Peregrine

Connect-It
Référence de programmation

PART NUMBER CIT-3.2.1-FRE-01002-00292

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

PARTIE I Introduction

1 Bases essentielles de programmation

CHAPITRE

Ce chapitre présente les composants fondamentaux du langage Basic disponible sous Connect-It. Si vous possédez des bases de programmation et que vous avez déjà pratiqué d'autres langages, la très grande majorité de ce chapitre vous sera familière. Nous vous invitons néanmoins à parcourir rapidement les différentes sections proposées, certaines fonctionnalités classiques étant volontairement limitées ou absentes du Basic de Connect-It.

Introduction aux variables

Les variables sont utilisées pour stocker des données au cours de l'exécution d'un programme. Elles sont identifiées par :

- leur nom, utilisé pour référencer la valeur contenue par la variable.
- leur type, qui détermine quelles données peuvent être stockées dans la variable.

On distingue en général deux types de variables :

- Les tableaux,
- Les variables scalaires qui regroupent toutes les variables qui ne sont pas des tableaux.

Déclarer une variable

Toute variable doit être explicitement déclarée avant d'être utilisée. La syntaxe de déclaration est la suivante :

```
Dim <Nom de la variable> [As <Type de la variable>]
```

Note :

La déclaration explicite des variables dans le Basic Connect-It correspond à l'utilisation du mot clé **Option Explicit** de Microsoft Visual Basic.

Le nom d'une variable doit respecter les contraintes suivantes :

- Il doit commencer par une lettre majuscule ou minuscule,
- Il ne doit pas comporter plus de 40 caractères,
- Il peut contenir les lettres de A à Z et de a à z, les chiffres de 0 à 9, ainsi que le caractère underscore ("_").

  Note :

  Les caractères accentués sont autorisés mais leur utilisation est fortement déconseillée.

- Il ne peut être un mot clé réservé. Par exemple, tous les noms de fonctions Basic ou les clauses sont des mots clés réservés.

La clause optionnelle **As** permet de définir le type de la variable déclarée. Le type précise le type d'information stocké dans la variable. Les types disponibles sont par exemple : **String**, **Integer**, **Variant**, ...

Si la clause **As** est omise, la variable est considérée comme étant de type **Variant**.

Déclaration simple

Dans le cas d'une déclaration simple, chaque ligne déclarative concerne une seule variable, comme dans l'exemple suivant :

```
Dim I As Integer
Dim strName As String
Dim dNumber As Double
```

Déclaration combinée

Dans le cas d'une déclaration combinée, chaque ligne déclarative peut concerner un nombre quelconque de variables, comme dans l'exemple suivant :

```
Dim I As Integer, strName As String, dNumber As Double
Dim A, B, C As Integer
```

Note :

Comme il a été décrit précédemment, toute variable dont le type n'est pas précisé est considérée comme étant de type **Variant**. Ainsi, dans la deuxième ligne de l'exemple précédent, les variables **A** et **B** sont de type **Variant** et la variable **C** de type **Integer**.

Types des données

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.
Single	Nombre à virgule flottante de 4 octets (simple précision).
Double	Nombre à virgule flottante de 8 octets (double précision).
String	Texte pour lequel tous les caractères sont acceptés.
Date	Date ou Date+Heure.
Variant	Type générique pouvant représenter n'importe quel type.

Les types numériques

Le Basic disponible sous Connect-It propose plusieurs types de données numériques : Integer, Long, Single et Double. L'utilisation d'un type de données numérique occupe généralement moins de place en mémoire qu'un **Variant**.

Si vous avez la certitude qu'une variable stockera systématiquement des nombres entiers (par exemple 123) et non des nombres fractionnels (comme

3.14), il est préférable de la déclarer comme un **Integer** ou un **Long**. Les opérations effectuées sur ces types sont plus rapides et moins coûteuses en mémoire que pour les autres types de données. Ces types de donnée sont particulièrement adaptés pour les compteurs utilisés dans les boucles.

Si une variable doit contenir un nombre fractionnel, déclarez-la comme **Single** ou **Double**.

Note :

Les nombres à virgule flottante (**Single** ou **Double**) peuvent être sujets à des erreurs d'arrondis.

Le type String

Si vous avez la certitude qu'une variable stockera systématiquement une chaîne de caractères, déclarez-la comme **String** :

```
Dim MyString As String
```

Vous pouvez alors stocker des chaînes de caractères dans cette variable et manipuler son contenu en utilisant les fonctions dédiées au traitement des chaînes de caractères :

```
MyString = "This is a string"
MyString = Right(MyString,6)
```

Par défaut, une variable de type **String** possède une taille variable. L'espace réservé au stockage des chaînes de caractères augmente ou diminue en fonction de la taille des données affectées à la variable. Il est toutefois possible de déclarer une variable de type **String** dont la taille est fixe, en utilisant la syntaxe suivante :

```
Dim <Nom de la variable> As String * <Taille de la chaîne stockée>
```

L'exemple suivant déclare une variable qui contiendra 20 caractères :

```
Dim MyString As String * 20
```

Si vous stockez dans cette variable une chaîne de moins de 20 caractères, des espaces seront rajoutés en fin de chaîne jusqu'à concurrence de la taille prévue. A l'inverse, si vous stockez une chaîne de plus de 20 caractères, la chaîne sera tronquée à partir du vingtième et unième caractère.

Le type Variant

Le **Variant** est un type générique qui peut se substituer à tous les autres types. Vous n'avez pas à vous soucier d'éventuels problèmes de conversion entre les différents types de données qu'un **Variant** peut représenter. La conversion est réalisée automatiquement, comme dans l'exemple suivant :

```
Dim MyVariant As Variant
MyVariant = "123"
MyVariant = MyVariant - 23
MyVariant = "Top " & MyVariant
```

Bien que la conversion entre les différents types soit automatique, veillez à respecter les règles suivantes :

- Si vous réalisez des opérations arithmétiques sur un **Variant**, celui-ci doit impérativement contenir un nombre, même si celui-ci est représenté par une chaîne de caractères.
- Si une opération de concaténation de chaînes fait intervenir un **Variant**, utilisez l'opérateur **&** de préférence à l'opérateur **+**.

Un **Variant** peut également contenir deux valeurs particulières : la valeur vide et la valeur **Null**.

