Peregrine

Connect-It 3.0.0 - Référence de programmation

PART NUMBER CIT-3.0.0-FRE-01002-00188

Cette édition s'applique à la version 3.0.0 du programme sous contrat de licence Connect-It

Peregrine Systems, Inc.
Worldwide Corporate Campus and Executive Briefing Center
3611 Valley Centre Drive San Diego, CA 92130
Tel 800.638.5231 or 858.481.5000
Fax 858.481.1751
www.peregrine.com

Référence de programmation

Table des matières

II. Référence alphabétique 35

I. Introduction

1 Champ d'application des fonctions

CHAPITRE

Les fonctions décrites dans ce document sont toutes utilisables dans les fenêtres de script de Connect-It.

2 Conventions

CHAPITRE

Ce chapitre contient des informations sur les conventions d'écriture utilisées dans ce document et sur le format de certaines constantes.

Conventions d'écriture

La syntaxe des fonctions et des exemples proposés respecte les conventions d'écriture suivantes :

[]	Exception : dans les scripts Basic, lorsque les crochets encadrent le chemin d'accès à des données de la base, ils doivent figurer dans le script, comme le montre l'exemple ci-dessous : **`[Lien.Lien.Champ]`** Ces crochets encadrent un paramètre optionnel. Ne les tapez pas dans votre commande.
<>	Ces crochets encadrent un paramètre décrit en langage clair. Ne les tapez pas dans votre

	commande et remplacez le texte qu'ils encadrent par l'information qui doit y figurer.
{}	Ces accolades encadrent des paramètres parmi lesquels un seul doit être choisi. Ne tapez pas les accolades dans votre commande.
|	La barre verticale sépare les paramètres possibles qui figurent dans les accolades.
*	L'astérisque ajouté à droite de crochets indique que la formule qu'ils encadrent peut être répétée plusieurs fois.

Format des constantes de type Date+Heure dans les scripts

Les dates référencées dans les scripts sont exprimées au format international, indépendamment des options d'affichage spécifiées par l'utilisateur :

```
yyyy/mm/dd hh:mm:ss
```

Exemple :

```
RetVal="1998/07/12 13:05:00"
```

Note : Le tiret ("-") peut également être utilisé comme séparateur de date.

Date Basic et Date Unix

Les dates sont exprimées différemment en Basic et Unix :

- En Basic, une date peut être exprimée au format international ou sous la forme d'un nombre à virgule flottante (type "Double"). Dans ce dernier cas, la partie entière de ce nombre représente le nombre de jours écoulés depuis le 30/12/1899 à 0:00, la partie décimale représente la fraction écoulée dans le jour courant.

- Sous Unix, les dates sont exprimées sous la forme d'un entier long (type "Long" 32 bits) qui représente le nombre de secondes écoulées depuis le 01/01/1970 à 0:00, indépendamment d'un quelconque fuseau horaire (heure UTC).

Format des constantes de type Durée

Dans les scripts, les durées sont stockées et exprimées en secondes. Par exemple, pour fixer la valeur par défaut d'un champ de type "Durée" à 3 jours, vous devez utiliser le script suivant :

```
RetVal=259200
```

De même, les fonctions qui calculent ou traitent une durée fournissent un résultat en secondes.

Note : Dans les calculs financiers, Connect-It tient compte des simplifications habituellement usitées. Dans ce cas seulement, une année vaut 12 mois et un mois vaut 30 jours (d'où : 1 année = 360 jours).

3 Définitions

CHAPITRE

Ce chapitre regroupe les définitions de quelques termes essentiels.

Définition d'une fonction

Une fonction est un programme qui effectue des opérations et renvoie à l'utilisateur une valeur, appelée "valeur de retour" ou "code de retour".

Voici un exemple de syntaxe d'appel d'une fonction par le Basic interne de Connect-It :

```
PifIgnoreDocumentMapping(strMsg As String) As Long
```

Définition d'un code d'erreur

Lorsque l'exécution d'une fonction échoue, un code d'erreur est renvoyé.

Vous pouvez déclencher volontairement un message d'erreur en utilisant la fonction `Err.Raise` dont la syntaxe est la suivante :

```
Err.Raise (<Numéro de l'erreur>, <Message
d'erreur>)
```

4 Typage des fonctions et des paramètres de fonctions

CHAPITRE

Liste des types

Le tableau ci-dessous récapitule les différents types possibles pour une fonction ou un paramètre :

Type	Signification
Integer	Nombre entier de -32 768 à +32 767.
Long	Nombre entier de -2 147 483 647 à +2 147 483 646.
Double	Nombre à virgule flottante de 8 octets.
String	Texte pour lequel tous les caractères sont acceptés.
Date	Date ou Date+Heure.
Variant	Type générique pouvant représenter n'importe quel type.

Type d'une fonction

Le type d'une fonction correspond au type de la valeur retournée par la fonction. Nous vous invitons à faire particulièrement attention à cette information car elle peut être à l'origine d'erreurs de compilation et d'exécution de vos programmes.

Type d'un paramètre

Les paramètres utilisés dans les fonctions possèdent également un type que vous devez impérativement respecter pour la bonne exécution de la fonction. Dans la syntaxe des fonctions, les paramètres sont préfixés en fonction de leur type. Le tableau ci-dessous résume les différents types utilisés et le préfixe qui leur est associé :

Type	Préfixe utilisé dans la syntaxe Basic
Integer	"i"
Long	"l"
Double	"d"
String	"str"
Date	"dt"
Variant	"v"

II. Référence alphabétique

CHAPITRE 5 Référence alphabétique

Abs()

Syntaxe BASIC interne

```
Function Abs(dValue As Double) As Double
```

Description

Renvoie la valeur absolue d'un nombre.

Entrée

- *dValue* : Nombre dont vous souhaitez connaître la valeur absolue.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim iSeed as Integer
iSeed = Int((10*Rnd)-5)
RetVal = Abs(iSeed)
```

AppendOperand()

Syntaxe BASIC interne

```
Function AppendOperand(strExpr As String,
strOperator As String, strOperand As String) As
String
```

Description

Concatène une chaîne en fonction des paramètres passés à la fonction. Le résultat a la forme suivante :

```
strExpr
strOperator
strOperand
```

Entrée

- *strExpr* : Expression à concaténer.
- *strOperator* : Opérateur à concaténer.
- *strOperand* : Opérande à concaténer.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Remarques

Note : Si l'un des paramètres strExpr ou strOperand est omis, strOperator n'est pas utilisé dans la concaténation.

ApplyNewVals()

Syntaxe BASIC interne

```
Function ApplyNewVals(strValues As String,
strNewVals As String, strRows As String,
strRowFormat As String) As String
```

Description

Affecte des valeurs identiques pour les cellules identifiées d'un contrôle "ListBox".

Entrée

- *strValues* : Chaîne source contenant les valeurs d'un contrôle "ListBox" à traiter.
- *strNewVals* : Nouvelle valeur à affecter aux cellules concernées.
- *strRows* : Identifiants des lignes à traiter. Les identifiants sont séparés par une virgule.
- *strRowFormat* : Instructions de formatage de la sous-liste. Les instructions sont séparées par le caractère "|". Chaque instruction représente le numéro de la colonne qui contiendra *strNewVals*.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Asc()

Syntaxe BASIC interne

```
Function Asc(strAsc As String)
```

Description

Renvoie le code ASCII du premier caractère d'une chaîne.

Entrée

- *strAsc* : Chaîne de caractères sur laquelle opère la fonction.

Exemple

```
Dim iCount as Integer
Dim strString as String
  For iCount=Asc("A") To Asc("Z")
    strString = strString & Str(iCount)
  Next iCount
  RetVal=strString
```

Atn()

Syntaxe BASIC interne

```
Function Atn(dValue As Double) As Double
```

Description

Renvoie l'arc tangente d'un nombre, exprimé en radians

Entrée

- *dValue* : Nombre dont vous souhaitez connaître l'arc tangente.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim dPi as Double
Dim strString as String
  dPi=4*Atn(1)
  strString = Str(dPi)
  RetVal=strString
```

BasicToLocalDate()

Syntaxe BASIC interne

```
Function BasicToLocalDate(strDateBasic As String)
As String
```

Description

Cette fonction convertit une date au format Basic en une date au format chaîne (telle qu'elle est affichée dans le "control panel" de Windows).

Entrée

- *strDateBasic* : Date au format Basic à convertir.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

BasicToLocalTime()

Syntaxe BASIC interne

```
Function BasicToLocalTime(strTimeBasic As String)
As String
```

Description

Cette fonction convertit une heure au format Basic en une heure au format chaîne (telle qu'elle est affichée dans le "control panel" de Windows).

Entrée

- *strTimeBasic* : Heure au format Basic à convertir.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

BasicToLocalTimeStamp()

Syntaxe BASIC interne

```
Function BasicToLocalTimeStamp(strTSBasic As
String) As String
```

Description

Cette fonction convertit un ensemble Date+Heure au format Basic en un ensemble Date+Heure au format chaîne (telle qu'elle est affichée dans le "control panel" de Windows).

Entrée

- *`strTSBasic`* : Date+Heure au format Basic à convertir.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Beep()

Syntaxe BASIC interne

```
Function Beep()
```

Description

Emet un son (un beep) sur la machine.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

CDbl()

Syntaxe BASIC interne

```
Function CDbl(dValue As Double) As Double
```

Description

Convertit une expression en un double ("Double").

Entrée

- *dValue* : Expression à convertir.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim dNumber As Double
Dim iInteger as Integer
  iInteger = 25
  dNumber=CDbl(iInteger)
  RetVal=dNumber
```

ChDir()

Syntaxe BASIC interne

```
Function ChDir(strDirectory As String)
```

Description

Change le répertoire courant.

Entrée

- *`strDirectory`* : Nouveau répertoire courant.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

ChDrive()

Syntaxe BASIC interne

```
Function ChDrive(strDrive As String)
```

Description

Change le lecteur courant.

Entrée

- *`strDrive`* : Nouveau lecteur.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Chr()

Syntaxe BASIC interne

```
Function Chr(iChr As Long) As String
```

Description

Renvoie la chaîne correspondant au code ASCII passé par le paramètre *iChr*.

Entrée

- *iChr* : Code ASCII du caractère.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim iCount as Integer
Dim iIteration as Integer
Dim strMessage as String
Dim strLF as String
  strLF=Chr(10)
  For iIteration=1 To 2
    For iCount=Asc("A") To Asc("Z")
      strMessage=strMessage+Chr(iCount)
    Next iCount
```

```
    strMessage=strMessage+strLF
  Next iIteration
  RetVal=strMessage
```

CInt()

Syntaxe BASIC interne

```
Function CInt(iValue As Long) As Long
```

Description

Convertit une expression en un entier ("Integer").

Entrée

- *iValue* : Expression à convertir.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim iNumber As Integer
Dim dDouble as Double
  dDouble = 25.24589
  iNumber=CInt(dDouble)
  RetVal=iNumber
```

CLng()

Syntaxe BASIC interne

```
Function CLng(lValue As Long) As Long
```

Description

Convertit une expression en un long ("Long").

Entrée

- *lValue* : Expression à convertir.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim lNumber As Long
Dim iInteger as Integer
  iInteger = 25
  lNumber=CLng(iInteger)
  RetVal=lNumber
```

Cos()

Syntaxe BASIC interne

```
Function Cos(dValue As Double) As Double
```

Description

Renvoie le cosinus d'un nombre, exprimé en radians.

Entrée

- *dValue* : Nombre dont vous souhaitez connaître le cosinus.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim dCalc as Double
  dCalc=Cos(150)
  RetVal=dCalc
```

CountOccurences()

Syntaxe BASIC interne

```
Function CountOccurences(strSearched As String,
strPattern As String, strEscChar As String) As Long
```

Description

Compte le nombre d'occurrences d'une chaîne de caractères à l'intérieur d'une autre chaîne.

Entrée

- *strSearched* : Chaîne de caractères à l'intérieur de laquelle s'effectue la recherche.

- *strPattern* : Chaîne de caractères à rechercher à l'intérieur de *strSearched*.
- *strEscChar* : Caractère d'échappement. Si la fonction rencontre ce caractère à l'intérieur de la chaîne *strSearched*, la recherche s'arrête.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim MyStr
  MyStr=CountOccurences("toi|moi|toi,moi|toi",
"toi", ",") :'Renvoie la valeur "2"
  MyStr=CountOccurences("toi|moi|toi,moi|toi",
"toi", "|") :'Renvoie la valeur "1"
```

CountValues()

Syntaxe BASIC interne

```
Function CountValues(strSearched As String,
strSeparator As String, strEscChar As String) As
Long
```

Description

Compte le nombre d'éléments dans une chaîne de caractères en tenant compte d'un séparateur et d'un caractère d'échappement.

