Peregrine

Connect-It 3.0.0 - プログラム用参考ガイド

Peregrine SYSTEMS

PART NUMBER CIT-3.0.0-JAP-01002-00188

本書の内容は、ライセンス契約に基づくプログラムのバージョン3.0.0に適用されます。

Connect-It

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

プログラム用
参考ガイド

目次

I. はじめに

1 関数の適用場所

本マニュアル内で説明されている全関数は、Connect-Itのスクリプトウィンドウ内で使用可能です。

2 表記法と形式

本章では、本マニュアル内で使用されている表記法と一部の定数のフォーマットについて説明します。

表記法

関数のシンタックスと用例の表記法は以下の表の通りです。

[ ]	ただし、BASICスクリプトでは、次の例のように、データへのパスを大括弧で囲みます。 `[Link.Link.Field]` 大括弧は、オプションのパラメータを示します。実際にコマンドのパラメータを入力するときは、大括弧は必要ありません。
<>	山形括弧は、パラメータの短い説明（英語の略語）を示します。実際にパラメータを入力する時は、山形括弧を使わずに、括弧内にあるテキストに該当する情報のみを入力してください。

{}	中括弧は、複数のパラメータの中から1つのパラメータを選択する場合に使います。実際にコマンドのパラメータを入力するときは、中括弧は必要ありません。
|	縦線（パイプ文字）は、中括弧に囲まれた複数のパラメータを区切る場合に使います。
*	山形括弧の右にあるアステリスクは、括弧で囲まれた数式が数回繰り返される可能性があることを示しています。

スクリプト内での「日付+時間」型定数のフォーマット

ユーザが定義している表示形式に関係なく、スクリプトで日付を記述するときは、次のような国際標準形式を使います。

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

例

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

注意: 年月日の区切りに、ハイフン（-）を使うこともできます。

Basic用の日付とUnix用の日付

BasicとUnixでは日付は異なった方法で表記されます。

- Basicでは、国際標準形式とDouble（倍精度）型浮動小数点数の両方を使用できます。後者の場合、整数部分は1899年12月30日の午前0時から数えて現在まで経過した日数を表し、小数部分は本日の午前0時から現在まで経過した時間の1日（86400秒）に対する割合（現在までの経過秒数を86400で割ったもの）を表します。
- UNIXでは、Long（倍長）型の整数で記述します。現地時間に関係なく、UTC（協定世界時）で1870年1月1日の午前0時から現在までに経過した秒数を示します。

Duration（時間）定数の形式

スクリプトでは、時間を秒単位で記述して保存します。例えば、Duration型フィールドのデフォルト値を3日に設定するには、次のスクリプトを使います。

```
RetVal=259200
```

同様に、時間を計算する関数も秒単位の時間を返します。

注意: Connect-Itの会計計算では、最も一般的な計算方法を使います。この方法では、1年は12ヶ月、1月は30日として計算するため、1年は360日になります。

3 定義

本章では、主要な用語の定義について説明します。

「関数」の定義

「関数」とは、なんらかの処理を実行し、値をユーザに返すプログラムです。
返される値は「戻り値」または「戻りコード」と呼ばれます。
Connect-It内部Basicで関数を呼び出すシンタックスの例は次の通りです。

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

「エラーコード」の定義

関数が正しく実行されなかった場合は、エラーコードが返ります。
次のシンタックスの`Err.Raise`関数を使うと、エラーメッセージを発生させることができます。

```
Err.Raise (<エラー番号>, <エラーメッセージ>)
```

4 関数とパラメータのデータ型

データ型のリスト

関数とパラメータで使用可能なデータ型とその説明は、以下の表の通りです。

データ型	説明
Integer（整数）	-32,768～+32,767の整数
Long（倍長整数）	-2,147,483,647～+2,147,483,646の整数
Double（倍精度）	8バイト浮動小数点数
String（文字列）	任意の文字からなるテキスト
Date（日付）	日付または日付+時刻
Variant（可変）	汎用の任意の型

関数の型

関数の型は、その関数の返す値の型に対応しています。プログラムのコンパイルエラーやランタイムエラーが発生するのを防ぐために、正しい型の関数を適切な場所で使う必要があります。

パラメータの型

関数で使うパラメータには型が決まっており、関数を正しく実行するためには、正しい型のパラメータを使用する必要があります。関数のシンタックス内では、パラメータの型を示す接頭辞がパラメータの先頭に付いています。使用される型と、型に関連付けられている接頭辞は、以下の表の通りです。

データ型	BASIC関数のシンタックスで使われている接頭辞
Integer（整数）	i
Long（倍長整数）	l
Double（倍精度）	d
String（文字列）	str
Date（日付）	dt
Variant（可変）	v