La valeur vide

Avant qu'une valeur soit affectée pour la première fois à une variable de type **Variant**, celle-ci contient la valeur vide. Cette valeur est une valeur particulière, différente de 0, d'une chaîne vide ou encore de la valeur **Null**. Pour tester si un **Variant** contient la valeur vide, utilisez la fonction Basic **IsEmpty()**, comme l'illustre l'exemple suivant :

```
Dim MyFirstVariant As Variant
Dim MySecondVariant As Variant
If IsEmpty(MyFirstVariant) Then MyFirstVariant = 0
MySecondVariant = 0
If IsEmpty(MySecondVariant) Then MySecondVariant = 123
```

Un **Variant** contenant la valeur vide peut être utilisé dans les expressions. Suivant les cas, il sera traîté soit comme de valeur 0, soit comme contenant une chaîne vide. Pour réaffecter la valeur vide à un **Variant**, utilisez le mot clé **Empty**, comme dans l'exemple suivant :

```
Dim MyVariant As Variant
MyVariant = 123
MyVariant = Empty
```

La valeur Null

La valeur **Null** est communément utilisée dans les bases de données pour indiquer une valeur inconnue ou manquante. Cette valeur possède des caractéristiques particulières :

- Les expressions faisant intervenir la valeur **Null** renvoient toujours la valeur **Null**. On parle de propagation de la valeur **Null** dans les expressions. Si une partie de l'expression vaut **Null** alors l'intégralité de l'expression vaut **Null**.
- En règle générale, si un paramètre de fonction a la valeur **Null**, la fonction renvoie la valeur **Null**.

Tableaux de données

Un tableau permet de stocker et de référencer un ensemble de variables par un nom unique et d'utiliser un nombre (un index) pour les identifier de façon unique. Tous les éléments d'un tableau partagent obligatoirement le même type de données. Vous ne pouvez pas créer un tableau contenant à la fois des variables de type **String** et **Double**. Evidemment, cette limitation peut être levée en utilisant des variables de type **Variant**.

Déclaration d'un tableau

Un tableau est un ensemble de variables.

Par convention, les notions suivantes sont présentées comme suit :

- Limite inférieure du tableau : index du premier élément.

  Note :

  Par défaut la limite inférieure d'un tableau est 0.

- Limite supérieure du tableau : index du dernier élément.

  Note :

  La limite supérieure d'un tableau ne peut excéder la taille d'un **Long** (2 147 483 646 éléments).

La déclaration d'un tableau est similaire à celle d'une variable :

```
Dim <Nom du tableau>(<Limite supérieure du tableau>) [As <Type des variab
les contenues dans le tableau>]
```

Exemples :

```
Dim MyFirstArray(30) As String ' 31 elements
Dim MySecondArray(9) As Double ' 10 elements
```

Vous pouvez également préciser la limite inférieure du tableau en utilisant la déclaration suivante :

```
Dim <Nom du tableau>(<Limite inférieure du tableau> To <Limite supérieure
 du tableau>) [As <Type des variables contenues dans le tableau>]
```

Exemples :

```
Dim MyFirstArray(1 To 30) As String ' 30 elements
Dim MySecondArray(5 To 9) As Double ' 5 elements
```

Limitations

Les limitations suivantes s'appliquent à la gestion des tableaux dans le Basic Connect-It :

- Les tableaux de taille variable ne sont pas supportés. En particulier, il est impossible de redimensionner un tableau à la voléee.
- Les tableaux à plusieurs dimensions ne sont pas supportés.

Structures de contrôle

Comme leur nom l'indique, les structures de contrôle permettent de contrôler l'exécution d'un programme. Elles sont de deux types :

- Les structures de décision : redirigent et orientent l'exécution d'un programme en fonction de la réalisation de certains critères,
- Les structures de boucles : permettent de répéter l'exécution d'un ensemble d'instructions en fonction de certains critères.

Structures de décision

Une structure de décision réalise l'exécution conditionnelle d'instructions en fonction du résultat d'un test. Les structures de décision disponibles sont les suivantes :

- **If...Then**
- **If...Then...Else...End If**
- **Select Case**

If...Then

Utilisez cette structure pour exécuter conditionnellement une ou plusieurs instructions. La syntaxe de cette structure peut être mono ou multiligne, la syntaxe monoligne ne permettant l'exécution que d'une seule instruction :

```
If <Condition> Then <Instruction>
```

```
If <Condition> Then
  <Instructions>
End If
```

La condition est généralement une comparaison, mais toute expression dont le résultat est une valeur numérique peut être utilisée. Cette valeur est alors interprétée comme étant **True** (Vraie) ou **False** (Fausse) par le Basic. **False** correspond à la valeur numérique 0, toute autre valeur étant considérée comme **True**.

Si la condition est évaluée comme **True**, la ou les instructions suivant le mot clé **Then** seront exécutées.

If...Then...Else...End If

Utilisez cette structure pour définir plusieurs blocs d'instructions conditionnelles. Un seul au plus de ces blocs est exécuté (le premier évalué comme **True**).

```
If <Condition1> Then
  <Instructions1>
ElseIf <Condition2> Then
  <Instructions2>
...
Else
  <InstructionsN>
End If
```

La première condition est testée, si le résultat est évalué comme **False**, la deuxième condition est testée et ainsi de suite jusqu'à ce que l'une d'entre elles soit évaluée comme **True**. Le jeu d'instructions situé après le mot clé **Then** de cette condition est alors exécuté.

Le mot clé **Else** est optionnel. Il permet de définir un jeu d'instructions à exécuter si toutes les conditions sont évaluées comme **False**.

Note :

Vous pouvez imbriquer autant de **ElseIf** que vous le souhaitez dans la structure de décision. Néanmoins, si vous comparez systématiquement la même expression à une valeur différente, la syntaxe de la structure de décision peut devenir inutilement complexe et difficile à lire. Nous vous conseillons, dans ce cas, d'utiliser de préférence un structure de décision de type **Select...Case**.