Entrée

- *strSearched* : Chaîne de caractères à traiter.
- *strSeparator* : Séparateur utilisé pour délimiter les éléments.
- *strEscChar* : Caractère d'échappement. Si ce caractère préfixe un séparateur, ce dernier sera ignoré.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim MyStr
  MyStr=CountValues("toi|moi|toi\|moi|toi", "|",
"\") :'Renvoie la valeur 4
  MyStr=CountValues("toi|moi|toi\|moi|toi", "|",
"") :'Renvoie la valeur 5
```

CSng()

Syntaxe BASIC interne

```
Function CSng(fValue As Single) As Single
```

Description

Convertit une expression en un nombre à virgule flottante ("Float").

Entrée

- *fValue* : Expression à convertir.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim dNumber As Double
Dim iInteger as Integer
  iInteger = 25
  dNumber=CSng(iInteger)
  RetVal=dNumber
```

CStr()

Syntaxe BASIC interne

```
Function CStr(strValue As String) As String
```

Description

Convertit une expression en une chaîne de caractères ("String").

Entrée

- *strValue* : Expression à convertir.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim dNumber As Double
Dim strMessage as String
  dNumber = 2,452873
  strMessage=CStr(dNumber)
  RetVal=strMessage
```

CurDir()

Syntaxe BASIC interne

```
Function CurDir() As String
```

Description

Renvoie le chemin courant.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

CVar()

Syntaxe BASIC interne

```
Function CVar(vValue As Variant) As Variant
```

Description

Convertit une expression en un variant ("Variant").

Entrée

- *vValue* : Expression à convertir.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Date()

Syntaxe BASIC interne

```
Function Date() As Date
```

Description

Renvoie la date courante du système.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

DateAdd()

Syntaxe BASIC interne

```
Function DateAdd(tmStart As Date, tsDuration As
Long) As Date
```

Description

Cette fonction calcule une nouvelle date en fonction d'une date de départ à laquelle est ajoutée une durée réelle.

Entrée

- *tmStart* : Ce paramètre contient la date à laquelle sera ajoutée une durée.
- *tsDuration* : Ce paramètre contient la durée à ajouter à la date *tmStart*.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

DateAddLogical()

Syntaxe BASIC interne

```
Function DateAddLogical(tmStart As Date, tsDuration
As Long) As Date
```

Description

Cette fonction calcule une nouvelle date en fonction d'une date de départ à laquelle est ajoutée une durée logique (un mois comporte 30 jours).

Entrée

- *tmStart* : Ce paramètre contient la date à laquelle sera ajoutée une durée.
- *tsDuration* : Ce paramètre contient la durée, exprimée en secondes, à ajouter à la date *tmStart*.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

DateDiff()

Syntaxe BASIC interne

```
Function DateDiff(tmEnd As Date, tmStart As Date)
As Date
```

Description

Cette fonction calcule en secondes la durée écoulée entre deux dates.

Entrée

- *tmEnd* : Ce paramètre contient la date de fin de la période sur laquelle est effectué le calcul.
- *tmStart* : Ce paramètre contient la date de début de la période sur laquelle est effectué le calcul.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

L'exemple suivant calcule la durée écoulée entre le 01/01/98 et le 01/01/99.

```
AmDateDiff("1998/01/01 00:00:00", "1999/01/01
00:00:00")
```

DateSerial()

Syntaxe BASIC interne

```
Function DateSerial(iYear As Long, iMonth As Long,
iDay As Long) As Date
```

Description

Cette fonction renvoie une date formatée en fonction des paramètres *iYear*, *iMonth* et *iDay*.

Entrée

- *iYear* : Année. Si sa valeur est comprise entre 0 et 99, ce paramètre décrit les années de 1900 à 1999. Pour toutes les autres années, vous devez utiliser un nombre de quatre chiffres (par exemple 1800).
- *iMonth* : Mois.
- *iDay* : Jour.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

Chacun de ces paramètres peut prendre pour valeur une expression numérique représentant un nombre de jours, de mois ou d'années. Ainsi l'exemple suivant :

```
DateSerial(1999-10, 3-2, 15-8)
```

renvoie la valeur :

```
1989/1/7
```

Lorsque la valeur d'un paramètre est en dehors de l'intervalle de valeurs généralement admis (c'est à dire 1-31 pour les jours, 1-12 pour les mois, ...), la fonction renvoie une date vide.

DateValue()

Syntaxe BASIC interne

```
Function DateValue(tmDate As Date) As Date
```

Description

Cette fonction renvoie la partie date d'une valeur "Date+Heure"

Entrée

- *`tmDate`* : Date au format "Date+Heure".

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

L'exemple suivant :

```
DateValue ("1999/09/24 15:00:00")
```

renvoie la valeur :

```
1999/09/24
```

Day()

Syntaxe BASIC interne

```
Function Day(tmDate As Date) As Long
```

Description

Renvoie le jour contenu dans le paramètre *tmDate*.

Entrée

- *tmDate* : Paramètre au format Date+Heure à traiter.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim strDay as String
  strDay=Day(Date())
  RetVal=strDay
```

EscapeSeparators()

Syntaxe BASIC interne

```
Function EscapeSeparators(strSource As String,
strSeparators As String, strEscChar As String) As
String
```

Description

Préfixe un ou plusieurs caractère(s) défini(s) comme séparateur(s) par un caractère d'échappement.

Entrée

- *strSource* : Chaîne de caractères à traiter.
- *strSeparators* : Liste des séparateurs à préfixer. Si vous souhaitez déclarer plusieurs séparateurs, vous devez les séparer par le caractère utilisé comme caractère d'échappement (indiqué dans le paramètre *strEscChar*.
- *strEscChar* : Caractère d'échappement. Il préfixera tous les séparateurs définis dans *strSeparators*.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim MyStr
  MyStr=EscapeSeparators("toi|moi|toi,moi|toi",
"|\,", "\") :'Renvoie la valeur
"toi\|moi\|toi\,moi\|toi"
```

ExeDir()

Syntaxe BASIC interne

```
Function ExeDir() As String
```

Description

Cette fonction renvoie le chemin complet de l'exécutable.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim strPath as string
  strPath=ExeDir()
```

Exp()

Syntaxe BASIC interne

```
Function Exp(dValue As Double) As Double
```

Description

Renvoie l'exponentielle d'un nombre.

Entrée

- *dValue* : Nombre dont vous souhaitez connaître l'exponentielle.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim iSeed as Integer
  iSeed = Int((10*Rnd)-5)
  RetVal = Exp(iSeed)
```

ExtractValue()

Syntaxe BASIC interne

```
Function ExtractValue(pstrData As String,
strSeparator As String, strEscChar As String) As
String
```

Description

Extrait d'une chaîne de caractères les valeurs délimitées par un séparateur. La valeur récupérée est alors effacée de la chaîne source. Cette opération tient compte d'un éventuel caractère d'échappement. Si le séparateur n'est pas trouvé dans la chaîne source, l'intégralité de la chaîne est renvoyée et la chaîne source est entièrement effacée.

Entrée

- *`pstrData`* : Chaîne source à traiter.
- *`strSeparator`* : Caractère utilisé comme séparateur dans la chaîne source.
- *`strEscChar`* : Caractère d'échappement. Si ce caractère préfixe le séparateur, ce dernier est ignoré.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim MyStr
  MyStr=ExtractValue("toi,moi", ",", "\") :'Renvoie
 "toi" et laisse "moi" dans la chaîne source
  MyStr=ExtractValue(",toi,moi", ",", "\")
:'Renvoie "" et laisse "toi,moi" dans la chaîne
source
  MyStr=ExtractValue("toi", ",", "\") :'Renvoie
"toi" et laisse "" dans la chaîne source
  MyStr=ExtractValue("toi\,moi", ",", "\")
:'Renvoie "toi\,moi" et laisse "" dans la chaîne
source
  MyStr=ExtractValue("toi\,moi", ",", "") :'Renvoie
 "toi\" et laisse "moi" dans la chaîne source
  RetVal=""
```

FileCopy()

Syntaxe BASIC interne

```
Function FileCopy(strSource As String, strDest As
String) As Long
```

Description

Copie un fichier ou un répertoire.

Entrée

- *strSource* : Chemin complet du fichier ou du répertoire à copier.
- *strDest* : chemin complet du fichier ou du répertoire de destination.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

FileDateTime()

Syntaxe BASIC interne

```
Function FileDateTime(strFileName As String) As
Date
```

Description

Renvoie la date et l'heure d'un fichier sous la forme d'un "Long".

Entrée

- *`strFileName`* : Chemin complet du fichier concerné par l'opération.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

FileExists()

Syntaxe BASIC interne

Description

Cette fonction teste l'existence d'un fichier.

FileLen()

Syntaxe BASIC interne

```
Function FileLen(strFileName As String) As Long
```

Description

Renvoie la taille d'un fichier.

Entrée

- *strFileName* : Chemin complet du fichier concerné par l'opération.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Fix()

Syntaxe BASIC interne

```
Function Fix(dValue As Double) As Long
```

Description

Renvoie la partie entière (premier entier supérieur dans le cas d'un nombre négatif) d'un nombre à virgule.

Entrée

- *dValue* : Nombre dont vous souhaitez connaître la partie entière.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim dSeed as Double
  dSeed = (10*Rnd)-5
  RetVal = Fix(dSeed)
```

FormatResString()

Syntaxe BASIC interne

```
Function FormatResString(strResString As String,
strParamOne As String, strParamTwo As String,
strParamThree As String, strParamFour As String,
strParamFive As String) As String
```

Description

Cette fonction traite une chaîne source en remplaçant les variables $1, $2, $3, $4 et $5 respectivement par les chaînes contenues dans les paramètres *strParamOne*, *strParamTwo*, *strParamThree*, *strParamFour* et *strParamFive*.

Entrée

- *strResString* : Chaîne source à traiter.
- *strParamOne* : Chaîne de remplacement de la variable $1.
- *strParamTwo* : Chaîne de remplacement de la variable $2.
- *strParamThree* : Chaîne de remplacement de la variable $3.
- *strParamFour* : Chaîne de remplacement de la variable $4.

- *strParamFive* : Chaîne de remplacement de la variable $5.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

L'exemple suivant :

```
FormatResString("je$1il$2vous$3", "tu", "nous",
"ils")
```

renvoie "jetuilnousvousils".

FV()

Syntaxe BASIC interne

```
Function FV(dblRate As Double, iNper As Long,
dblPmt As Double, dblPV As Double, iType As Long)
As Double
```

Description

Cette fonction renvoie le futur montant d'une annuité basée sur des versements constants et périodiques, et sur un taux d'intérêt fixe.

Entrée

- *dblRate* : Ce paramètre indique le taux d'intérêt par échéance. Par exemple pour un prêt à taux d'intérêt annuel de 6%, remboursé suivant une périodicité mensuelle, le taux par échéance est de :

```
0,06/12=0,005 soit 0,5%
```

- *iNper* : Ce paramètre contient le nombre total d'échéances de l'opération financière.
- *dblPmt* : Ce paramètre indique le montant du paiement à effectuer à chaque échéance. Le paiement comporte généralement le principal et les intérêts.
- *dblPV* : Ce paramètre contient la valeur actuelle (ou somme globale) d'une série de paiements devant être effectués dans le futur.
- *iType* : Ce paramètre indique la date d'échéances des paiements. Il peut prendre les valeurs suivantes :
  - *0* si les paiements sont dus à terme échu (c'est à dire en fin de période)
  - *1* si les paiements sont dus à terme à échoir (c'est à dire en début de période)

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Remarques

Note : Les paramètres Rate et Nper doivent être calculés à l'aide d'échéances exprimées dans les mêmes unités.Les sommes versées (exprimées notamment par le paramètre Pmt) sont représentées par des nombres négatifs. Les sommes reçues sont représentées par des nombres positifs.

GetListItem()

Syntaxe BASIC interne

```
Function GetListItem(strFrom As String, strSep As
String, lNb As Long, strEscChar As String) As
String
```

Description

Renvoie la *lNb*ième portion d'une chaîne délimitée par des séparateurs.

Entrée

- *strFrom* : Chaîne source à traiter.
- *strSep* : Caractère utilisé comme séparateur dans la chaîne source.
- *lNb* : Position de la chaîne à récupérer.
- *strEscChar* : Caractère d'échappement. Si ce caractère préfixe un séparateur, ce dernier sera ignoré.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

L'exemple suivant :

```
GetListItem("ceci_est_un_test", "_", 2, "%")
```

renvoie "est".

```
GetListItem("ceci%_est_un_test", "_", 2, "%")
```

renvoie "un".

Hex()

Syntaxe BASIC interne

```
Function Hex(dValue As Double) As String
```

Description

Renvoie la valeur hexadécimale d'un nombre.

Entrée

- *dValue* : Nombre dont vous souhaitez connaître la valeur hexadécimale.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Hour()

Syntaxe BASIC interne

```
Function Hour(tmTime As Date) As Long
```

Description

Renvoie la valeur de l'heure contenue dans le paramètre *tmTime*.

Entrée

- *tmTime* : Paramètre au format Date+Heure à traiter.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim strHour as String
  strHour=Hour(Date())
  RetVal=strHour
```

InStr()

Syntaxe BASIC interne

```
Function InStr(iPosition As Long, strSource As
String, strPattern As String) As Long
```

Description

Renvoie la position de la première occurrence d'une chaîne de caractères à l'intérieur d'une autre chaîne de caractères.