II. 関数の説明

5 関数の説明

Abs()

内部BASICシンタックス

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

説明

数値の絶対値を返します。

パラメータ

- *dValue*：絶対値を取得する数値

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

AppendOperand()

内部BASICシンタックス

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

説明

関数に渡されるパラメータに応じて文字列を連結します。文字列は次のように連結されます。

```
strExpr
strOperator
strOperand
```

パラメータ

- *strExpr*：連結される文字列式
- *strOperator*：連結する演算子
- *strOperand*：連結するオペランド

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

注

注意: strExprまたはstrOperandパラメータのいずれかを省略すると、strOperatorは使用されません。

ApplyNewVals()

内部BASICシンタックス

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

説明

［ListBox］コントロール内の同じセルに同じ値を割り当てます。

パラメータ

- *strValues*：処理する［ListBox］コントロールの値が含まれている文字列
- *strNewVals*：セルに割り当てる新しい値
- *strRows*：処理する行のID。各IDをコンマ（,）で区切ります。
- *strRowFormat*：サブリストの書式化命令。各命令をパイプ文字（|）で区切ります。個々の命令には、*strNewVals*パラメータを含んでいる列の番号を指定します。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

Asc()

内部BASICシンタックス

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

説明

文字列の最初の文字のASCIIコード（数値）を返します。

パラメータ

- *strAsc*：処理する文字列

例

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

Atn()

内部BASICシンタックス

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

説明

数値のアークタンジェントを返します。単位はラジアンです。

パラメータ

- *dValue*：アークタンジェントを取得する数値

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

BasicToLocalDate()

内部BASICシンタックス

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

説明

BASIC形式の日付を文字列の日付（Windowsのコントロールパネルに表示される形式）に変換します。

パラメータ

- *strDateBasic*：変換するBASIC形式の日付

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

BasicToLocalTime()

内部BASICシンタックス

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

説明

BASIC形式の時刻を文字列の時刻（Windowsのコントロールパネルに表示される形式）に変換します。

パラメータ

- *strTimeBasic*：変換するBASIC形式の時刻

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

BasicToLocalTimeStamp()

内部BASICシンタックス

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

説明

BASIC形式の日付+時刻を文字列の日付+時刻（Windowsのコントロールパネルに表示される形式）に変換します。

パラメータ

- *strTSBasic*：変換するBASIC形式の日付+時刻

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

Beep()

内部BASICシンタックス

```
Function Beep()
```

説明

コンピュータで警告音を鳴らします。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

CDbl()

内部BASICシンタックス

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

説明

式を倍精度型（Double型）に変換します。

パラメータ

- *dValue*：変換する式

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim dNumber As Double
Dim iInteger as Integer
```

```
iInteger = 25
dNumber=CDbl(iInteger)
RetVal=dNumber
```

ChDir()

内部BASICシンタックス

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

説明

現在のディレクトリを変更します。

パラメータ

- *strDirectory*：新しいディレクトリ

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

ChDrive()

内部BASICシンタックス

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

説明

現在のドライブを変更します。

パラメータ

- *strDrive*：新しいドライブ名

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

Chr()

内部BASICシンタックス

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

説明

*iChr*パラメータから渡されたASCIIコードに対応する文字列を返します。

パラメータ

- *iChr*：文字のASCIIコード

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
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()

内部BASICシンタックス

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

説明

指定した式を整数型に変換します。

パラメータ

- *iValue*：変換する式

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

CLng()

内部BASICシンタックス

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

説明

指定した式を倍長整数型に変換します。

パラメータ

- *lValue*：変換する式

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

Cos()

内部BASICシンタックス

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

説明

数値のコサインを返します。単位はラジアンです。

パラメータ

- *dValue*：コサインを取得する数値

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

CountOccurences()

内部BASICシンタックス

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

説明

１つの文字列内に特定の文字列が何個含まれているかをカウントします。

パラメータ

- *strSearched*：処理する文字列
- *strPattern*：*strSearched*パラメータ内で検索する文字列
- *strEscChar*：エスケープ文字。関数が*strSearched*文字列内でこの文字を検出すると、検索を中止します。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim MyStr
 MyStr=CountOccurences("you|me|you,me|you", "you", ",") :' "2"を返します。
 MyStr=CountOccurences("you|me|you,me|you", "you", "|") :' "1"を返します。
```

CountValues()

内部BASICシンタックス

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

説明

１つの文字列に含まれる要素数をカウントします。区切り文字やエスケープ文字を区別します。

パラメータ

- *strSearched*：**処理する文字列**
- *strSeparator*：**要素を区切る区切り文字**
- *strEscChar*：**エスケープ文字。この文字が区切り文字の前に付くと、その区切り文字は無視されます。**

