q	Espace libre en MégaOctets
FreeSpaceGb.d	Espace libre en GigaOctets
TotalSpaceMb.d	Taille totale en MégaOctets
TotalSpaceGb.d	Taille totale en GigaOctets

madlib_ConfigDownloadFile

madlib_ConfigDownloadFile

Utilisée dans les fonctions :

- madlib_DownloadFile
- madlib_DownloadFileList

Description :

Membres	Désignation
URLLocation$	Chemin et fichier HTTP du fichier à télécharger (ex : http://www.toto.com/toto.exe)
FileNameLocationReceive$	Chemin et nom du fichier local pour la réception.
FileSizeDownloaded.i	Taille du fichier à qui a été télécharger (membre qui se lie après téléchargement)
MD5FileDownloaded$	MD5 du fichier qui été téléchargé (membre qui se lie après téléchargement)
DateDownloaded.i	Date de téléchargement effecuté (membre qui se lie après téléchargement)
RetryOnFailed.i	Nombre de tentative de téléchargement effectué

madlib_XML

madlib_XML

 Cette partie n'est pas active dans cette version de MadLib.

Utilisée dans les fonctions :

- madlib_LoadXML
- madlib_BrowseXML
- madlib_FindNodeXML

Description :

Membres	Désignation
Name$	Nom de la balise
Content$	Contenu de la balise
Location$	Chemin de la balise contenu dans le fichier
*Node	Pointeur du noeud lors de la récuperation dans le fichier
Map Attributs.s()	Attributs de la balise. Il se peut, en fonction de l'utilisation, que les sous balises soit dans le dictionnaire.

madlib_ConfigExpandString

madlib_ConfigExpandString

Utilisée dans les fonctions :

- madlib_ExpandString

Description :

Membres	Désignation
MatchKey.s{255}	Caractère d'abastraction, il est

	possible de spécifier plusieurs caractère séparé par une virgule.
LoadSystemVariable.b	Si l'option est à #True, les variables d'environements seront chargées.
LoadInternalVariable.b	Si l'option est à #True, les variables interne seront chargées.
LoadCustomVariable.b	Si l'option est à #True, le dictionnaire de hachage (MAP) sera executé.
CaseSensitive.b	Si l'option est à #True, les variables seront reconnu en "Case sensitive". Ex: %UsErNaMe% et %username% seront différent.

madlib_ConfigLogFile

madlib_ConfigLogFile

Utilisée dans les fonctions :

- madlib_Journal
- Services Windows
- madlib_ServeurIP
- madlib_ApplicationStart

Description :

Membres	Désignation
ind_init.i	Indiquateur pour la fonction de journalisation si la structure a été déjà une première fois configuré, il sert aussi à compter le nombre de fois que la structure est passé dans la fonction journal.
FileLocation.s{512}	Chemin du fichier à incrémenter. Chaines de longueur 512.
FileMaxSize.q	Taille maximum du fichier en octet.
FileType.s{5}	Type de fichier (html,csv,log).
DateFormat.s{30}	Formatage de la date. Par défaut la date sera complète (jour mois année heure minute seconde).
GlobalLevel.b	De 0 à 127 maximum (donc 127 niveaux). Ce niveau sert quand on utiliser des niveau relatif. (ex: NIV+3) il sera donc relatif sur le niveau global.

DefaultEventTypes.s{3}	Indiquateur du niveau global si besoin, Il sera possible de spécifier l'option "NIV,ATT ou ERR"
FileSave.b	Force la sauvegarde pour un passage dans la fonction. Une fois effective il passera automatiquement à #False.
MaxArchive.i	Nombre de fichiers "archive" maximum.
FileHeader.s{1024}	Dépose une entête de fichier. pour les nouveaux fichier créer
ArchiveSuffixName.s{10}	Place un suffixe dans le nom quand le fichier est archivée.
HandleFile.i	ID du fichier ouvert
CountAutoSave.i	Il est possible de spécifier un compteur de sauvegarde. En effet si on spécifie 5, Le premier passage dans la fonction le fichier est ouvert, mais n'est pas refermé, puis au bou de la 5eme fois, le fichier enregistre les données et le fichier est refermé. La 6eme il réouvre le fichier et recommance le comptage à 5.
PassCount.i	Compteur de passage, avant la sauvegarde. Cette propriété de la structure, n'est pas utilisable directement. Il faut pour définir
SpaceIndentation.s{2}	Spécifie le caractère qui servira l'indentation (génération de l'espace). Il est possible de spécifier deux caractères d'indentations. Cette option ne sera utilisé que pour une fichier texte par défaut.