Entrée

- *`iPosition`* : Position de départ de la recherche. Ce paramètre ne peut être négatif et ne doit pas dépasser 65.535.
- *`strSource`* : Chaîne dans laquelle s'effectue la recherche.
- *`strPattern`* : Chaîne de caractères à rechercher.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim strSource as String
Dim strToSearch as String
Dim iPosition
strSource = "Good Bye"
strToSearch = "Bye"
iPosition = Instr(2, strSource, strToSearch)
RetVal=iPosition
```

Int()

Syntaxe BASIC interne

```
Function Int(dValue As Double) As Long
```

Description

Renvoie la partie entière (premier nombre entier inférieur dans le cas d'un nombre négatif) d'un nombre à virgule.

Entrée

- *dValue* : Nombre dont vous souhaitez connaître la partie entière.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim iSeed as Integer
  iSeed = Int((10*Rnd)-5)
  RetVal = Abs(iSeed)
```

IPMT()

Syntaxe BASIC interne

```
Function IPMT(dblRate As Double, iPer As Long,
iNper As Long, dblPV As Double, dblFV As Double,
iType As Long) As Double
```

Description

Cette fonction renvoie le montant des intérêts pour une échéance donnée d'une annuité.

Entrée

- *dblRate* : Ce paramètre indique le taux d'intérêt par échéance. Par exemple pour un prêt à taux d'intérêt annuel de 6%, remboursé suivant une périodicité mensuelle, le taux par échéance est de :

```
0,06/12=0,005 soit 0,5%
```

- *iPer* : Ce paramètre indique la période concernée par le calcul, comprise entre 1 et la valeur du paramètre *Nper*.
- *iNper* : Ce paramètre contient le nombre total d'échéances de l'opération financière.
- *dblPV* : Ce paramètre contient la valeur actuelle (ou somme globale) d'une série de paiements devant être effectués dans le futur.
- *dblFV* : Ce paramètre contient la valeur future ou le solde que vous souhaitez obtenir après avoir effectué le dernier paiement. En règle générale, et dans le cas d'un remboursement d'un emprunt en particulier, ce paramètre prend la valeur "0". En effet, une fois toutes les échéances remboursées, la valeur de l'emprunt est nulle.
- *iType* : Ce paramètre indique la date d'échéances des paiements. Il peut prendre les valeurs suivantes :

- *0* si les paiements sont dus à terme échu (c'est à dire en fin de période)
- *1* si les paiements sont dus à terme à échoir (c'est à dire en début de période)

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Remarques

Note : Les paramètres Rate et Nper doivent être calculés à l'aide d'échéances exprimées dans les mêmes unités.Les sommes versées (exprimées notamment par le paramètre Pmt) sont représentées par des nombres négatifs. Les sommes reçues sont représentées par des nombres positifs.

IsNumeric()

Syntaxe BASIC interne

```
Function IsNumeric(strString As String) As Long
```

Description

Cette fonction permet de déterminer si une chaîne de caractères contient une valeur numérique.

Entrée

- *strString* : Ce paramètre contient la chaîne de caractères à analyser.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Kill()

Syntaxe BASIC interne

```
Function Kill(strKilledFile As String) As Long
```

Description

Efface un fichier.

Entrée

- *strKilledFile* : Chemin complet du fichier concerné par l'opération.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

LCase()

Syntaxe BASIC interne

```
Function LCase(strString As String) As String
```

Description

Passe tous les caractères d'une chaîne en minuscules.

Entrée

- *strString* : Chaîne de caractères à passer en minuscules.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
' This example uses the LTrim and RTrim functions
 to strip leading ' and trailing spaces,
respectively, from a string variable.
' It uses the Trim function alone to strip both
types of spaces.
' LCase and UCase are also shown in this example
as well as the use
' of nested function calls

Dim strString as String
Dim strTrimString as String
  strString = "  <-Trim->  " :' Initialize string.
  strTrimString = LTrim(strString) :' strTrimString
 = "<-Trim->  ".
```

```
  strTrimString = LCase(RTrim(strString)) :'
strTrimString = "  <-trim->".
  strTrimString = LTrim(RTrim(strString)) :'
strTrimString = "<-Trim->".
  ' Using the Trim function alone achieves the
same result.
  strTrimString = UCase(Trim(strString)) :'
strTrimString = "<-TRIM->".
  RetVal= "|" & strTrimString & "|"
```

Left()

Syntaxe BASIC interne

```
Function Left(strString As String, iNumber As Long)
As String
```

Description

Renvoie les iNumber premiers caractères d'une chaîne en partant de la gauche.

Entrée

- *`strString`* : Chaîne de caractères à traiter.
- *`iNumber`* : Nombre de caractères à renvoyer.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim lWord, strMsg, rWord, iPos :' Declare
variables.
  strMsg = "Left() Test."
  iPos = InStr(1, strMsg, " ") :' Find space.
  lWord = Left(strMsg, iPos - 1) :' Get left word.
  rWord = Right(strMsg, Len(strMsg) - iPos) :' Get
 right word.
  strMsg=rWord+lWord :' And swap them
  RetVal=strMsg
```

LeftPart()

Syntaxe BASIC interne

```
Function LeftPart(strFrom As String, strSep As
String, bCaseSensitive As Long) As String
```

Description

Extrait la portion d'une chaîne de caractères située à gauche du séparateur précisé dans le paramètre *strSep*.

La recherche du séparateur s'effectue de la gauche vers la droite.

La recherche peut tenir compte ou non de la casse en fonction de la valeur du paramètre *bCaseSensitive*.

Entrée

- *strFrom* : Chaîne source à traiter.
- *strSep* : Caractère utilisé comme séparateur dans la chaîne source.
- *bCaseSensitive* : En fonction de la valeur de ce paramètre, la recherche respecte (=1) ou non (=0) la casse.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

Ces exemples illustrent l'utilisation des fonctions `LeftPart`, `LeftPartFromRight`, `RightPart` et `RightPartFromLeft` sur une même chaîne de caractères: "Ceci_est_un_test" :

```
LeftPart("Ceci_est_un_test","_",0)
```

Renvoie la chaîne "Ceci".

```
LeftPartFromRight("Ceci_est_un_test","_",0)
```

Renvoie la chaîne "Ceci_est_un".

```
RightPart("Ceci_est_un_test","_",0)
```

Renvoie la chaîne "test".

```
RightPartFromLeft("Ceci_est_un_test","_",0)
```

Renvoie la chaîne "est_un_test".

LeftPartFromRight()

Syntaxe BASIC interne

```
Function LeftPartFromRight(strFrom As String,
strSep As String, bCaseSensitive As Long) As String
```

Description

Extrait la portion d'une chaîne de caractères située à gauche du séparateur précisé dans le paramètre *`strSep`*.

La recherche du séparateur s'effectue de la droite vers la gauche.

La recherche peut tenir compte ou non de la casse en fonction de la valeur du paramètre *bCaseSensitive*.

Entrée

- *strFrom* : Chaîne source à traiter.
- *strSep* : Caractère utilisé comme séparateur dans la chaîne source.
- *bCaseSensitive* : En fonction de la valeur de ce paramètre, la recherche respecte (=1) ou non (=0) la casse.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

Ces exemples illustrent l'utilisation des fonctions `LeftPart`, `LeftPartFromRight`, `RightPart` et `RightPartFromLeft` sur une même chaîne de caractères: "Ceci_est_un_test" :

```
LeftPart("Ceci_est_un_test","_",0)
```

Renvoie la chaîne "Ceci".

```
LeftPartFromRight("Ceci_est_un_test","_",0)
```

Renvoie la chaîne "Ceci_est_un".

```
RightPart("Ceci_est_un_test","_",0)
```

Renvoie la chaîne "test".

```
RightPartFromLeft("Ceci_est_un_test","_",0)
```

Renvoie la chaîne "est_un_test".

Len()

Syntaxe BASIC interne

```
Function Len(vValue As Variant) As Long
```

Description

Renvoie le nombre de caractères d'une chaîne ou d'un variant.

Entrée

- *vValue* : Variant concerné par l'opération.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim strTest as String
Dim iLength as Integer
  strTest = "Peregrine Systems"
  iLength = Len(strTest) :'The value of iLength
is 17
  RetVal=iLength
```

LocalToBasicDate()

Syntaxe BASIC interne

```
Function LocalToBasicDate(strDateLocal As String)
As String
```

Description

Cette fonction convertit une date au format chaîne (telle qu'elle est affichée dans le "control panel" de Windows) en une date au format Basic.

Entrée

- *strDateLocal* : Date au format chaîne à convertir.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

LocalToBasicTime()

Syntaxe BASIC interne

```
Function LocalToBasicTime(strTimeLocal As String)
As String
```

Description

Cette fonction convertit une heure au format chaîne (telle qu'elle est affichée dans le "control panel" de Windows) en une heure au format Basic.

Entrée

- *strTimeLocal* : Heure au format chaîne à convertir.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

LocalToBasicTimeStamp()

Syntaxe BASIC interne

```
Function LocalToBasicTimeStamp(strTSLocal As
String) As String
```

Description

Cette fonction convertit un ensemble Date+Heure au format chaîne (telle qu'elle est affichée dans le "control panel" de Windows) en un ensemble Date+Heure au format Basic.

Entrée

- *`strTSLocal`* : Date+Heure au format chaîne à convertir.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

LocalToUTCDate()

Syntaxe BASIC interne

```
Function LocalToUTCDate(tmLocal As Date) As Date
```

Description

Cette fonction convertit une date au format "Date+Heure" en une date au format UTC (indépendante d'un quelconque fuseau horaire).

Entrée

- *tmLocal* : Date au format "Date+Heure".

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Log()

Syntaxe BASIC interne

```
Function Log(dValue As Double) As Double
```

Description

Renvoie le logarithme néperien d'un nombre.

Entrée

- *dValue* : Nombre dont vous souhaitez connaître le logarithme.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim dSeed as Double
  dSeed = Int((10*Rnd)-5)
  RetVal = Log(dSeed)
```

LTrim()

Syntaxe BASIC interne

```
Function LTrim(strString As String) As String
```

Description

Supprime tous les espaces précédant le premier caractère (différent d'un espace) d'une chaîne.

Entrée

- *strString* : Chaîne de caractères à traiter.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
' This example uses the LTrim and RTrim functions
 to strip leading ' and trailing spaces,
respectively, from a string variable.
' It uses the Trim function alone to strip both
types of spaces.
' LCase and UCase are also shown in this example
as well as the use
' of nested function calls

Dim strString as String
Dim strTrimString as String
  strString = "  <-Trim->  " :' Initialize string.
  strTrimString = LTrim(strString) :' strTrimString
```

```
 = "<-Trim->  ".
  strTrimString = LCase(RTrim(strString)) :'
strTrimString = "  <-trim->".
  strTrimString = LTrim(RTrim(strString)) :'
strTrimString = "<-Trim->".
  ' Using the Trim function alone achieves the
same result.
  strTrimString = UCase(Trim(strString)) :'
strTrimString = "<-TRIM->".
  RetVal= "|" & strTrimString & "|"
```

MakeInvertBool()

Syntaxe BASIC interne

```
Function MakeInvertBool(lValue As Long) As Long
```

Description

Cette fonction renvoie un booléen inversé (0 devient 1, tout autre nombre devient 0).

Entrée

- *lValue* : Nombre concerné par l'opération.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim MyValue
  MyValue=MakeInvertBool(0) :'Renvoie la valeur 1
```

```
 MyValue=MakeInvertBool(1) :'Renvoie la valeur 0
 MyValue=MakeInvertBool(254) :'Renvoie la valeur
0
```

Mid()

Syntaxe BASIC interne

```
Function Mid(strString As String, iStart As Long,
iLen As Long) As String
```

Description

Extrait une chaîne de caractères contenue dans une autre chaîne.

Entrée

- *strString* : Chaîne de caractères concernée par l'opération.
- *iStart* : Position de départ de la chaîne à extraire à l'intérieur de strString.
- *iLen* : longueur de la chaîne à extraire.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim strTest as String
  strTest="One Two Three" :' Defines the test
string
  strTest=Mid(strTest,5,3) :' strTest="Two"
  RetVal=strTest
```

Minute()

Syntaxe BASIC interne

```
Function Minute(tmTime As Date) As Long
```

Description

Renvoie le nombre de minutes contenues l'heure exprimée par le paramètre *tmTime*.

Entrée

- *tmTime* : Paramètre au format Date+Heure à traiter.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim strMinute
  strMinute=Minute(Date())
  RetVal=strMinute :'Renvoie le nombre de minutes
 écoulées dans l'heure courante par exemple "45"
s'il est actuellement 15:45:30
```

MkDir()

Syntaxe BASIC interne

```
Function MkDir(strMkDirectory As String) As Long
```

Description

Crée un répertoire.

Entrée

- *strMkDirectory* : Chemin complet du répertoire à créer.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Month()

Syntaxe BASIC interne

```
Function Month(tmDate As Date) As Long
```

Description

Renvoie le mois contenu dans la date exprimée par le paramètre *tmDate*.