Select...Case

La fonctionnalité de cette structure est identique à celle des structures de décision précédentes, mais le code produit est en général plus lisible. Une structure **Select...Case** effectue un test unique en début de structure et compare le résultat du test aux valeurs de chaque mot clé **Case** dans la structure. S'il y a correspondance, le jeu d'instructions associé au mot clé **Case** est exécuté.

```
Select Case <Test>
[Case <Liste de valeurs 1>
<Instructions1>]
[Case <Liste de valeurs 2>
<Instructions2>]
...
[Case Else
<Instructionsn>]
End Select
```

Chaque liste de valeurs contient une ou plusieurs valeurs, séparées par des virgules. Si plusieurs mots clés **Case** déclarent une ou plusieurs valeurs correspondant au résultat du test, seul le jeu d'instructions associé au premier mot clé **Case** correspondant est exécuté.

Le jeu d'instructions associé au mot clé **Case Else** est exécuté si aucune correspondance n'a été détectée pour les mots clés **Case**.

Structures de boucle

Une structure de boucle permet de répéter l'exécution d'une série d'instructions. Les structures de boucle disponibles sont les suivantes :

- **Do...Loop**
- **For...Next**

Do...Loop

Utilisez cette structure pour exécuter une série d'instructions un nombre de fois non défini. La sortie de la boucle est effectuée lorsqu'une condition est réalisée ou non. Cette condition est une valeur ou une expression qui est évaluée comme **False** (0) ou **True** (différent de 0).

Note :

La sortie de la boucle peut être forcée en utilisant le mot clé **Exit Do** dans les instructions exécutées.

Il existe plusieurs variations de cette structure, mais la plus usitée est la suivante :

```
Do While <Condition>
  <Instructions>
Loop
```

Dans ce cas, la condition est évaluée en premier. Si elle est vérifiée (**True**), les instructions sont exécutées et le programme retourne au mot clé **Do While**, teste à nouveau la condition et ainsi de suite. La sortie de boucle est réalisée si la condition est évaluée comme **False**.

L'exemple suivant teste la valeur d'un compteur, incrémenté à chaque passage dans la boucle (ou itération). La sortie de la boucle est réalisée si le compteur contient la valeur 20.

```
Dim iCounter As Integer
iCounter = 0
Do While iCounter < 20
  iCounter = iCounter +1
Loop
```

L'exemple suivant reprend l'exemple précédent mais force la sortie de la boucle par un **Exit Do** si le compteur contient la valeur 10.

```
Dim iCounter As Integer
iCounter = 0
Do While iCounter < 20
  iCounter = iCounter +1
  If iCounter = 10 Then Exit Do
Loop
```

Dans ce type de structure **Do...Loop**, la condition est évaluée avant d'exécuter les instructions. Si vous souhaitez exécuter les instructions puis tester la condition, utilisez la structure **Do...Loop** suivante :

```
Do
  <Instructions>
Loop While <Condition>
```

Note :

Ce type de structure garantit au moins une exécution des instructions.

Les deux types de structure **Do...Loop** précédents itèrent tant que la condition est réalisée (**True**). Si vous souhaitez itérer tant que la condition n'est pas réalisée (**False**), utilisez l'une des deux structures suivantes :

```
Do Until <Condition>
  <Instructions>
Loop

Do
  <Instructions>
Loop Until <Condition>
```

En utilisant ce type de structure, l'exemple précédent peut s'écrire :

```
Dim iCounter As Integer
iCounter = 0
Do Until iCounter = 20
  iCounter = iCounter +1
Loop
```

For...Next

Utilisez cette structure pour exécuter une série d'instructions un nombre de fois déterminé. A l'inverse des structures **Do...Loop**, une boucle **For...Next** utilise une variable appelée compteur dont la valeur augmente ou diminue à chaque itération.

Note :

La sortie de la boucle peut être forcée en utilisant le mot clé **Exit For** dans les instructions exécutées.

```
For <Compteur> = <Valeur initiale> To <Valeur finale> [Step <Incrément>]
  <Instructions>
Next [<Compteur>]
```

IMPORTANT :

Les arguments **Compteur**, **Valeur initiale**, **Valeur finale** et **Incrément** sont tous représentés par des valeurs numériques.

Note :

Incrément peut être une valeur positive ou négative. Si elle est positive la **Valeur initiale** doit être inférieure ou égale à la **Valeur finale** pour que les instructions soient exécutées. Si elle est négative, la **Valeur initiale** doit être supérieure ou égale à la **Valeur finale** pour que les instructions soient exécutées. Si l'**Incrément** n'est pas précisé, sa valeur par défaut est 1.

Lors de l'exécution d'une boucle **For...Next**, les opérations suivantes sont réalisées :

1. Le compteur est initialisé et stocke la valeur initiale,
2. Le Basic teste si la valeur du compteur est supérieure à la valeur finale. Si tel est le cas, le programme sort de la boucle.

   Note :

   Si l'incrément est négatif, le Basic teste si la valeur du compteur est inférieure à la valeur finale.

3. Les instructions sont exécutées,
4. Le compteur est incrémenté de 1 ou de la valeur spécifiée par l'incrément,
5. Les opérations 2 à 4 sont répétées.

L'exemple suivant effectue la somme des nombres pairs jusqu'à 1000 :

```
Dim iCounter As Integer, lSum As Long
For iCounter = 0 To 1000 Step 2
  lSum = lSum + iCounter
Next
```

L'exemple suivant reprend l'exemple précédent mais force la sortie de la boucle par un **Exit For** si le compteur contient la valeur 500.

```
Dim iCounter As Integer, lSum As Long
For iCounter = 0 To 1000 Step 2
  lSum = lSum + iCounter
  If iCounter = 500 Then Exit For
Next
```

Opérateurs

Les opérateurs sont des symboles qui permettent de réaliser des opérations simples (addition, multiplication, ...) entre des variables et de les évaluer ou les comparer. On distingue plusieurs types d'opérateurs :

- Les opérateurs d'affectation,
- Les opérateurs de calcul,
- Les opérateurs relationnels (également appelés opérateurs de comparaison),
- Les opérateurs logiques.

Les opérateurs d'affectation

Ce type d'opérateur permet d'affecter une valeur à une variable. Le Basic Connect-It utilise une seule variable d'affectation, le signe "=". La syntaxe d'affectation est la suivante :

```
<Variable> = <Valeur>
```

Les opérateurs de calcul

Les opérateurs de calcul permettent de modifier mathématiquement la valeur d'une variable, ou de réaliser des opérations mathématiques simples entre deux expressions.