madlib_XMLNodes

madlib_XMLNodes

Utilisée dans les fonctions :

- madlib_GetXMLBaseNodeList

Description :

Membres	Désignation
Name$	Nom de la balise

Content$	Contenu de la balise
Location$	Chemin de la balise contenu dans le fichier
TypeContent.a	Type du noeud. #PB_XML_Normal, etc ...
IDNode.l	ID unique du noued récuperé depuis l'ouverture et le parcours du fichier.

madlib_XMLNodesInfos

madlib_XMLNodesInfos

Utilisée dans les fonctions :

Description :

Membres	Désignation
NodeID.l	ID unique du noued récuperé depuis l'ouverture et le parcours du fichier.
TypeContent.a	Type du noeud. #PB_XML_Normal, etc ...
CountSubNode.i	Nombre de noeuds enfant.
ParentNodeID.l	ID du noeud parent dont le noeud courant dépend

madlib_ConfigRequestIPServer

madlib_ConfigService

Utilisée dans les fonctions :

madlib_RequestIPServer

Description :

Membres	Désignation
Server$	Adresse du serveur
Port.i	Port de connexion au serveur
*MemoryReception	Pointeur sur mémoire contenant le retour
*MemorySend	Pointeur sur mémoire d'envois pour la requete vers le serveur.
Text$	Texte à envoyer
WaitResponse.i	Délai d'attente de la réponse (en secondes)
Refresh.i	Rafraichissement du canal d'écoute du client
ConnexionType.i	Méthode de connexion (TCP / UDP)
FileReception$	Chemin du fichier de réception
FileSend$	Chemin du fichier d'envois vers le serveur.

madlib_ConfigSplachScreen

madlib_ConfigSplachScreen

Utilisée dans les fonctions :

madlib_RequestIPServer

Description :

Membres	Désignation
_IDSplachScreen.i	Id de la fenêtre splach. (réservé)
_IDGadgetText.i	ID du gadget Texte. (réservé)
_IDGadgetProgressBar.i	ID du gadget ProgressBar, il sera possible de l'utiliser afin de modifier la valeur de progression.
_IDImage.i	ID de l'image. (réservé)
ImageFile.s	chemin de l'image à afficher.
WindowHeight.i	Hauteur de la fenêtre.
WindowWidth.i	Largeur de la fenêtre.
Xpos.i	Position de la fenêtre sur la hauteur de l'écran.
Ypos.i	Position de la fenêtre sur la largeur de l'écran.
TimeOut.i	En secondes, temp max d'affichage du splach.
Foreground.b	Force la fenêtre à être au premier plan. (#True)
TextColor.s{12}	Couleur du texte (format #00AA00)
BackGroundTextColor.s{12}	Couleur du fond de texte (format #00AA00)
Alpha.a	Couche Alpha, permet de spécifier une transparence de fenêtre. Ne fonctionne que sous Windows. Option allant de 0 à 255 (opaque à transparent).
Text.b	Active le champ texte, si l'option est à #True
ProgressBar.b	Active le champ progressbar si l'option est à #True
ProgressBarColor.s{12}	Couleur de la progressBar (format #00AA00)
BackColorProgressBar.s{12}	Couleur de l'arrière plan de la progressBar (format #00AA00)

madlib_ADStreamInfo

madlib_ADStreamInfo

Utilisé dans les fonctions :

madlib_ListADStreamData

Description :

Membres	Désignation
Name.s{255}	Nom du flux alternatif
Type.s{10}	Type du flux ($DATA, etc...)
Size.q	Taille des données en octets.

madlib_ConfigWindowNotify

madlib_ConfigWindowNotify

Utilisé dans les fonctions :

madlib_ListADStreamData

Description :

Membres	Désignation
Title.s{255}	Titre de la fenêtre
Text.s{1024}	Texte
*Icon	Pointeur de l'icône chargé avec LoadImage() et ImageID()
Height.i	Hauteur de la fenêtre de notification (pixel)
Width.i	Largeur de la fenêtre de notification (pixel)
WindowColor.i	Couleur du fond de la fenêtre (valeur numérique)
TextColor.i	Couleur du texte (valeur numérique)
BorderColor.i	Couleur de la bordure du texte (valeur numérique)