Entrée

- *tmDate* : Paramètre au format Date+Heure à traiter.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim strMonth
  strMonth=Month(Date())
  RetVal=strMonth :'Renvoie le mois courant
```

Name()

Syntaxe BASIC interne

```
Function Name(strSource As String, strDest As
String)
```

Description

Renomme un fichier.

Entrée

- *strSource* : Chemin complet du fichier à renommer.
- *strDest* : Nouveau nom du fichier.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Now()

Syntaxe BASIC interne

```
Function Now() As Date
```

Description

Renvoie la date et l'heure courantes.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

NPER()

Syntaxe BASIC interne

```
Function NPER(dblRate As Double, dblPmt As Double,
dblPV As Double, dblFV As Double, iType As Long)
As Double
```

Description

Cette fonction renvoie le nombre d'échéances d'une annuité basée sur des versements constants et périodiques, et sur un taux d'intérêt fixe.

Entrée

- *dblRate* : Ce paramètre indique le taux d'intérêt par échéance. Par exemple pour un prêt à taux d'intérêt annuel de 6%, remboursé suivant une périodicité mensuelle, le taux par échéance est de :

```
0,06/12=0,005 soit 0,5%
```

- *dblPmt* : Ce paramètre indique le montant du paiement à effectuer à chaque échéance. Le paiement comporte généralement le principal et les intérêts.
- *dblPV* : Ce paramètre contient la valeur actuelle (ou somme globale) d'une série de paiements devant être effectués dans le futur.

- *dblFV* : Ce paramètre contient la valeur future ou le solde que vous souhaitez obtenir après avoir effectué le dernier paiement. En règle générale, et dans le cas d'un remboursement d'un emprunt en particulier, ce paramètre prend la valeur "0". En effet, une fois toutes les échéances remboursées, la valeur de l'emprunt est nulle.
- *iType* : Ce paramètre indique la date d'échéances des paiements. Il peut prendre les valeurs suivantes :
  - *0* si les paiements sont dus à terme échu (c'est à dire en fin de période)
  - *1* si les paiements sont dus à terme à échoir (c'est à dire en début de période)

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Remarques

Note : Les sommes versées (exprimées notamment par le paramètre Pmt) sont représentées par des nombres négatifs. Les sommes reçues sont représentées par des nombres positifs.

Oct()

Syntaxe BASIC interne

```
Function Oct(dValue As Double) As String
```

Description

Renvoie la valeur octale d'un nombre.

Entrée

- *dValue* : Nombre dont vous souhaitez connaître la valeur octale.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim dSeed as Double
  dSeed = Int((10*Rnd)-5)
  RetVal = Oct(dSeed)
```

ParseDate()

Syntaxe BASIC interne

```
Function ParseDate(strDate As String, strFormat As
String, strStep As String) As Date
```

Description

Cette fonction convertit une date exprimée sous la forme d'une chaîne de caractères en un objet date au sens Basic du term.

Entrée

- *strDate* : Date au format chaîne de caractères.
- *strFormat* : Ce paramètre contient le format de la date contenue dans la chaîne de caractères. Les valeurs possibles sont les suivantes :
  - DD/MM/YY
  - DD/MM/YYYY

- MM/DD/YY
- MM/DD/YYYY
- YYYY/MM/DD
- Date : date exprimée suivant les paramètres de date du poste client.
- DateInter : date exprimée au format international

- *strStep* : Ce paramètre optionnel contient le séparateur de date utilisé dans la chaîne de caractères. Les séparateurs autorisés sont "\" et "-".

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim dDate as date
  dDate=ParseDate("2001/05/01", "YYYY/MM/DD")
```

ParseDMYDate()

Syntaxe BASIC interne

```
Function ParseDMYDate(strDate As String) As Date
```

Description

Cette fonction renvoie un objet Date (au sens Basic) à partir d'une date formatée comme suit :

```
jj/mm/aaaa
```

Entrée

- *strDate* : Date stockée sous la forme d'une chaîne.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

ParseMDYDate()

Syntaxe BASIC interne

```
Function ParseMDYDate(strDate As String) As Date
```

Description

Cette fonction renvoie un objet Date (au sens Basic) à partir d'une date formatée comme suit :

```
mm/jj/aaaa
```

Entrée

- *strDate* : Date stockée sous la forme d'une chaîne.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

ParseYMDDate()

Syntaxe BASIC interne

```
Function ParseYMDDate(strDate As String) As Date
```

Description

Cette fonction convertit une chaîne de caractères représentant une date au format aaaa/mm/jj en une variable Basic de type Date.

Entrée

- *strDate* : Date stockée sous la forme d'une chaîne.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

PifFirstInCol()

Syntaxe BASIC interne

```
Function PifFirstInCol(strPathCol As String,
strChildCond As String, iStartCount As Long) As
Long
```

Description

Cette fonction renvoie le numéro du premier élément d'une collection qui remplit la condition exprimée dans le paramètre *strChildCond*.

Entrée

- *strPathCol* : Nom (chemin) de la collection sur laquelle s'effectue la recherche.
- *strChildCond* : Condition de recherche sur l'élément.
-

Note : Si n représente le nombre d'éléments de la collection, iStartCount doit être compris entre 0 et n-1.

iStartCount : Numéro (index) à partir duquel la recherche commence.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Exemple

```
Dim iToTal As Integer
Dim iIndex As Integer
iToTal = 0
iIndex = 0

Do
  iIndex = PifFirstInCol("Software",
"Brand=Peregrine", 0)
  If iIndex
```

PifGetBlobSize()

Syntaxe BASIC interne

```
Function PifGetBlobSize(strPath As String) As Long
```

Description

Cette fonction renvoie la taille d'un objet blob.

Entrée

- *strPath* : Ce paramètre contient le chemin complet de l'objet blob dans la collection.

Sortie

La fonction renvoie la taille de l'objet blob identifié par son chemin complet.

Exemple

```
Dim iSize a Integer
iSize = PifGetBlobSize("Description")
```

PifGetElementChildName()

Syntaxe BASIC interne

```
Function PifGetElementChildName(strPath As String,
iItem As Long) As String
```

Description

Cette fonction renvoie le nom du ième sous-élément d'un noeud identifié d'un type de document source.

Entrée

- *strPath* : Ce paramètre contient le chemin complet du noeud concerné par l'opération.
- *iItem* : Ce paramètre contient le numéro du sous-élément dont vous souhaitez récupérer le nom.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

L'exemple suivant renvoie le nom du troisième sous élément du noeud "MyDocument.MyNode" :

```
dim iRc as integer
  iRc =
pifGetElementChildName("MyDocument.MyNode",3)
```

PifGetElementCount()

Syntaxe BASIC interne

```
Function PifGetElementCount(strPath As String) As
Long
```

Description

Cette fonction renvoie le nombre de sous-éléments d'un noeud identifié d'un type de document source.

Entrée

- *strPath* : Ce paramètre contient le chemin complet du noeud concerné par l'opération.

Sortie

La fonction renvoie le nombre de sous-élements du noeud dont le chemin est précisé par *strPath*.

Exemple

```
dim iChildCount as integer
iChildCount = PifGetElementCount("Asset")
```

PifGetInstance()

Syntaxe BASIC interne

```
Function PifGetInstance() As String
```

Description

Cette fonction renvoie le numéro de l'élément actuellement traité au sein de la collection. Le premier élément traité a pour numéro "0".

Sortie

Le numéro de l'élément traité est renvoyé sous la forme d'une chaîne de caractères.

Exemple

```
Dim strMyElement as String
strMyElement= "Item #" & Cstr(PifGetInstance()+1)
```

PifGetItemCount()

Syntaxe BASIC interne

```
Function PifGetItemCount(strPath As String) As Long
```

Description

Cette fonction renvoie le nombre d'éléments dans une collection identifiée par son chemin d'accès.

Entrée

- *strPath* : Chemin d'accès complet de la collection concernée par l'opération.

Sortie

Si la collection n'existe pas, la fonction renvoie la valeur "0".

PifIgnoreDocumentMapping()

Syntaxe BASIC interne

```
Function PifIgnoreDocumentMapping(strMsg As String)
As Long
```

Description

Cette fonction permet d'ignorer le traitement d'un document.

Un message d'information, contenu dans le paramètre *strMsg*, peut être envoyé dans le journal.

Entrée

- *strMsg* : Paramètre optionnel. Chaîne de caractères envoyée dans le journal.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Exemple

```
If [BarCode] = "" Then
  PifIgnoreDocumentMapping([AssetTag])
Else
  RetVal = [BarCode]
End If
```

PifIgnoreNodeMapping()

Syntaxe BASIC interne

```
Function PifIgnoreNodeMapping(strMsg As String) As
Long
```

Description

Cette fonction permet d'ignorer le traitement d'un noeud d'un document.

Un message d'information, contenu dans le paramètre *strMsg*, peut être envoyé dans le journal.

Entrée

- *strMsg* : Paramètre optionnel. Chaîne de caractères envoyée dans le journal.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Exemple

```
If [Comment] = "" Then
  PifIgnoreNodeMapping("Le noeud courant n'a pas
été traité")
Else
  RetVal = [Comment]
End If
```

Remarques

Note : Le noeud n'étant pas traité, le message envoyé dans le journal est reporté sur le parent du noeud ignoré.

PifIsInMap()

Syntaxe BASIC interne

```
Function PifIsInMap(strKey As String, strMaptable
As String, bCaseSensitive As Long) As Long
```

Description

Cette fonction teste si un mot-clé donné appartient à une table de correspondance. La recherche du mot-clé peut être effectuée en respectant la casse.

Entrée

- *strKey* : Nom du mot-clé sur lequel porte la recherche.
- *strMaptable* : Nom de la table de correspondance dans laquelle s'effectue la recherche.
- *bCaseSensitive* : Ce paramètre précise si la recherche tient compte de la casse ou non.
  - *0* :La recherche ne tient pas compte de la casse.
  - *1* :La recherche tient compte de la casse.

Sortie

- 0 : Le mot-clé n'a pas été trouvé.
- 1 : Le mot-clé a été trouvé.

Exemple

```
If PifIsInMap("CAT_PC", "MainAsset") Then
  RetVal = 1
Else
```

```
  RetVal = 0
End If
```

PifLogInfoMsg()

Syntaxe BASIC interne

```
Function PifLogInfoMsg(strMsg As String) As Long
```

Description

Cette fonction envoie un message d'information, contenu dans le paramètre *strMsg*, dans le journal.

Entrée

- *strMsg* : Chaîne de caractères contenant le message d'information à envoyer dans le journal.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Exemple

```
Dim strBrand As String
strBrand = [DeviceBrand]
If strBrand = "" Then
  PifLogInfoMsg(PifStrVal("BRAND_UNREGISTERED"))
  RetVal = PifStrVal("BRAND_UNKNOWN")
Else
  RetVal = strBrand
End If
```

PifLogWarningMsg()

Syntaxe BASIC interne

```
Function PifLogWarningMsg(strMsg As String) As Long
```

Description

Cette fonction envoie un message d'avertissement, contenu dans le paramètre *strMsg*, dans le journal.

Entrée

- *strMsg* : Chaîne de caractères contenant le message d'avertissement à envoyer dans le journal.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Exemple

```
If [Brand] = "" Then
  PifLogWarningMsg("Marque inconnue")
Else
  RetVal = [Brand]
End if
```

PifMapValue()

Syntaxe BASIC interne

```
Function PifMapValue(strKey As String, strMaptable
As String, iPos As Long, strDefault As String,
bCaseSensitive As Long) As String
```

Description

Cette fonction renvoie la valeur d'un élément au sein d'une table de correspondance. L'élément est identifié de façon unique au moyen des paramètres *strKey*, *strMapTable*, *iPos*.

La recherche de l'élément peut tenir compte ou non de la casse.

Entrée

- *strKey* : Mot-clé permettant d'identifier la ligne de la table de correspondance qui contient l'élément.
- *strMaptable* : Nom de la table de correspondance dans laquelle s'effectue la recherche.
- *iPos* : Numéro de la colonne dans laquelle se trouve l'élément dont on veut récupérer la valeur.
- *strDefault* : Valeur par défaut renvoyée si l'élément n'est pas trouvé.
-
  - *0* :La recherche ne tient pas compte de la casse.
  - *1* :La recherche tient compte de la casse.

  bCaseSensitive : Ce paramètre précise si la recherche tient compte de la casse ou non.

Sortie

La fonction renvoie la chaîne trouvée ou la chaîne par défaut (celle contenue dans le paramètre *strDefault*) si l'élément n'est pas trouvé.

Exemple

```
RetVal =
PifMapValue([Link_ProductOID.Language],"Language",
 1, PifStrVal("TSC_UNKNOWN"), 0)
```

PifMapValueContaining()

Syntaxe BASIC interne