L'opérateur +

Cet opérateur permet d'effectuer la somme de deux valeurs. La syntaxe est la suivante :

```
<Résultat> = <Expression 1> + <Expression 2>
```

 Note :

Cet opérateur est utilisé tant pour effectuer la somme de deux nombres que pour concaténer des chaînes. Pour lever toute ambiguité, nous vous conseillons de réserver l'utilisation de cet opérateur à des sommes, et d'utiliser l'opérateur **&** pour concaténer des chaînes.

L'opérateur -

Cet opérateur permet d'effectuer la différence entre deux valeurs ou de signer négativement (opérateur monadique) une valeur. L'opérateur possède donc deux syntaxes :

```
<Résultat> = <Expression 1> - <Expression 2>
```

ou

```
- <Expression>
```

L'opérateur *

Cet opérateur permet d'effectuer la multiplication de deux valeurs. La syntaxe est la suivante :

```
<Résultat> = <Expression 1> * <Expression 2>
```

L'opérateur /

Cet opérateur permet d'effectuer la division de deux valeurs. La syntaxe est la suivante :

```
<Résultat> = <Expression 1> / <Expression 2>
```

L'opérateur ^

Cet opérateur permet d'élever une valeur à la puissance d'un exposant. La syntaxe est la suivante :

```
<Résultat> = <Expression 1> ^ <Expression 2>
```

Note :

Dans cette syntaxe l'expression 1 ne peut être négative que si l'expression 2 (l'exposant) est une valeur entière. Lorsqu'une expression effecute plusieurs opérations d'exposants en série, le Basic les interprète logiquement, de gauche à droite.

L'opérateur Mod

Cet opérateur permet de calculer le reste de la division euclidienne de deux valeurs. La syntaxe est la suivante :

```
<Résultat> = <Expression 1> Mod <Expression 2>
```

Note :

Les nombres à virgule flottante sont systématiquement arrondis à l'entier le plus proche.

L'exemple suivant renvoie la valeur 4 (6.8 est arrondi à l'entier le plus proche lors du calcul, soit 7):

```
Dim iValue As Integer
iValue = 25 Mod 6.8
```

Les opérateurs relationnels

Les opérateurs relationnels permettent de comparer des valeurs. Le tableau ci-dessous propose une vue d'ensemble des opérateurs relationnels :

Opérateur	Dénomination	Description	Syntaxe
=	Opérateur d'égalité	Compare deux valeurs et vérifie leur égalité	`<Expression 1> = < Expression 2>`
<	Opérateur d'infériorité stricte	Vérifie qu'une valeur est strictement inférieure à une autre	`<Expression 1> < < Expression 2>`
<=	Opérateur d'infériorité	Vérifie qu'une valeur est inférieure ou égale à une autre	`<Expression 1> <= <Expression 2>`
>	Opérateur de supériorité stricte	Vérifie qu'une valeur est strictement supérieure à une autre	`<Expression 1> > < Expression 2>`
>=	Opérateur de supériorité	Vérifie qu'une valeur est supérieure ou égale à une autre	`<Expression 1> >= <Expression 2>`
<>	Opérateur de différence	Vérifie qu'une valeur est différente d'une autre	`<Expression 1> <> <Expression 2>`

Les opérateurs logiques

Les opérateurs logiques permettent de vérifier la véracité de plusieurs conditions.

L'opérateur And

Cet opérateur effectue un ET logique (les deux conditions doivent être vérifiées) entre deux expressions. La syntaxe est la suivante :

```
<Résultat> = <Expression 1> And <Expression 2>
```

Si chaque expression est évaluée comme vraie (**True**), le résultat est vrai (**True**). Si l'une des deux expressions est évaluée comme fausse (**False**), le résultat est évalué comme faux (**False**).

L'opérateur Or

Cet opérateur effectue un OU logique (une des deux conditions doit être vérifiée) entre deux expressions. La syntaxe est la suivante :

```
<Résultat> = <Expression 1> Or <Expression 2>
```

Si l'une ou les deux expressions sont évaluées comme vraies (**True**), le résultat est vrai (**True**).

L'opérateur Xor

Cet opérateur effectue un OU exclusif (une seule des deux conditions doit être vérifiée) entre deux expressions. La syntaxe est la suivante :

```
<Résultat> = <Expression 1> Xor <Expression 2>
```

Si une seule des deux expressions est évaluée comme vraie (**True**), le résultat est vrai (**True**).

L'opérateur Not

Cet opérateur effectue la négation logique d'une expression. La syntaxe est la suivante :

```
<Résultat> = Not <Expression 1>
```

Si l'expression est évaluée comme vraie (**True**), le résultat est faux (**False**). Si l'expression est évaluée comme fausse (**False**), le résultat est vrai (**True**).

Priorité des opérateurs

Lorsque plusieurs opérateurs sont associés, l'ordre de priorité suivant est respecté dans l'évaluation des expressions. La liste ordonnée suivante classe les opérateurs dans un ordre de priorité décroissant :

1 ()
2 ^
3 -, +
4 /, *
5 Mod
6 =, >, <, <=, >=
7 Not
8 And
9 Or
10 Xor

Gestion de fichiers

Le Basic Connect-It permet de manipuler des fichiers de façon simple. Les opérations les plus usuelles (lecture, écriture,...) sont disponibles en standard.

Rappel préliminaire sur les fichiers

Un fichier est la façon dont un programme voit un objet externe. Il s'agit d'une collection d'enregistrements logiques, éventuellement structurée, sur laquelle le programme peut exécuter un ensemble d'opérations élémentaires (lecture, écriture, ...). Un enregistrement logique représente l'ensemble minimum de données qui peut être manipulé par une seule opération élémentaire.

Le Basic Connect-It gère uniquement les fichiers dits séquentiels. Dans un fichier séquentiel, les opérations se résument essentiellement en la lecture de l'enregistrement suivant ou l'écriture d'un nouvel enregistrement en fin du fichier séquentiel. Il n'est pas possible de réaliser simultanément lecture et écriture des enregistrements.

En lecture, le fichier séquentiel est initialement positionné sur le premier enregistrement logique. Chaque opération de lecture transfère un enregistrement dans une zone interne (en général une variable) du programme

et positionne le fichier sur l'enregistrement suivant. Une opération permet de déterminer s'il reste des enregistrements à lire (clause **EOF** : End Of File).