TimeOut.i	Temp exprimé en secondes. Si 0 est placé alors pas d'attente. Si #PB_Ignore est placé, alors, attente indéfini jusqu'a ce que la fenêtre soit fermé.
SpeedScrollUp.i	Vitesse de défilement quand la fenêtre de notification s'affiche.
SpeedScrollDown.i	Vitesse de défilement quand la fenêtre de notification s'éfface.
ForeFront.a	#True => Passage au premier plan systématique #False (par défaut) => Pas de premier si une fenêtre vient après par dessus elle ne sera plus visible.
TypeFontText.s{50}	Nom d'une police de caractère pour le texte.
TextSize.a	Taille de la police du texte.
TitleSize.a	Taille de la police du titre.
Desktop.a	Numéro du bureau d'affichage ou l'on veut récupérer la taille de l'écran.
TypeFontTitle.s{50}	Nom d'une police de caractère pour le titre.
NoCloseWindow.a	#True => Il sera impossible de fermer la fenêtre.

Constantes

Constantes

Cette partie consiste à décrire au possible des les constante utilisé. Elle seront regroupées par catégorie d'utilisation.

Serveur IP

Serveur IP

Utilisé par :

- madlib_ServeurIP
- madlib_RecupereDonneesServeurIP

Description :

Nom	Type	Valeur	Désignation
#MADLIB_SERVEURIP_ENCOURS_CREATION	Numérique (Dynamique)		Utilisé pour indiqué que le serveur n'est pas encore opérationel.
#MADLIB_SERVEURIP_ENCOURS_EXECUTION	Numérique (Dynamique)		Fonctionnement OK, le serveur IP fonctionne correctement.
#MADLIB_SERVEURIP_ERREUR_CREATION_SERVEUR	Numérique (Dynamique)		Erreur dans la création du serveur.
#MADLIB_SERVEURIP_ERREUR_ALLOCATION_MEMOIRE	Numérique (Dynamique)		Impossible d'allouer la mémoire pour le buffer.
#MADLIB_SERVEURIP_ERREUR_RECUPERATION_DONNEES	Numérique (Dynamique)		Impossible de récuperer ou de placer les données en mémoire.
#MADLIB_SERVEURIP_NOMBRE_MACHINES_MAX	Numérique (Dynamique)		Quota de machines atteintes.
#MADLIB_SERVEURIP_ATTENTE_RETOUR	Numérique (Dynamique)		Le serveur lancé, attent un signal, afin de poursuivre son execution.

Réalisation

Réalisation de la MADLib

Le langage de programmation est de conception Franco-Allemande. Il se nomme Purebasic.

Sans ce langage de "bas niveau", ce logiciel ne serait jamais aussi performant.

Pour plus d'informations sur purebasic : http://www.purebasic.fr

Comment et par quel méthode la compilation à été effectué ?

Le logiciel TAILBITE est utilisé pour la complilation des procédure.

Pour plus d'informations sur purebasic : http://www.tailbite.com

L'aide à été réalisée sous quel logiciel ?

L'ensemble de la documentation est réalisé avec HelpNDoc. Ce logiciel permet de créer très facilement la documentation qui sera exportée en plusieurs formats (WEB html, PDF, CHM, DOC, etc ...).

Pour plus d'informations sur HelpNDoc : http://www.helpndoc.com

Note relatives au passage de paramètres

Passage de paramètres dans les fonctions

Les fonctions dans la MadLib, sont compilées. Il est donc impossible de modifier ni le type ni le comportement des fonctions.

Aussi, il est indispenssable de bien respecter les types de paramètres demandés par les fonctions.

Un accent est à apporter sur les chaines de caractères.

Dans le language Purebasic, il y a deux types de chaines.

La chaine avec allocation mémoire (toto$ ou toto.s)
La chaine avec allocation sur la pile (toto.s{20})

Ces deux type de chaine sont différentes !

Dans la MadLib, les paramètres avec passage de chaines allocation mémoire, seront demandé sur des passages par valeurs.

En revanche, la chaine allocation sur la pile sera demandé par référence.