```
Function PifMapValueContaining(strKey As String,
strMaptable As String, iPos As Long, strDefault As
String, bCaseSensitive As Long) As String
```

Description

Cette fonction renvoie la valeur d'un élément au sein d'une table de correspondance. L'élément est identifié de façon unique au moyen des paramètres *strKey*, *strMapTable*, *iPos*.

La recherche de l'élément peut tenir compte ou non de la casse.

A la difference de la fonction PifMapValue, le paramètre *strKey* peut être un sur-ensemble du mot-clé recherché.

Par exemple si *strKey* contient "Moniteur", l'élément de mot-clé "Cat_Moniteur" sera trouvé.

Entrée

- *strKey* : Mot-clé ou sur-ensemble du mot-clé permettant d'identifier la ligne de la table de correspondance qui contient l'élément.
- *strMaptable* : Nom de la table de correspondance dans laquelle s'effectue la recherche.
- *iPos* : Numéro de la colonne dans laquelle se trouve l'élément dont on veut récupérer la valeur.

- *strDefault* : Valeur par défaut renvoyée si l'élément n'est pas trouvé.
-
  - *0* :La recherche ne tient pas compte de la casse.
  - *1* :La recherche tient compte de la casse.

  bCaseSensitive : Ce paramètre précise si la recherche tient compte de la casse ou non.

Sortie

La fonction renvoie la chaîne trouvée ou la chaîne par défaut (celle contenue dans le paramètre *strDefault*) si l'élément n'est pas trouvé.

Exemple

```
RetVal =
PifMapValueContaining([COMPUTER_MODEL_T.CMP_MODEL],
 "Brand", 1, PifStrVal("BRAND_UNKNOWN"))
```

PifNewQueryFromFmtName()

Syntaxe BASIC interne

```
Function PifNewQueryFromFmtName(strCntrName As
String, strFmtName As String, strLayer As String)
As Long
```

Description

Cette fonction crée une requête sur un type de document préalablement défini dans la liste des documents produits par une ressource.

Entrée

- *strCntrName* : Ce paramètre contient le nom de la ressource (celle sur laquelle la requête est effectuée).
- *strFmtName* : Ce paramètre contient l'identifiant du type de document (préalablement défini comme type de document produit).
- *strLayer* : Ce paramètre contient la clause WHERE d'une requête AQL.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
hQuery = PifNewQueryFromFmtName("Asset Management",
 "amEmplDept", "Name like 'A%'")
```

PifNewQueryFromXml()

Syntaxe BASIC interne

```
Function PifNewQueryFromXml(strCntrName As String,
strQuery As String, strLayer As String) As Long
```

Description

Cette fonction crée une requête pour une ressource. Le type de document doit être intégralement défini sous forme XML par le paramètre *strQuery*. Le traitement (clause d'une requête AQL) est défini sous forme XML par le paramètre *strQuery*.

Entrée

- *strCntrName* : Ce paramètre contient le nom de la ressource (celle sur laquelle la requête est effectuée).
- *strQuery* : Ce paramètre contient un document XML qui définit le type de document (attributs, structures, collections) sur lequel s'effectue la requête.
- *strLayer* : Ce paramètre contient un document XML qui définit le traitement (clause WHERE, ORDER BY, ...) de la requête.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
hQuery = PifNewQueryFromXML ("Asset Management",
strQuery, strLayer)
```

PifNodeExists()

Syntaxe BASIC interne

```
Function PifNodeExists(strPath As String) As Long
```

Description

Cette fonction permet de tester si un noeud, identifié par son chemin d'accès complet, existe au sein d'un document produit.

Entrée

- *strPath* : Chemin d'accès complet du noeud concerné par l'opération.

Sortie

La fonction renvoie l'une des valeurs suivantes :

- *0* :si le noeud n'existe pas.
- *1* :si le noeud existe

Exemple

```
If not
PifNodeExists("Hardware.Peripherials.Printer")
Then
  PifIgnoreNodeMapping
End If
```

PifQueryClose()

Syntaxe BASIC interne

```
Function PifQueryClose(lQueryHandle As Long) As
Long
```

Description

Cette fonction ferme la requête et libère toutes les ressources internes utilisées par la requête.

Entrée

- *lQueryHandle* : Ce paramètre contient un handle sur la requête créée au moyen des fonctions `PifNewQueryFromFmtName()` ou `PifNewQueryFromXml()`.

Sortie

- 0 : La fonction s'est exécutée normalement.

- Non nul : Code d'erreur.

PifQueryGetDateVal()

Syntaxe BASIC interne

```
Function PifQueryGetDateVal(lQueryHandle As Long,
strPath As String) As Date
```

Description

Cette fonction renvoie la valeur (sous la forme d'une Date) d'un noeud du document courant. Le document courant est celui sur lequel le curseur de la requête a été fixé au moyen de la fonction `PifQueryNext()`.

Entrée

- *`lQueryHandle`* : Ce paramètre contient un handle sur la requête créée au moyen des fonctions `PifNewQueryFromFmtName()` ou `PifNewQueryFromXml()`.
- *`strPath`* : Ce paramètre contient le chemin du noeud du document courant dont vous souhaitez récupérer la valeur.

Sortie

Valeur du noeud identifié

PifQueryGetDoubleVal()

Syntaxe BASIC interne

```
Function PifQueryGetDoubleVal(lQueryHandle As Long,
strPath As String) As Double
```

Description

Cette fonction renvoie la valeur (sous la forme d'un Double) d'un noeud du document courant. Le document courant est celui sur lequel le curseur de la requête a été fixé au moyen de la fonction `PifQueryNext()`.

Entrée

- *`lQueryHandle`* : Ce paramètre contient un handle sur la requête créée au moyen des fonctions `PifNewQueryFromFmtName()` ou `PifNewQueryFromXml()`.
- *`strPath`* : Ce paramètre contient le chemin du noeud du document courant dont vous souhaitez récupérer la valeur.

Sortie

Valeur du noeud identifié

PifQueryGetIntVal()

Syntaxe BASIC interne

```
Function PifQueryGetIntVal(lQueryHandle As Long,
strPath As String) As Long
```

Description

Cette fonction renvoie la valeur (sous la forme d'un Entier) d'un noeud du document courant. Le document courant est celui sur lequel le curseur de la requête a été fixé au moyen de la fonction `PifQueryNext()`.

Entrée

- *lQueryHandle* : Ce paramètre contient un handle sur la requête créée au moyen des fonctions `PifNewQueryFromFmtName()` ou `PifNewQueryFromXml()`.
- *strPath* : Ce paramètre contient le chemin du noeud du document courant dont vous souhaitez récupérer la valeur.

Sortie

Valeur du noeud identifié

PifQueryGetLongVal()

Syntaxe BASIC interne

```
Function PifQueryGetLongVal(lQueryHandle As Long,
strPath As String) As Long
```

Description

Cette fonction renvoie la valeur (sous la forme d'un Long) d'un noeud du document courant. Le document courant est celui sur lequel le curseur de la requête a été fixé au moyen de la fonction `PifQueryNext()`.

Entrée

- *lQueryHandle* : Ce paramètre contient un handle sur la requête créée au moyen des fonctions `PifNewQueryFromFmtName()` ou `PifNewQueryFromXml()`.
- *strPath* : Ce paramètre contient le chemin du noeud du document courant dont vous souhaitez récupérer la valeur.

Sortie

Valeur du noeud identifié

Exemple

```
strValue = PifQueryGetLongVal(hQuery, "Name")
```

PifQueryGetStringVal()

Syntaxe BASIC interne

```
Function PifQueryGetStringVal(lQueryHandle As Long,
strPath As String) As String
```

Description

Cette fonction renvoie la valeur (sous la forme d'une chaîne) d'un noeud du document courant. Le document courant est celui sur lequel le curseur de la requête a été fixé au moyen de la fonction `PifQueryNext()`.

Entrée

- *`lQueryHandle`* : Ce paramètre contient un handle sur la requête créée au moyen des fonctions `PifNewQueryFromFmtName()` ou `PifNewQueryFromXml()`.
- *`strPath`* : Ce paramètre contient le chemin du noeud du document courant dont vous souhaitez récupérer la valeur.

Sortie

Valeur du noeud identifié

Exemple

```
strValue = PifQueryGetStringVal(hQuery, "Name")
```

PifQueryNext()

Syntaxe BASIC interne

```
Function PifQueryNext(lQueryHandle As Long) As Long
```

Description

Cette fonction fixe le curseur de la requête sur le prochain résultat. Le curseur n'est pas fixé sur le premier document après un appel aux fonctions `PifNewQueryFromFmtName()` ou `PifNewQueryFromXml()`. L'utilisateur doit faire appel à la fonction `PifQueryNext()` pour avoir accès aux valeurs du document courant avec une des fonctions suivantes :

- `PifQueryGetStringVal()`
- `PifQueryGetDateVal()`
- `PifQueryGetDoubleVal()`
- `PifQueryGetLongVal()`
- `PifQueryGetIntVal()`

Entrée

- *`lQueryHandle`* : Ce paramètre contient un handle sur la requête créée au moyen des fonctions `PifNewQueryFromFmtName()` ou `PifNewQueryFromXml()`.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Remarques

La fonction sort en erreur lorsqu'il n'y a plus de données à parcourir (code d'erreur -2003).

PifRejectDocumentMapping()

Syntaxe BASIC interne

```
Function PifRejectDocumentMapping(strMsg As String)
As Long
```

Description

Cette fonction rejette un document. Le document n'est pas transmis au connecteur suivant.

Un message d'information, contenu dans le paramètre *strMsg*, peut être envoyé dans le journal.

Entrée

- *strMsg* : Paramètre optionnel. Chaîne de caractères envoyée dans le journal.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Exemple

```
Dim strNetAddress As String
strNetAddress = [Hardware.TCPIP.PhysicalAddress]

If strNetAddress = "" Then
```

```
  PifRejectDocumentMapping("Document rejected:
missing MAC address")
Else
  RetVal = strNetAddress
End If
```

PifRejectNodeMapping()

Syntaxe BASIC interne

```
Function PifRejectNodeMapping(strMsg As String) As
Long
```

Description

Un message d'information, contenu dans le paramètre *strMsg*, peut être envoyé dans le journal.

Entrée

- *strMsg* : Paramètre optionnel. Chaîne de caractères envoyée dans le journal.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Exemple

```
If [Location.Name] = "" Then

PifRejectNodeMapping(PifStrVal("UNKNOWN_LOCATION"))
End If
```

PifSetDateVal()

Syntaxe BASIC interne

```
Function PifSetDateVal(strPath As String, dttVal
As Date) As Long
```

Description

Cette fonction permet de fixer la valeur d'un noeud dans le document cible. Si le noeud n'existe pas, la fonction le crée.

Entrée

- *strPath* : Ce paramètre contient le chemin complet du noeud concerné par l'opération.
- *dttVal* : Ce paramètre contient la valeur (date) que vous souhaitez affecter au noeud.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Exemple

```
Dim dtCurrent as Date
dtCurrent = Date()
PifSetDateVal("ValueDate", dtCurrent)
```

PifSetDoubleVal()

Syntaxe BASIC interne

```
Function PifSetDoubleVal(strPath As String, dVal
As Double) As Long
```

Description

Cette fonction permet de fixer la valeur d'un noeud dans le document cible. Si le noeud n'existe pas, la fonction le crée.

Entrée

- *strPath* : Ce paramètre contient le chemin complet du noeud concerné par l'opération.
- *dVal* : Ce paramètre contient la valeur (double) que vous souhaitez affecter au noeud.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Exemple

```
Dim d as Double
d = 2.5
PifSetDoubleVal("ValueDouble", d)
```

PifSetLongVal()

Syntaxe BASIC interne

```
Function PifSetLongVal(strPath As String, lVal As
Long) As Long
```

Description

Cette fonction permet de fixer la valeur d'un noeud dans le document cible. Si le noeud n'existe pas, la fonction le crée.

Entrée

- *strPath* : Ce paramètre contient le chemin complet du noeud concerné par l'opération.
- *lVal* : Ce paramètre contient la valeur (entier long) que vous souhaitez affecter au noeud.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Exemple

```
Dim long as Long
l = 2
PifSetLongVal("ValueLong", l)
```

PifSetStringVal()

Syntaxe BASIC interne

```
Function PifSetStringVal(strPath As String, strVal
As String) As Long
```

Description

Cette fonction permet de fixer la valeur d'un noeud dans le document cible. Si le noeud n'existe pas, la fonction le crée.

Entrée

- *strPath* : Ce paramètre contient le chemin complet du noeud concerné par l'opération.
- *strVal* : Ce paramètre contient la valeur (chaîne de caractères) que vous souhaitez affecter au noeud.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Exemple

```
Dim str as Long
str = "UNKNOWN"
PifSetStringVal("ValueString", str)
```

PifStrVal()

Syntaxe BASIC interne

```
Function PifStrVal(strID As String) As String
```

Description

Cette fonction renvoie la chaîne de caractères associée à l'identifiant contenu dans le paramètre *strId*

Entrée

- *strID* : Identifiant de la chaîne de caractères à récupérer.

Sortie

Si l'identifiant n'est pas trouvé, la fonction renvoie une chaîne vide et inscrit une erreur dans le journal de Connect-It. Si l'identifiant est trouvé, la fonction renvoie la chaîne de caractères qui lui est associée.

Exemple

```
If [DeviceType] = "" Then
  RetVal = PifStrVal("BRAND_UNKNOWN")
End If
```

PifUserFmtStrToVar()

Syntaxe BASIC interne