En écriture, le fichier séquentiel peut être initialement vide ou positionné après le dernier enregistrement du fichier. Chaque opération d'écriture transfère des données stockées dans une zone interne (en général une variable) du programme, dans un enregistrement du fichier et positionne le fichier après cet enregistrement.

Note :

Une des caractéristiques essentielles d'un fichier séquentiel est que les enregistrements sont lus dans l'ordre où ils ont été écrits.

Ouvrir et fermer des fichiers

La clause Open

Il s'agit de la clause de base pour toute manipulation de fichier. Elle permet d'ouvrir un fichier, que ce soit pour le lire, le créer, ou y écrire. La syntaxe est la suivante :

```
Open <Chemin du fichier> For <Mode> [Access <Type d'accès>] As [#]<Numéro
 de fichier>
```

Les paramètres de cette clause sont détaillés dans le tableau suivant :

Paramètre	Description
<Chemin du fichier>	Chaîne de caractères spécifiant le fichier concerné par l'opération. Cette chaîne peut contenir le chemin complet du fichier.

Paramètre	Description
<Mode>	Précise le mode de traitement du fichier. Ce paramètre peut contenir l'une des valeurs suivantes : • **Input** : le fichier est ouvert en lecture. • **Output** : le fichier est ouvert en écriture. Si le fichier existe et possède du contenu, celui-ci est écrasé. • **Append** : le fichier est ouvert en écriture. Si le fichier existe et possède du contenu, le nouveau contenu est ajouté en fin de fichier. • **Binary** : le fichier est ouvert en lecture binaire.
<Type d'accès>	Précise les opérations réalisables sur un fichier ouvert. Si le fichier est ouvert par un autre processus et que le type d'accès précisé n'est pas autorisé, la commande d'ouverture de fichier échoue. Ce paramètre peut contenir l'une des valeurs suivantes : • **Read** : le fichier est ouvert en lecture seule • **Write** : le fichier est ouvert en écriture seule • **Read Write** : le fichier est ouvert en lecture-écriture. Ce type d'accès n'est disponible que pour des accès en mode **Binary** ou **Append**.
<Numéro de fichier>	Identifie le fichier par un numéro unique compris entre 1 et 511. La fonction **FreeFile()** permet de déterminer le prochain numéro de fichier disponible.

Note :

Gardez en mémoire les points suivants :

- Tout fichier doit être ouvert par la clause **Open** avant de lire ou d'écrire des informations dans ce fichier.
- En mode **Append**, **Binary** ou **Output**, si le fichier référencé n'existe pas, il est créé.
- En mode **Binary** ou **Input**, vous pouvez ouvrir un fichier en utilisant un numéro différent sans fermer le fichier au préalable. En mode **Append** ou **Output**, vous devez impérativement fermer un fichier avant de l'ouvrir à nouveau sous un numéro différent.

La clause Close

Cette clause permet de fermer un fichier préalablement ouvert au moyen de la clause **Open()**. La syntaxe est la suivante :

```
Close [<Liste de fichiers>]
```

L'argument optionnel **<Liste de fichiers>** peut contenir un ou plusieurs numéros de fichiers. La syntaxe de cet argument optionnel est la suivante :

```
[[#]<Numéro de fichier>][,[#]<Numéro de fichier>]...
```

Note :

Si vous omettez le paramètre de cette clause, tous les fichiers actifs ouverts par une clause **Open()** sont fermés.

Lire des données d'un fichier

Deux clauses sont disponibles pour la lecture des données d'un fichier. L'utilisation de l'une ou l'autre de ces clauses dépend du mode d'accès spécifié pour le fichier. Les deux clauses sont les suivantes :

- **Input**
- **Line Input**

La clause Input

Cette clause est utilisée pour lire un nombre déterminé de caractères à partir d'un fichier ouvert en mode **Binary** ou **Input**. La syntaxe de cette clause est la suivante :

```
Input (<Nombre de caractères à lire>,[#]<Numéro de fichier>)
```

La clause Line Input

Cette clause est utilisée pour lire une ligne de données d'un fichier séquentiel, et la stocker dans une variable de type **String** ou **Variant**. La syntaxe de cette clause est la suivante :

```
Line Input #<Numéro de fichier>, <Nom de la variable>
```

IMPORTANT :

La clause lit les caractères un par un jusqu'à ce qu'un retour chariot ou un ensemble retour chariot - fin de ligne soit rencontré.

Ecrire des données dans un fichier

Une seule clause, **Print**, permet l'écriture de données dans un fichier. La syntaxe de cette clause est la suivante :

```
Print #<Numéro de fichier>, [<Données>]
```

2 Champ d'application des fonctions

CHAPITRE

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

3 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 :

[]	Ces crochets encadrent un paramètre optionnel. Ne les tapez pas dans votre commande. 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 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 la définition d'un noeud ou d'un bloc de script multilignes pour une propriété.
|	La barre verticale sépare les paramètres possibles qui figurent dans les accolades.

Les mises en forme suivantes ont des significations particulières :

`Police fixe`	Commande DOS, paramètre de fonction ou formatage de données.
`Exemple`	Exemple de code ou de commande.
...	Portion de code ou de commande omise.
Nom d'objet	Les noms de champs, d'onglets, de menus, de fichiers sont en caractères gras.
Note :	Note importante.

Note

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).

---

4 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.

La description et le code de la dernière erreur peuvent être retrouvés respectivement au moyen des fonctions **Err.Description** et **Err.Number**.

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>)
```

Le tableau ci-dessous répertorie les codes d'erreur les plus fréquents :

Code d'erreur	Signification
12001	Erreur indéfinie
12002	Mauvais paramètre pour une fonction
12003	Handle invalide ou objet détruit
12004	Plus de données disponible. Cette erreur arrive classiquement lors de l'exécution de requêtes. Quand le résultat de la requête ne renvoie aucune donnée, cette erreur est déclenchée.
12006	Valeur non valide (type incorrect pour un paramètre, etc.)
12009	Fonction obsolète ou non implémentée

5 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.
Single	Nombre à virgule flottante de 4 octets (simple précision).
Double	Nombre à virgule flottante de 8 octets (double précision).
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"

PARTIE II

Référence alphabétique

CHAPITRE 6

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 :

```
strExprstrOperatorstrOperand
```

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) As Long

Description

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

Entrée

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

Sortie

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

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(lChr As Long) As String

Description

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

Entrée

- **lChr** : 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 v
aleur "2"
  MyStr=CountOccurences("toi|moi|toi,moi|toi", "toi", "|") :'Renvoie la v
aleur "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 (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.

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,mo
i" dans la chaîne source
  MyStr=ExtractValue("toi", ",", "\") :'Renvoie "toi" et laisse "" dans l
a 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

Function FileExists(strFileName As String) As Long

Description

Cette fonction teste l'existence d'un fichier. La fonction renvoie les valeurs suivantes :

- 0 : le fichier n'existe pas.
- 1 : le fichier existe.

Entrée

- **strFileName** : Ce paramètre contient le chemin complet du fichier dont vous souhaitez tester l'existence.

Sortie

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

Exemple

```
If FileExists("c:\tmp\myfile.log") Then
  strFileName = "c:\archive\"  +  FormatDate(Date, "dddd d mmm yyyy") + "
.log"
  FileCopy("c:\tmp\myfile.log", strFileName)
End if
```

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)
```

FormatDate()

Syntaxe Basic interne

Function FormatDate(tmFormat As Date, strFormat As String) As String

Description

Formate une date en fonction de l'expression contenue dans le paramètre **strFormat.**

Entrée

- **tmFormat :** Date à formater.
- **strFormat :** Expression contenant les instructions de formatage.

Sortie

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

Exemple

L'exemple de code suivant montre comment formater une date :

```
Dim MyDate
  MyDate="2000/03/14"
  RetVal=FormatDate(MyDate, "dddd d mmmm yyyy") :'Renvoie "Tuesday 14 Mar
ch 2000"
```

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".

FormatString()

Syntaxe Basic interne

Description

Formate une chaîne de caractères en fonction de l'expression contenue dans le paramètre **strFormat**.

Exemple

L'exemple de code suivant montre comment formater une chaîne de caractères :

```
Dim MyString
  MyString="2000/03/14"
  RetVal=FormatString(MyString, "dddd d mmmm yyyy") :'Renvoie "Tuesday 14
 March 2000"
```

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'échéance 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.

GetEnvVar()

Syntaxe Basic interne

Function GetEnvVar(strVar As String, bExpand As Long) As String

Description

Cette fonction renvoie la valeur d'une variable d'environnement. Une valeur vide est renvoyée si la variable d'environnement n'existe pas.

Entrée

- **strVar :** Ce paramètre contient le nom de la variable d'environnement.
- **bExpand :** Ce paramètre booléen est utile quand la variable d'environnement fait référence à une ou plusieurs autres variables d'environnement. Dans ce cas, quand ce paramètre a pour valeur 1 (valeur par défaut), chaque variable référencée est remplacée par sa valeur. Dans le cas contraire, elle ne l'est pas.

Sortie

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

Exemple

```
RetVal = getEnvVar("PROMPT")
```

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'échéance 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 c
ourante 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'échéance 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 terme.

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 :

- jj/mm/aa
- jj/mm/aaaa
- mm/jj/aa
- mm/jj/aaaa
- aaaa/mm/jj
- 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", "aaaa/mm/jj")
```

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.

PifDateToTimezone()

Syntaxe Basic interne

Function PifDateToTimezone(tmSrc As Date, strTmznSrc As String, bWithDayLightSavingSrc As Long, strTmznDst As String, bDayLightSavingDst As Long) As Date

Description

Cette fonction convertit une date (au sens date et heure) exprimée dans un fuseau horaire donné dans un autre fuseau horaire en prenant en compte, si besoin est, les paramètres d'heure d'hiver.

Un message d'erreur est renvoyé si le fuseau horaire spécifié n'existe pas.

Note :

Le fuseau horaire 'UTC' (Universal Time Coordinated) peut être utilisé pour spécifier une date GMT 0 sans heure d'hiver.

Entrée

- **tmSrc** : Ce paramètre contient la date à convertir.
- **strTmznSrc** : Ce paramètre contient le nom du fuseau horaire source pour la conversion.
- **bWithDayLightSavingSrc** : En fonction de la valeur de ce paramètre, la fonction tient (=1) ou non (=0) compte des paramètres de l'heure d'hiver pour le fuseau horaire source.
- **strTmznDst** : Ce paramètre contient le nom du fuseau horaire de destination pour la conversion.
- **bDayLightSavingDst** : En fonction de la valeur de ce paramètre, la fonction tient (=1) ou non (=0) compte des paramètres de l'heure d'hiver pour le fuseau horaire destination.

Sortie

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

Exemple

```
PifDateToTimezone("2002-08-29 12:00:00", "Romance Standard Time", 1, "UTC
", 0)
```

renvoie la valeur :

```
2002-08-29 10:00:00
```

```
PifDateToTimezone("2002-08-29 12:00:00", "Romance Standard Time", 0, "UTC
", 0)
```

renvoie la valeur :

```
2002-08-29 11:00:00
```

```
PifDateToTimezone("2002-12-25 12:00:00", "Romance Standard Time", 1, "UTC
", 0)
```

renvoie la valeur :