```
Function PifUserFmtStrToVar(strData As String,
strUserFmtName As String) As Variant
```

Description

Cette fonction traite une chaîne de caractères en fonction d'un format prédéfini au moyen d'un assistant dans l'interface graphique de Connect-It, et renvoie un nombre variant de type Date ou Nombre en fonction de la nature du format prédéfini.

Entrée

- *strData* : Ce paramètre contient la chaîne à traiter.
- *strUserFmtName* : Ce paramètre contient le nom du format prédéfini.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

Si l'on considère les deux formats prédéfinis suivants :

- origine = "yyy'-'mm'-'dd"
- destination = "Le 'dddd' 'dd' 'mmmm' 'yyyy'"

Alors le script :

```
Dim dTmp as Variant

dTmp=PifUserFmtVarToStr([date_modified],"origine")
  RetVal = PifUserFmtStrToVar(dTmp,"destination")
```

Renvoie pour [date_modified]= 2001-05-30 la valeur "Le jeudi 30 mai 2001"

Remarques

Note : Pour plus d'information sur les formats prédéfinis, consultez le Guide Utilisateur de Connect-It

PifUserFmtVarToStr()

Syntaxe BASIC interne

```
Function PifUserFmtVarToStr(vData As Variant,
strUserFmtName As String) As String
```

Description

Cette fonction traite un variant en fonction d'un format prédéfini et renvoie une chaîne de caractères.

Entrée

- *vData* : Ce paramètre contient le variant traité par la fonction.
- *strUserFmtName* : Ce paramètre contient le nom du format prédéfini.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

Si l'on considère les deux formats prédéfinis suivants :

- origine = "yyy'-'mm'-'dd"

- destination = "Le 'dddd' 'dd' 'mmmm' 'yyyy'"

Alors le script :

```
Dim dTmp as Variant

dTmp=PifUserFmtVarToStr([date_modified],"origine")
  RetVal = PifUserFmtStrToVar(dTmp,"destination")
```

Renvoie pour [date_modified]= 2001-05-30 la valeur "Le jeudi 30 mai 2001"

Remarques

Note : Pour plus d'information sur les format prédéfinis, consultez le Guide Utilisateur de Connect-It

PMT()

Syntaxe BASIC interne

```
Function PMT(dblRate As Double, iNper As Long,
dblPV As Double, dblFV As Double, iType As Long)
As Double
```

Description

Cette fonction renvoie le montant d'une annuité basée sur des versements constants et périodiques, et sur un taux d'intérêt fixe.

Entrée

- *dblRate* : Ce paramètre indique le taux d'intérêt par échéance. Par exemple pour un prêt à taux d'intérêt annuel de 6%, remboursé suivant une périodicité mensuelle, le taux par échéance est de :

```
0,06/12=0,005 soit 0,5%
```

- *iNper* : Ce paramètre contient le nombre total d'échéances de l'opération financière.
- *dblPV* : Ce paramètre contient la valeur actuelle (ou somme globale) d'une série de paiements devant être effectués dans le futur.
- *dblFV* : Ce paramètre contient la valeur future ou le solde que vous souhaitez obtenir après avoir effectué le dernier paiement. En règle générale, et dans le cas d'un remboursement d'un emprunt en particulier, ce paramètre prend la valeur "0". En effet, une fois toutes les échéances remboursées, la valeur de l'emprunt est nulle.
- *iType* : Ce paramètre indique la date d'échéances des paiements. Il peut prendre les valeurs suivantes :
  - *0* si les paiements sont dus à terme échu (c'est à dire en fin de période)
  - *1* si les paiements sont dus à terme à échoir (c'est à dire en début de période)

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Remarques

Note : Les paramètres Rate et Nper doivent être calculés à l'aide d'échéances exprimées dans les mêmes unités.Les sommes versées (exprimées notamment par le paramètre Pmt) sont

représentées par des nombres négatifs. Les sommes reçues sont représentées par des nombres positifs.

PPMT()

Syntaxe BASIC interne

```
Function PPMT(dblRate As Double, iPer As Long,
iNper As Long, dblPV As Double, dblFV As Double,
iType As Long) As Double
```

Description

Cette fonction renvoie le montant du remboursement du capital, pour une échéance donnée, d'une annuité basée sur des versements constants et périodiques et sur un taux d'intérêt fixe.

Entrée

- *dblRate* : Ce paramètre indique le taux d'intérêt par échéance. Par exemple pour un prêt à taux d'intérêt annuel de 6%, remboursé suivant une périodicité mensuelle, le taux par échéance est de :

```
0,06/12=0,005 soit 0,5%
```

- *iPer* : Ce paramètre indique la période concernée par le calcul, comprise entre 1 et la valeur du paramètre *Nper*.
- *iNper* : Ce paramètre contient le nombre total d'échéances de l'opération financière
- *dblPV* : Ce paramètre contient la valeur actuelle (ou somme globale) d'une série de paiements devant être effectués dans le futur.
- *dblFV* : Ce paramètre contient la valeur future ou le solde que vous souhaitez obtenir après avoir effectué le dernier paiement. En règle générale, et dans le cas d'un remboursement d'un emprunt en

particulier, ce paramètre prend la valeur "0". En effet, une fois toutes les échéances remboursées, la valeur de l'emprunt est nulle.

- *iType* : Ce paramètre indique la date d'échéances des paiements. Il peut prendre les valeurs suivantes :
  - *0* si les paiements sont dus à terme échu (c'est à dire en fin de période)
  - *1* si les paiements sont dus à terme à échoir (c'est à dire en début de période)

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Remarques

Note : Les paramètres Rate et Nper doivent être calculés à l'aide d'échéances exprimées dans les mêmes unités.Les sommes versées (exprimées notamment par le paramètre Pmt) sont représentées par des nombres négatifs. Les sommes reçues sont représentées par des nombres positifs.

PV()

Syntaxe BASIC interne

```
Function PV(dblRate As Double, iNper As Long,
dblPmt As Double, dblFV As Double, iType As Long)
As Double
```

Description

Cette fonction renvoie le montant actuel d'une annuité basée sur des échéances futures constantes et périodiques, et sur un taux d'intérêt fixe.

Entrée

- *dblRate* : Ce paramètre indique le taux d'intérêt par échéance. Par exemple pour un prêt à taux d'intérêt annuel de 6%, remboursé suivant une périodicité mensuelle, le taux par échéance est de :

```
0,06/12=0,005 soit 0,5%
```

- *iNper* : Ce paramètre contient le nombre total d'échéances de l'opération financière.
- *dblPmt* : Ce paramètre indique le montant du paiement à effectuer à chaque échéance. Le paiement comporte généralement le principal et les intérêts.
- *dblFV* : Ce paramètre contient la valeur future ou le solde que vous souhaitez obtenir après avoir effectué le dernier paiement. En règle générale, et dans le cas d'un remboursement d'un emprunt en particulier, ce paramètre prend la valeur "0". En effet, une fois toutes les échéances remboursées, la valeur de l'emprunt est nulle.
- *iType* : Ce paramètre indique la date d'échéances des paiements. Il peut prendre les valeurs suivantes :
  - *0* si les paiements sont dus à terme échu (c'est à dire en fin de période)
  - *1* si les paiements sont dus à terme à échoir (c'est à dire en début de période)

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Remarques

Note : Les paramètres Rate et Nper doivent être calculés à l'aide d'échéances exprimées dans les mêmes unités.Les sommes versées (exprimées notamment par le paramètre Pmt) sont représentées par des nombres négatifs. Les sommes reçues sont représentées par des nombres positifs.

Randomize()

Syntaxe BASIC interne

```
Function Randomize(lValue As Long)
```

Description

Initialise le générateur de nombres aléatoires.

Entrée

- *lValue* : Paramètre optionnel utilisé pour initialiser le générateur de nombres aléatoires de la fonction Rnd en lui donnant une nouvelle valeur initiale. Si ce paramètre est omis, la valeur renvoyée par l'horloge système est utilisée comme valeur initiale.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim MyNumber
  Randomize
  MyNumber= Int((10*Rnd)+1) :'Renvoie une valeur
aléatoire comprise entre 1 et 10.
  RetVal=MyNumber
```

RATE()

Syntaxe BASIC interne

```
Function RATE(iNper As Long, dblPmt As Double,
dblFV As Double, dblPV As Double, iType As Long,
dblGuess As Double) As Double
```

Description

Cette fonction renvoie le taux d'intérêt par échéance pour une annuité.

Entrée

- *iNper* : Ce paramètre contient le nombre total d'échéances de l'opération financière.
- *dblPmt* : Ce paramètre indique le montant du paiement à effectuer à chaque échéance. Le paiement comporte généralement le principal et les intérêts.
- *dblFV* : Ce paramètre contient la valeur future ou le solde que vous souhaitez obtenir après avoir effectué le dernier paiement. En règle générale, et dans le cas d'un remboursement d'un emprunt en particulier, ce paramètre prend la valeur "0". En effet, une fois toutes les échéances remboursées, la valeur de l'emprunt est nulle.
- *dblPV* : Ce paramètre contient la valeur actuelle (ou somme globale) d'une série de paiements devant être effectués dans le futur.

- *iType* : Ce paramètre indique la date d'échéances des paiements. Il peut prendre les valeurs suivantes :
  - *0* si les paiements sont dus à terme échu (c'est à dire en fin de période)
  - *1* si les paiements sont dus à terme à échoir (c'est à dire en début de période)
- *dblGuess* : Ce paramètre contient la valeur estimée du taux d'intérêt par échéance.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Remarques

Note : Les sommes versées (exprimées notamment par le paramètre Pmt) sont représentées par des nombres négatifs. Les sommes reçues sont représentées par des nombres positifs.Cette fonction effectue les calculs par itération, en commençant par la valeur attribuée au paramètre Guess. Si aucun résultat n'est trouvé au bout de vingt itérations, la fonction échoue.

RemoveRows()

Syntaxe BASIC interne

```
Function RemoveRows(strList As String, strRowNames
As String) As String
```

Description

Supprime dans une liste les lignes identifiées par le paramètre *strRowNames*.

Cette fonction est utile lors du traitement des valeurs d'un contrôle de type "ListBox". Les valeurs d'un tel contrôle sont représentées par des chaînes bi-dimensionnelles dont les caractéristiques sont les suivantes :

- Le caractère "|" est utilisé comme séparateur de colonnes.
- Le caractère "," est utilisé comme séparateur de lignes.
- Chaque ligne est terminée par un identifiant unique situé à droite du signe "="

Entrée

- *strList* : Chaîne source contenant les valeurs d'un contrôle "ListBox" à traiter.
- *strRowNames* : Identifiants des lignes à supprimer. Les identifiants sont séparés par des virgules.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim MyStr
  MyStr=RemoveRows("a1|a2=a0,b1|b2=b0", "a0,c0")
:'Renvoie "b1|b2=b0"
  RetVal=MyStr
```

Replace()

Syntaxe BASIC interne

```
Function Replace(strData As String, strOldPattern
As String, strNewPattern As String, bCaseSensitive
As Long) As String
```

Description

Remplace toutes les occurrences du paramètre *strOldPattern* par la valeur du paramètre *strNewPattern* au sein de la chaîne de caractères contenue dans *strData*. La recherche de *strOldPattern* peut tenir compte ou non de la casse en fonction de la valeur du paramètre *bCaseSensitive*.

Entrée

- *strData* : Chaîne de caractères contenant les occurrences à remplacer.
- *strOldPattern* : Occurrence à recherche dans la chaîne de caractères contenue dans *strData*.
- *strNewPattern* : Texte remplaçant toute occurrence trouvée.
- *bCaseSensitive* : En fonction de la valeur de ce paramètre, la recherche respecte (=1) ou non (=0) la casse.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim MyStr
  MyStr=Replace("toimoitoimoitoi", "toi", "moi",0)
```

```
:'Renvoie "moimoimoimoimoi"
 MyStr=Replace("toimoitoimoitoi", "Toi", "moi",1)
:'Renvoie "toimoitoimoitoi"
 MyStr=Replace("toimoiToimoitoi", "Toi", "moi",1)
:'Renvoie "toimoimoimoitoi"
 RetVal=""
```

Right()

Syntaxe BASIC interne

```
Function Right(strString As String, iNumber As
Long) As String
```

Description

Renvoie iNumber caractères d'une chaîne en partant de la droite.

Entrée

- *strString* : Chaîne de caractères à traiter.
- *iNumber* : Nombre de caractères à renvoyer.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim lWord, strMsg, rWord, iPos :' Declare
variables.
  strMsg = "Left() Test."
  iPos = InStr(1, strMsg, " ") :' Find space.
```

```
 lWord = Left(strMsg, iPos - 1) :' Get left word.
 rWord = Right(strMsg, Len(strMsg) - iPos) :' Get
right word.
 strMsg=rWord+lWord :' And swap them
 RetVal=strMsg
```

RightPart()

Syntaxe BASIC interne

```
Function RightPart(strFrom As String, strSep As
String, bCaseSensitive As Long) As String
```

Description

Extrait la portion d'une chaîne de caractères située à droite du séparateur précisé dans le paramètre *strSep*.

La recherche du séparateur s'effectue de la droite vers la gauche.

La recherche peut tenir compte ou non de la casse en fonction de la valeur du paramètre *bCaseSensitive*.

Entrée

- *strFrom* : Chaîne source à traiter.
- *strSep* : Caractère utilisé comme séparateur dans la chaîne source.
- *bCaseSensitive* : En fonction de la valeur de ce paramètre, la recherche respecte (=1) ou non (=0) la casse.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

Ces exemples illustrent l'utilisation des fonctions `LeftPart`, `LeftPartFromRight`, `RightPart` et `RightPartFromLeft` sur une même chaîne de caractères: "Ceci_est_un_test" :

```
LeftPart("Ceci_est_un_test","_",0)
```

Renvoie la chaîne "Ceci".