```
2002-12-25 11:00:00
```

Remarques

Liste et description des fuseaux horaires :

Nom	Description
Dateline Standard Time	(GMT-12:00) Eniwetok, Kwajalein
Samoa Standard Time	(GMT-11:00) Midway Island, Samoa
Hawaiian Standard Time	(GMT-10:00) Hawaii
Alaskan Standard Time	(GMT-09:00) Alaska
Pacific Standard Time	(GMT-08:00) Pacific Time (US & Canada); Tijuana
US Mountain Standard Time	(GMT-07:00) Arizona
Mountain Standard Time	(GMT-07:00) Mountain Time (US & Canada)
Central America Standard Time	(GMT-06:00) Central America
Central Standard Time	(GMT-06:00) Central Time (US & Canada)
Mexico Standard Time	(GMT-06:00) Mexico City
Canada Central Standard Time	(GMT-06:00) Saskatchewan
SA Pacific Standard Time	(GMT-05:00) Bogota, Lima, Quito
Eastern Standard Time	(GMT-05:00) Eastern Time (US & Canada)
US Eastern Standard Time	(GMT-05:00) Indiana (East)
Atlantic Standard Time	(GMT-04:00) Atlantic Time (Canada)
SA Western Standard Time	(GMT-04:00) Caracas, La Paz
Pacific SA Standard Time	(GMT-04:00) Santiago
Newfoundland Standard Time	(GMT-03:30) Newfoundland
E. South America Standard Time	(GMT-03:00) Brasilia
SA Eastern Standard Time	(GMT-03:00) Buenos Aires, Georgetown
Greenland Standard Time	(GMT-03:00) Greenland
Mid-Atlantic Standard Time	(GMT-02:00) Mid-Atlantic
Azores Standard Time	(GMT-01:00) Azores
Cape Verde Standard Time	(GMT-01:00) Cape Verde Is.
Greenwich Standard Time	(GMT) Casablanca, Monrovia
GMT Standard Time	(GMT) Greenwich Mean Time : Dublin, Edinburgh, Lisbon, London
UTC	(GMT) Universal Coordinated Time
W. Europe Standard Time	(GMT+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna
Central Europe Standard Time	(GMT+01:00) Belgrade, Bratislava, Budapest, Ljubljana, Prague
Romance Standard Time	(GMT+01:00) Brussels, Copenhagen, Madrid, Paris

Nom	Description
Central European Standard Time	(GMT+01:00) Sarajevo, Skopje, Sofija, Vilnius, Warsaw, Zagreb
W. Central Africa Standard Time	(GMT+01:00) West Central Africa
GTB Standard Time	(GMT+02:00) Athens, Istanbul, Minsk
E. Europe Standard Time	(GMT+02:00) Bucharest
Egypt Standard Time	(GMT+02:00) Cairo
South Africa Standard Time	(GMT+02:00) Harare, Pretoria
FLE Standard Time	(GMT+02:00) Helsinki, Riga, Tallinn
Jerusalem Standard Time	(GMT+02:00) Jerusalem
Arabic Standard Time	(GMT+03:00) Baghdad
Arab Standard Time	(GMT+03:00) Kuwait, Riyadh
Russian Standard Time	(GMT+03:00) Moscow, St. Petersburg, Volgograd
E. Africa Standard Time	(GMT+03:00) Nairobi
Iran Standard Time	(GMT+03:30) Tehran
Arabian Standard Time	(GMT+04:00) Abu Dhabi, Muscat
Caucasus Standard Time	(GMT+04:00) Baku, Tbilisi, Yerevan
Afghanistan Standard Time	(GMT+04:30) Kabul
Ekaterinburg Standard Time	(GMT+05:00) Ekaterinburg
West Asia Standard Time	(GMT+05:00) Islamabad, Karachi, Tashkent
India Standard Time	(GMT+05:30) Calcutta, Chennai, Mumbai, New Delhi
Nepal Standard Time	(GMT+05:45) Kathmandu
N. Central Asia Standard Time	(GMT+06:00) Almaty, Novosibirsk
Central Asia Standard Time	(GMT+06:00) Astana, Dhaka
Sri Lanka Standard Time	(GMT+06:00) Sri Jayawardenepura
Myanmar Standard Time	(GMT+06:30) Rangoon
SE Asia Standard Time	(GMT+07:00) Bangkok, Hanoi, Jakarta
North Asia Standard Time	(GMT+07:00) Krasnoyarsk
China Standard Time	(GMT+08:00) Beijing, Chongqing, Hong Kong, Urumqi
North Asia East Standard Time	(GMT+08:00) Irkutsk, Ulaan Bataar
Malay Peninsula Standard Time	(GMT+08:00) Kuala Lumpur, Singapore
W. Australia Standard Time	(GMT+08:00) Perth
Taipei Standard Time	(GMT+08:00) Taipei
Tokyo Standard Time	(GMT+09:00) Osaka, Sapporo, Tokyo
Korea Standard Time	(GMT+09:00) Seoul
Yakutsk Standard Time	(GMT+09:00) Yakutsk
Cen. Australia Standard Time	(GMT+09:30) Adelaide
AUS Central Standard Time	(GMT+09:30) Darwin
E. Australia Standard Time	(GMT+10:00) Brisbane

Nom	Description
AUS Eastern Standard Time	(GMT+10:00) Canberra, Melbourne, Sydney
West Pacific Standard Time	(GMT+10:00) Guam, Port Moresby
Tasmania Standard Time	(GMT+10:00) Hobart
Vladivostok Standard Time	(GMT+10:00) Vladivostok
Central Pacific Standard Time	(GMT+11:00) Magadan, Solomon Is., New Caledonia
New Zealand Standard Time	(GMT+12:00) Auckland, Wellington
Fiji Standard Time	(GMT+12:00) Fiji, Kamchatka, Marshall Is.
Tonga Standard Time	(GMT+13:00) Nuku'alofa

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 <> 0 Then
    iToTal = iToTal + 1
  End If
Loop Until iIndex = 0
```

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".

Exemple

```
Dim lNetworkCardCount As Long
lNetworkCardCount = PifGetItemCount("NetworkCards")
```

PifIgnoreCollectionMapping()

Syntaxe Basic interne

Description

Cette fonction permet d'ignorer le traitement d'une collection.

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

Sortie

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

PifIgnoreDocumentMapping()

Syntaxe Basic interne

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.

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

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.

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.
- - **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

- 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, bLogErrIfOutOfRange 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.

  Note :

  Ce paramètre peut prendre toute valeur supérieure ou égale à 0; 0 correspondant à la première colonne.

- **strDefault** : Valeur par défaut renvoyée si l'élément n'est pas trouvé.
- **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

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)
```

Remarques

Note :

Cette fonction tient compte des caractères joker suivants :

- ? : correspond à n'importe quel caractère.
- * : correspond à un nombre indéfini de caractères quelconques.

PifMapValueContaining()

Syntaxe Basic interne

Function PifMapValueContaining(strKey As String, strMaptable As String, iPos As Long, strDefault As String, bCaseSensitive As Long, bLogErrIfOutOfRange 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.

  Note :

  Ce paramètre peut prendre toute valeur supérieure ou égale à 0; 0 correspondant à la première colonne.

- **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"))
```

Remarques

Note :

Cette fonction tient compte des caractères joker suivants :

- ? : correspond à n'importe quel caractère.
- * : correspond à un nombre indéfini de caractères quelconques.