```
LeftPartFromRight("Ceci_est_un_test","_",0)
```

Renvoie la chaîne "Ceci_est_un".

```
RightPart("Ceci_est_un_test","_",0)
```

Renvoie la chaîne "test".

```
RightPartFromLeft("Ceci_est_un_test","_",0)
```

Renvoie la chaîne "est_un_test".

RightPartFromLeft()

Syntaxe BASIC interne

```
Function RightPartFromLeft(strFrom As String,
strSep As String, bCaseSensitive As Long) As String
```

Description

Extrait la portion d'une chaîne de caractères située à droite du séparateur précisé dans le paramètre *`strSep`*.

La recherche du séparateur s'effectue de la gauche vers la droite.

La recherche peut tenir compte ou non de la casse en fonction de la valeur du paramètre *`bCaseSensitive`*.

Entrée

- *`strFrom`* : Chaîne source à traiter.

- *strSep* : Caractère utilisé comme séparateur dans la chaîne source.
- *bCaseSensitive* : En fonction de la valeur de ce paramètre, la recherche respecte (=1) ou non (=0) la casse.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

Ces exemples illustrent l'utilisation des fonctions `LeftPart`, `LeftPartFromRight`, `RightPart` et `RightPartFromLeft` sur une même chaîne de caractères: "Ceci_est_un_test" :

```
LeftPart("Ceci_est_un_test","_",0)
```

Renvoie la chaîne "Ceci".

```
LeftPartFromRight("Ceci_est_un_test","_",0)
```

Renvoie la chaîne "Ceci_est_un".

```
RightPart("Ceci_est_un_test","_",0)
```

Renvoie la chaîne "test".

```
RightPartFromLeft("Ceci_est_un_test","_",0)
```

Renvoie la chaîne "est_un_test".

RmDir()

Syntaxe BASIC interne

```
Function RmDir(strRmDirectory As String) As Long
```

Description

Détruit un répertoire.

Entrée

- *strRmDirectory* : Chemin complet du répertoire à détruire.

Sortie

- 0 : La fonction s'est exécutée normalement.
- Non nul : Code d'erreur.

Rnd()

Syntaxe BASIC interne

```
Function Rnd(dValue As Double) As Double
```

Description

Renvoie une valeur contenant un nombre aléatoire.

Entrée

- *dValue* : Paramètre optionnel dont la valeur définit le mode de génération adopté par la fonction :
  - Inférieur à zéro : Le même nombre est généré à chaque fois.
  - Supérieur à zéro : Nombre aléatoire suivant dans la série.
  - Egal à zéro : Dernier nombre aléatoire généré.
  - Omis : Nombre aléatoire suivant dans la série.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim MyNumber
  Randomize
  MyNumber= Int((10*Rnd)+1) :'Renvoie une valeur
aléatoire comprise entre 1 et 10.
  RetVal=MyNumber
```

Remarques

Note : Avant d'appeler cette fonction, vous devez utiliser la fonction Randomize, sans aucun paramètre, pour initialiser le générateur de nombres aléatoires.

RTrim()

Syntaxe BASIC interne

```
Function RTrim(strString As String) As String
```

Description

Supprime tous les espaces suivant le dernier caractère (qui n'est pas un espace) d'une chaîne.

Entrée

- *`strString`* : Chaîne de caractères à traiter.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
' This example uses the LTrim and RTrim functions
 to strip leading ' and trailing spaces,
respectively, from a string variable.
' It uses the Trim function alone to strip both
types of spaces.
' LCase and UCase are also shown in this example
as well as the use
' of nested function calls

Dim strString as String
Dim strTrimString as String
  strString = "  <-Trim->  " :' Initialize string.
  strTrimString = LTrim(strString) :' strTrimString
 = "<-Trim->  ".
  strTrimString = LCase(RTrim(strString)) :'
strTrimString = "  <-trim->".
  strTrimString = LTrim(RTrim(strString)) :'
strTrimString = "<-Trim->".
  ' Using the Trim function alone achieves the
same result.
  strTrimString = UCase(Trim(strString)) :'
strTrimString = "<-TRIM->".
  RetVal= "|" & strTrimString & "|"
```

Second()

Syntaxe BASIC interne

```
Function Second(tmTime As Date) As Long
```

Description

Renvoie le nombre de secondes contenu dans la l'heure exprimée par le paramètre *tmTime*.

Entrée

- *tmTime* : Paramètre au format Date+Heure à traiter.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim strSecond
  strSecond=Second(Date())
  RetVal=strSecond :'Renvoie le nombre de secondes
 écoulées dans l'heure courante par exemple "30"
s'il est actuellement 15:45:30
```

SetSubList()

Syntaxe BASIC interne

```
Function SetSubList(strValues As String, strRows
As String, strRowFormat As String) As String
```

Description

Définit les valeurs d'une sous-liste pour un contrôle "ListBox".

Entrée

- *strValues* : Chaîne source contenant les valeurs d'un contrôle "ListBox" à traiter.
- *strRows* : Liste de valeurs à ajouter ou à substituer à celles de la chaîne contenue dans le paramètre *strValues*. Les valeurs sont séparées par le caractère "|". Les lignes traitées sont identifiées par leur identifiant, situé à droite du signe "=". Les lignes inconnues de sont pas traitées.
- *strRowFormat* : Instructions de formatage de la sous-liste. Les instructions sont séparées par le caractère "|". Ce paramètre possède les caractéristiques suivantes :
  - "1" représente les informations contenues dans la première colonne de la sous-liste.
  - "i-j" peut être utilisé pour définir un ensemble de colonnes.
  - "-" prend en compte toutes les colonnes.
  - Une colonne inconnue ne renvoie aucune valeur.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim MyStr

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0",
 "A2|A1=a0, B2|B1=b0", "2|1") :'Renvoie
"A1|A2|a3=a0,B1|B2|b3=b0,c1|c2|c3=c0"

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0",
 "Z2=*,B2=b0", "2") :'Renvoie
"a1|Z2|a3=a0,b1|B2|b3=b0,c1|Z2|c3=c0"
```

```
MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0",
 "B5|B6|B7=b0,C5|C6,C7=c0", "5-7") :'Renvoie
"a1|a2|a3=a0,b1|b2|b3||B5|B6|B7=b0,c1|c2|c3||C5|C6|C7=c0"

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0",
 "B1|B2|B3|B4=b0", "-") :'Renvoie
"a1|a2|a3=a0,B1|B2|B3|B4=b0,c1|c2|c3=c0"
  MyStr=SubList("A|B|C,D|E|F", "X=*", "2")
:'Renvoie "A|X|C,D|X|F"
  RetVal=""
```

Sgn()

Syntaxe BASIC interne

```
Function Sgn(dValue As Double) As Double
```

Description

Renvoie une valeur indiquant le signe d'un nombre.

Entrée

- *dValue* : Nombre dont vous souhaitez connaître le signe.

Sortie

La fonction peut renvoyer une des valeurs suivante :

- 1 : Le nombre est supérieur à zéro.
- 0 : Le nombre est égal à zéro.
- -1 : Le nombre est inférieur à zéro.

Exemple

```
Dim dNumber as Double
  dNumber=-256
  RetVal=Sgn(dNumber)
```

Shell()

Syntaxe BASIC interne

```
Function Shell(strExec As String) As Long
```

Description

Lance un programme exécutable.

Entrée

- *strExec* : Chemin complet de l'exécutable à lancer.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim MyId
  MyId=Shell("C:\WinNT\notepad.exe")
  RetVal=""
```

Sin()

Syntaxe BASIC interne

```
Function Sin(dValue As Double) As Double
```

Description

Renvoie le sinus d'un nombre, exprimé en radians.

Entrée

- *dValue* : Nombre dont vous souhaitez connaître le sinus.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim dCalc as Double
  dCalc=Sin(150)
  RetVal=dCalc
```

Space()

Syntaxe BASIC interne

```
Function Space(iSpace As Long) As String
```

Description

Crée une chaînes de caractères comprenant le nombre d'espaces indiqué par le paramètre *iSpace*.

Entrée

- *iSpace* : Nombre d'espaces à insérer dans la chaîne.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim MyString
' Renvoie une chaîne de 10 espaces.
  MyString = Space(10)
 :' Insère 10 espaces entre deux chaînes.
  MyString = "Espace" & Space(10) & "inséré"
  RetVal=MyString
```

Remarques

Note : Cette fonction peut servir à formater des chaînes ou à effacer des données dans des chaînes de longueur fixe.

Sqr()

Syntaxe BASIC interne

```
Function Sqr(dValue As Double) As Double
```

Description

Renvoie la racine carrée d'un nombre.

Entrée

- *dValue* : Nombre dont vous souhaitez connaître la racine carrée.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim dCalc as Double
  dCalc=Sqr(81)
  RetVal=dCalc
```

Str()

Syntaxe BASIC interne

```
Function Str(strValue As String) As String
```

Description

Convertit un nombre en une chaîne de caractères.

Entrée

- *strValue* : nombre à convertir en chaîne.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim dNumber as Double
  dNumber=Cos(150)
  RetVal=Str(dCalc)
```

StrComp()

Syntaxe BASIC interne

```
Function StrComp(strString1 As String, strString2
As String, iOptionCompare As Long) As Long
```

Description

Effectue la comparaison entre deux chaînes de caractères.

Entrée

- *strString1* : Première chaîne de caractères.
- *strString2* : Deuxième chaîne de caractères.
- *iOptionCompare* : type de comparaison. Ce paramètre peut prendre la valeur "0" pour une comparaison binaire, ou "1" pour une comparaison du texte des deux chaînes.

Sortie

- -1 : *strString1* est supérieure à *strString2*.
- 0 : *strString1* est égale à *strString2*.
- 1 : *strString1* est inférieure à *strString2*.

String()

Syntaxe BASIC interne

```
Function String(iCount As Long, strString As
String) As String
```

Description

Renvoie une chaîne composée de *iCount* fois le caractère *strString*.

Entrée

- *iCount* : Nombre d'occurrences du caractère *strString*.
- *strString* : caractère utilisé pour la composition de la chaîne.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim iCount as Integer
Dim strTest as String
  strTest="T"
  iCount=5
  RetVal=String(iCount,strTest)
```

SubList()

Syntaxe BASIC interne

```
Function SubList(strValues As String, strRows As
String, strRowFormat As String) As String
```

Description

Renvoie une sous-liste d'une liste de valeurs contenue dans une chaîne de caractères représentant les valeurs d'un contrôle "ListBox".

Entrée

- *strValues* : Chaîne source contenant les valeurs d'un contrôle "ListBox" à traiter.
- *strRows* : Identifiants des lignes à inclure dans la sous-liste. Les identifiants sont séparés par une virgule. Certains jokers sont acceptés :
  - "*" inclut tous les identifiants dans la sous-liste.
  - Un identifiant inconnu renvoie une valeur vide pour la sous-liste.
- *strRowFormat* : Instructions de formatage de la sous-liste. Les instructions sont séparées par le caractère "|". Ce paramètre possède les caractéristiques suivantes :
  - "1" représente les informations contenues dans la première colonne de la liste dont on extrait une sous-liste.
  - "0" représente l'identifiant de la ligne de la liste dont on extrait une sous-liste.
  - "*" représente les informations contenues dans toutes les colonnes (à l'exception de l'identifiant de la ligne).
  - Une colonne inconnue ne renvoie aucune valeur.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim MyStr

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0",
 "a0,b0,a0", "3|2|3") :'Renvoie
"a3|a2|a3,b3|b2|b3,a3|a2|a3"

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0",
 "*", "*|0") :'Renvoie
"a1|a2|a3|a0,b1|b2|b3|b0,c1|c2|c3|c0"

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0",
 "*", "*=0") :'Renvoie
"a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0"

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0",
 "*", "999=0") :'Renvoie "=a0,=b0,=c0"

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0",
 "z0", "*=0") :'Renvoie ""

MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0",
 "*", "=1") :'Renvoie "=a1,=b1,=c1"
  MyStr=SubList("A|B|C,D|E|F", "*", "2=0")
:'Renvoie "B,E"
  RetVal=""
```

Tan()

Syntaxe BASIC interne

```
Function Tan(dValue As Double) As Double
```

Description

Renvoie la tangente d'un nombre, exprimé en radians.