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

```
dim hQuery as long
dim iRc as long

hQuery = pifNewQueryFromFmtName("Asset Management", "amEmplDept", "Name l
ike 'A%'")

Dim strValue as string

while (iRc = 0)
  iRc = pifQueryNext(hQuery)
  if iRc = 0 then
    strValue = pifQueryGetStringVal(hQuery, "Name")
    piflogInfoMsg strValue
  end if
wend

iRc = pifQueryClose(hQuery)
```

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 **strLayer**.

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 PifNodeExists("Hardware.Peripherials.Printer") = 0 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.

Exemple

```
dim hQuery as long
dim iRc as long

hQuery = pifNewQueryFromFmtName("Asset Management", "amEmplDept", "Name l
ike 'A%'")

Dim strValue as string
```

```
while (iRc = 0)
  iRc = pifQueryNext(hQuery)
  if iRc = 0 then
    strValue = pifQueryGetStringVal(hQuery, "Name")
    piflogInfoMsg strValue
  end if
wend

iRc = pifQueryClose(hQuery)
```

PifQueryGetDateVal()

Syntaxe Basic interne

Function PifQueryGetDateVal(lQueryHandle As Long, strPath As String, bPathMustExists As Long) 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, bPathMustExists As Long) 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, bPathMustExists As Long) 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, bPathMustExists As Long) 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, bPathMustExists As Long) 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

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.

Exemple

```
dim hQuery as long
dim iRc as long

hQuery = pifNewQueryFromFmtName("Asset Management", "amEmplDept", "Name l
ike 'A%'")

Dim strValue as string

while (iRc = 0)
  iRc = pifQueryNext(hQuery)
  if iRc = 0 then
    strValue = pifQueryGetStringVal(hQuery, "Name")
    piflogInfoMsg strValue
  end if
wend

iRc = pifQueryClose(hQuery)
```

Remarques

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

PifRejectCollectionMapping()

Syntaxe Basic interne

Description

Cette fonction permet de rejeter tous les éléments d'une collection, **uniquement dans un mapping collection à collection.**

Pour mémoire, la fonction **PifRejectNodeMapping()** permet simplement de rejeter l'élément courant d'une collection.

Un message, contenu dans le paramètre **strMsg** est envoyé dans le fichier journal.

Note :

L'utilisation de cette fonction équivaut à un rejet partiel du document. Les rejets partiels peuvent être récupérés dans les bilans de traitement de la boîte de mapping.

Exemple

Dans l'exemple suivant, tous les éléments de la collection sont rejetés si au moins l'un des éléments possède un noeud **quantity** dont la valeur vaut '-1' :

```
if [root.item(pifGetInstance).quantity] = -1 then
  pifRejectCollectionMapping("invalid quantity")
end if
```

A titre de comparaison avec la fonction **PifRejectNodeMapping()**, dans l'exemple suivant, seuls les éléments de la collection possédant un noeud **quantity** dont la valeur vaut '-1' sont rejetés :

```
if [root.item(pifGetInstance).quantity] = -1 then
  pifRejectNodeMapping
end if
```

PifRejectDocumentMapping()

Syntaxe Basic interne

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.

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

Description

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

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.

Exemple

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

PifSetDoubleVal()

Syntaxe Basic interne

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.

Exemple

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

PifSetLongVal()

Syntaxe Basic interne

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.

Exemple

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

PifSetNullVal()

Syntaxe Basic interne

Description

Cette fonction permet de renseigner un noeud du document cible avec la valeur 'NULL'. Si le noeud n'existe pas, la fonction procède à sa création.

Exemple

```
If [Name] = "" Then
  PifSetNullVal("FirstName")
End if
```

PifSetStringVal()

Syntaxe Basic interne

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.

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 = "yyyy'-'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 = "yyyy'-'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'échéance 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'échéance 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'échéance 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'échéance 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 "moimoimoimo
imoi"
  MyStr=Replace("toimoitoimoitoi", "Toi", "moi",1) :'Renvoie "toimoitoimo
itoi"
  MyStr=Replace("toimoiToimoitoi", "Toi", "moi",1) :'Renvoie "toimoimoimo
itoi"
  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
```

SetMaxInst()

Syntaxe Basic interne

Function SetMaxInst(lMaxInst As Long) As Long

Description

Cette fonction permet de fixer le nombre maximal d'instructions qu'un script Basic peut exécuter. Par défaut, le nombre d'instructions est limité à 10000.

Entrée

- **lMaxInst** : Ce paramètre contient le nombre d'instructions maximal exécutables par un script.

Sortie

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

Remarques

Note :

Si vous affectez la valeur "0" au paramètre **lMaxInst**, le nombre d'instructions exécutables par un script est illimité.

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 ne 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=b
0", "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\explorer.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") :'Renv
oie "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") :'Renv
oie "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") :'Re
nvoie "=a0,=b0,=c0"
  MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "z0", "*=0") :'Ren
voie ""
  MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "*", "=1") :'Renvo
ie "=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 "t
oi|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 valeu
r "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.

PARTIE III Index

Fonctions disponibles - Tous

INDEX

- 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
- PifDateToTimezone
- PifFirstInCol
- PifGetBlobSize
- PifGetElementChildName
- PifGetElementCount
- PifGetInstance
- PifGetItemCount
- PifIgnoreCollectionMapping
- PifIgnoreDocumentMapping
- PifIgnoreNodeMapping
- PifIsInMap
- PifLogInfoMsg
- PifLogWarningMsg
- PifMapValue
- PifMapValueContaining
- PifNewQueryFromFmtName
- PifNewQueryFromXml
- PifNodeExists
- PifQueryClose
- PifQueryGetDateVal
- PifQueryGetDoubleVal
- PifQueryGetIntVal
- PifQueryGetLongVal
- PifQueryGetStringVal
- PifQueryNext
- PifRejectCollectionMapping
- PifRejectDocumentMapping
- PifRejectNodeMapping
- PifSetDateVal
- PifSetDoubleVal
- PifSetLongVal
- PifSetNullVal
- PifSetStringVal
- PifStrVal
- PifUserFmtStrToVar
- PifUserFmtVarToStr
- PMT
- PPMT
- PV

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

Fonctions disponibles - Connect-It

INDEX

Fonctions disponibles - Builtin

INDEX

- 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

Fonctions disponibles -

INDEX


Peregrine
SYSTEMS