Entrée

- *dValue* : Nombre dont vous souhaitez connaître la tangente.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim dCalc as Double
  dCalc=Tan(150)
  RetVal=dCalc
```

Time()

Syntaxe BASIC interne

```
Function Time() As Date
```

Description

Renvoie l'heure courante.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Timer()

Syntaxe BASIC interne

```
Function Timer() As Double
```

Description

Renvoie le nombre de secondes écoulées depuis 12:00 AM.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

TimeSerial()

Syntaxe BASIC interne

```
Function TimeSerial(iHour As Long, iMinute As Long,
iSecond As Long) As Date
```

Description

Cette fonction renvoie une heure formatée en fonction des paramètres *`iHour`*, *`iMinute`* et *`iSecond`*.

Entrée

- *`iHour`* : Heure.

- *iMinute* : Minutes.
- *iSecond* : Secondes.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

Chacun de ces paramètres peut prendre pour valeur une expression numérique représentant un nombre d'heures, de minutes ou de secondes. Ainsi l'exemple suivant :

```
TimeSerial(12-8, -10, 0)
```

renvoie la valeur :

```
3:50:00
```

Lorsque la valeur d'un paramètre est en dehors de l'intervalle de valeurs généralement admis (c'est à dire 0-59 pour les minutes et les secondes et 0-23 pour les heures), elle est convertie vers le paramètre immédiatement supérieur. Ainsi, si vous entrez "75" comme valeur pour le paramètre *iMinute*, ce dernier sera interprété comme 1 heure et 15 minutes.

L'exemple suivant :

```
TimeSerial (16, 50, 45)
```

renvoie la valeur :

```
16:50:45
```

TimeValue()

Syntaxe BASIC interne

```
Function TimeValue(tmTime As Date) As Date
```

Description

Cette fonction renvoie la partie heure d'une valeur "Date+Heure"

Entrée

- *tmTime* : Date au format "Date+Heure".

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

L'exemple suivant :

```
TimeValue ("1999/09/24 15:00:00")
```

renvoie la valeur :

```
15:00:00
```

ToSmart()

Syntaxe BASIC interne

```
Function ToSmart(strString As String) As String
```

Description

Cette fonction reformate un chaîne source en mettant des majuscules au début de chaque mot.

Entrée

- *strString* : Chaîne source à reformater.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Trim()

Syntaxe BASIC interne

```
Function Trim(strString As String) As String
```

Description

Supprime tous les espaces précédant le premier caractère (qui n'est pas un espace) d'une chaîne et tous les espaces suivant le dernier caractère (qui n'est pas un espace) d'une chaîne.

Entrée

- *strString* : Chaîne de caractères à traiter.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
' This example uses the LTrim and RTrim functions
 to strip leading ' and trailing spaces,
respectively, from a string variable.
' It uses the Trim function alone to strip both
types of spaces.
' LCase and UCase are also shown in this example
as well as the use
' of nested function calls

Dim strString as String
Dim strTrimString as String
  strString = "  <-Trim->  " :' Initialize string.
  strTrimString = LTrim(strString) :' strTrimString
 = "<-Trim->  ".
  strTrimString = LCase(RTrim(strString)) :'
strTrimString = "  <-trim->".
  strTrimString = LTrim(RTrim(strString)) :'
strTrimString = "<-Trim->".
  ' Using the Trim function alone achieves the
same result.
  strTrimString = UCase(Trim(strString)) :'
strTrimString = "<-TRIM->".
  RetVal= "|" & strTrimString & "|"
```

UCase()

Syntaxe BASIC interne

```
Function UCase(strString As String) As String
```

Description

Passe tous les caractères d'une chaîne en majuscules.

Entrée

- *strString* : Chaîne de caractères à passer en majuscules.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
' This example uses the LTrim and RTrim functions
 to strip leading ' and trailing spaces,
respectively, from a string variable.
' It uses the Trim function alone to strip both
types of spaces.
' LCase and UCase are also shown in this example
as well as the use
' of nested function calls

Dim strString as String
Dim strTrimString as String
  strString = "  <-Trim->  " :' Initialize string.
  strTrimString = LTrim(strString) :' strTrimString
 = "<-Trim->  ".
  strTrimString = LCase(RTrim(strString)) :'
strTrimString = "  <-trim->".
  strTrimString = LTrim(RTrim(strString)) :'
strTrimString = "<-Trim->".
  ' Using the Trim function alone achieves the
same result.
  strTrimString = UCase(Trim(strString)) :'
strTrimString = "<-TRIM->".
  RetVal= "|" & strTrimString & "|"
```

UnEscapeSeparators()

Syntaxe BASIC interne

```
Function UnEscapeSeparators(strSource As String,
strEscChar As String) As String
```

Description

Supprime tous les caractères d'échappement d'une chaîne de caractères.

Entrée

- *strSource* : Chaîne de caractères à traiter.
- *strEscChar* : Caractère d'échappement à supprimer.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim MyStr
  MyStr=UnEscapeSeparators("toi\|moi\|toi\|", "\")
 :'Renvoie la valeur "toi|moi|toi|"
  RetVal=""
```

Union()

Syntaxe BASIC interne

```
Function Union(strListOne As String, strListTwo As
String, strSeparator As String, strEscChar As
String) As String
```

Description

Rassemble deux chaînes de caractères délimitées par des séparateurs. Les doublons sont supprimés.

Entrée

- *strListOne* : Première chaîne de caractères.
- *strListTwo* : Deuxième chaîne de caractères.
- *strSeparator* : Séparateur utilisé pour délimiter les éléments contenus dans les chaînes.
- *strEscChar* : Caractère d'échappement. Si ce caractère préfixe le séparateur, ce dernier est ignoré.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim MyStr
  MyStr=Union("a1|a2,b1|b2", "a1|a3,b1|b2", ",",
"\") :'Renvoie la valeur "a1|a2,b1|b2,a1|a3"
  MyStr=Union("a1|a2,b1|b2", "a1|a3\,b1|b2", ",",
 "\") :'Renvoie la valeur
```

```
"a1|a2,b1|b2,a1|a3\,b1|b2"
  RetVal=""
```

UTCToLocalDate()

Syntaxe BASIC interne

```
Function UTCToLocalDate(tmUTC As Date) As Date
```

Description

Cette fonction convertit une date au format UTC (indépendante d'un quelconque fuseau horaire) en une date au format "Date+Heure".

Entrée

- *tmUTC* : Date au format UTC.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Val()

Syntaxe BASIC interne

```
Function Val(strString As String) As Double
```

Description

Convertit une chaîne de caractères représentant un nombre en un nombre de type "Double".

Entrée

- *strString* : Chaîne à convertir.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

Exemple

```
Dim strYear
Dim dYear as Double
  strYear=Year(Date())
 dYear=Val(strYear)
  RetVal=dYear :'Renvoie l'année en cours
```

WeekDay()

Syntaxe BASIC interne

```
Function WeekDay(tmDate As Date) As Long
```

Description

Renvoie le jour de la semaine contenu dans la date exprimée par le paramètre *tmDate*.

Entrée

- *tmDate* : Paramètre au format Date+Heure à traiter.

Sortie

Le nombre retourné correspond à un jour de la semaine, le "1" représentant le dimanche, le "2" le lundi, ..., le "7" le samedi.

Exemple

```
Dim strWeekDay
  strWeekDay=WeekDay(Date())
  RetVal=strWeekDay :'Renvoie le jour de la semaine
```

Year()

Syntaxe BASIC interne

```
Function Year(tmDate As Date) As Long
```

Description

Renvoie l'année contenue dans la date exprimée par le paramètre *tmDate*.

Entrée

- *tmDate* : Paramètre au format Date+Heure à traiter.

Sortie

En cas d'erreur, un message est inscrit dans le fichier journal de Connect-It.

III. Index

6 Fonctions disponibles - Domaine : Tous

CHAPITRE

- Abs
- AppendOperand
- ApplyNewVals
- Asc
- Atn
- BasicToLocalDate
- BasicToLocalTime
- BasicToLocalTimeStamp
- Beep
- CDbl
- ChDir
- ChDrive
- Chr
- CInt
- CLng
- Cos

- CountOccurences
- CountValues
- CSng
- CStr
- CurDir
- CVar
- Date
- DateAdd
- DateAddLogical
- DateDiff
- DateSerial
- DateValue
- Day
- EscapeSeparators
- ExeDir
- Exp
- ExtractValue
- FileCopy
- FileDateTime
- FileExists
- FileLen
- Fix
- FormatResString
- FV
- GetListItem
- Hex
- Hour
- InStr
- Int
- IPMT
- IsNumeric

- Kill
- LCase
- Left
- LeftPart
- LeftPartFromRight
- Len
- LocalToBasicDate
- LocalToBasicTime
- LocalToBasicTimeStamp
- LocalToUTCDate
- Log
- LTrim
- MakeInvertBool
- Mid
- Minute
- MkDir
- Month
- Name
- Now
- NPER
- Oct
- ParseDate
- ParseDMYDate
- ParseMDYDate
- ParseYMDDate
- PifFirstInCol
- PifGetBlobSize
- PifGetElementChildName
- PifGetElementCount
- PifGetInstance
- PifGetItemCount

- PifIgnoreDocumentMapping
- PifIgnoreNodeMapping
- PifIsInMap
- PifLogInfoMsg
- PifLogWarningMsg
- PifMapValue
- PifMapValueContaining
- PifNewQueryFromFmtName
- PifNewQueryFromXml
- PifNodeExists
- PifQueryClose
- PifQueryGetDateVal
- PifQueryGetDoubleVal
- PifQueryGetIntVal
- PifQueryGetLongVal
- PifQueryGetStringVal
- PifQueryNext
- PifRejectDocumentMapping
- PifRejectNodeMapping
- PifSetDateVal
- PifSetDoubleVal
- PifSetLongVal
- PifSetStringVal
- PifStrVal
- PifUserFmtStrToVar
- PifUserFmtVarToStr
- PMT
- PPMT
- PV
- Randomize
- RATE

- RemoveRows
- Replace
- Right
- RightPart
- RightPartFromLeft
- RmDir
- Rnd
- RTrim
- Second
- SetSubList
- Sgn
- Shell
- Sin
- Space
- Sqr
- Str
- StrComp
- String
- SubList
- Tan
- Time
- Timer
- TimeSerial
- TimeValue
- ToSmart
- Trim
- UCase
- UnEscapeSeparators
- Union
- UTCToLocalDate
- Val

- `WeekDay`
- `Year`

7 Fonctions disponibles - Domaine : Pif

CHAPITRE

- DateAdd
- DateAddLogical
- DateDiff
- PifFirstInCol
- PifGetBlobSize
- PifGetElementChildName
- PifGetElementCount
- PifGetInstance
- PifGetItemCount
- PifIgnoreDocumentMapping
- PifIgnoreNodeMapping
- PifIsInMap
- PifLogInfoMsg
- PifLogWarningMsg
- PifMapValue
- PifMapValueContaining

- PifNewQueryFromFmtName
- PifNewQueryFromXml
- PifNodeExists
- PifQueryClose
- PifQueryGetDateVal
- PifQueryGetDoubleVal
- PifQueryGetIntVal
- PifQueryGetLongVal
- PifQueryGetStringVal
- PifQueryNext
- PifRejectDocumentMapping
- PifRejectNodeMapping
- PifSetDateVal
- PifSetDoubleVal
- PifSetLongVal
- PifSetStringVal
- PifStrVal
- PifUserFmtStrToVar
- PifUserFmtVarToStr

8 Fonctions disponibles - Domaine : Builtin

CHAPITRE

- Abs
- AppendOperand
- ApplyNewVals
- Asc
- Atn
- BasicToLocalDate
- BasicToLocalTime
- BasicToLocalTimeStamp
- Beep
- CDbl
- ChDir
- ChDrive
- Chr
- CInt
- CLng
- Cos

- CountOccurences
- CountValues
- CSng
- CStr
- CurDir
- CVar
- Date
- DateSerial
- DateValue
- Day
- EscapeSeparators
- ExeDir
- Exp
- ExtractValue
- FileCopy
- FileDateTime
- FileExists
- FileLen
- Fix
- FormatResString
- FV
- GetListItem
- Hex
- Hour
- InStr
- Int
- IPMT
- IsNumeric
- Kill
- LCase
- Left

- LeftPart
- LeftPartFromRight
- Len
- LocalToBasicDate
- LocalToBasicTime
- LocalToBasicTimeStamp
- LocalToUTCDate
- Log
- LTrim
- MakeInvertBool
- Mid
- Minute
- MkDir
- Month
- Name
- Now
- NPER
- Oct
- ParseDate
- ParseDMYDate
- ParseMDYDate
- ParseYMDDate
- PMT
- PPMT
- PV
- Randomize
- RATE
- RemoveRows
- Replace
- Right
- RightPart

- RightPartFromLeft
- RmDir
- Rnd
- RTrim
- Second
- SetSubList
- Sgn
- Shell
- Sin
- Space
- Sqr
- Str
- StrComp
- String
- SubList
- Tan
- Time
- Timer
- TimeSerial
- TimeValue
- ToSmart
- Trim
- UCase
- UnEscapeSeparators
- Union
- UTCToLocalDate
- Val
- WeekDay
- Year