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim MyStr
  MyStr=CountValues("you|me|you \|me|you", "|", "\") : ' 4を返します。
  MyStr=CountValues("you|me|you \|me|you", "|", "") : ' 5を返します。
```

CSng()

内部BASICシンタックス

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

説明

指定した式を浮動小数点数型（Float型）に変換します。

パラメータ

- *fValue*：変換する式

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

CStr()

内部BASICシンタックス

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

説明

指定した式を文字列型に変換します。

パラメータ

- *strValue*：変換する式

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

CurDir()

内部BASICシンタックス

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

説明

現在のパスを返します。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

CVar()

内部BASICシンタックス

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

説明

指定した式を可変型（Variant型）に変換します。

パラメータ

- *vValue*：変換する式

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

Date()

内部BASICシンタックス

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

説明

現在のシステムの日付を返します。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

DateAdd()

内部BASICシンタックス

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

説明

開始日に実際の経過時間を追加して、新たな日付を計算します。

パラメータ

- *tmStart*：経過時間を追加する日付
- *tsDuration*：*tmStart*の日付に追加する時間

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

DateAddLogical()

内部BASICシンタックス

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

説明

開始日に論理上の経過時間を追加して、新たな日付を計算します（1ヶ月を30日として計算します）。

パラメータ

- *tmStart*：経過時間を追加する日付
- *tsDuration*：*tmStart*の日付に追加する時間（秒単位）

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

DateDiff()

内部BASICシンタックス

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

説明

指定した２つの日付の間の時間を秒単位で計算します。

パラメータ

- *tmEnd*：計算する期間の終了日
- *tmStart*：計算する期間の開始日

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

1998年１月１日から1999年１月１日までの間に経過する時間を計算します。

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

DateSerial()

内部BASICシンタックス

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

説明

iYear、*iMonth*、および*iDay*パラメータで指定した形式で日付型の値を返します。

パラメータ

- *iYear*：西暦。0 - 99の間の場合は、1900年 - 1999年までの年を表します。その他の年に関しては、４桁（1800など）で指定する必要があります。
- *iMonth*：月
- *iDay*：日

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

各パラメータに、それぞれ日、月、または年を表す数式を利用できます。

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

例えば、上記の例は次の値を返します。

```
1989/1/7
```

パラメータの値が予想される範囲（日付ならば１-31、月ならば１-12など）以外の値の場合、関数は空の日付を返します。

DateValue()

内部BASICシンタックス

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

説明

「日付＋時刻」の値の日付の部分を返します。

パラメータ

- *tmDate*：「日付＋時刻」形式の日付

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

次の例は、

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

次の値を返します。

```
1999/09/24
```

Day()

内部BASICシンタックス

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

説明

*tmDate*パラメータの日付を返します。

パラメータ

- *tmDate*：処理する「時刻+日付」形式のパラメータ

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

EscapeSeparators()

内部BASICシンタックス

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

説明

区切り文字の前にエスケープ文字を付けます。

パラメータ

- *strSource*：処理する文字列
- *strSeparators*：エスケープ文字を付ける区切り文字。複数の区切り文字を宣言する場合は、*strEscChar*パラメータで指定するエスケープ文字で区切る必要があります。
- *strEscChar*：エスケープ文字。*strSeparators*パラメータのすべての区切り文字の前にこのエスケープ文字が付けられます。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim MyStr
 MyStr=EscapeSeparators("you|me|you,me|you", "¦\", " \") :' "you \me \|you \,me \|you"を返します。
```

ExeDir()

内部BASICシンタックス

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

説明

この関数は実行可能ファイルのフルパスを返します。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

Exp()

内部BASICシンタックス

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

説明

数値のべき乗を返します。

パラメータ

- *dValue*：べき乗を取得する数値。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

ExtractValue()

内部BASICシンタックス

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

説明

１つの文字列内で、区切り文字が前に付いていない値を取り出します。取り出した値は、その文字列から削除されます。この関数は、エスケープ文字も区別します。値を抽出する文字列に区切り文字が含まれていない場合は、その文字列全体が抽出され、元の文字列は完全に削除されます。

パラメータ

- *pstrData*：処理する文字列
- *strSeparator*：処理する文字列の区切り文字
- *strEscChar*：エスケープ文字。この文字が区切り文字の前に付くと、その区切り文字は無視されます。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim MyStr
 MyStr=ExtractValue("you,me", ",", "\") :' "you"を返し、"me"を文字列に残します。
 MyStr=ExtractValue(",you,me", ",", "\") :' ""を返し、"you,me"を文字列に残します。
 MyStr=ExtractValue("you", ",", "\") :' "you"を返し、""を文字列に残します。
 MyStr=ExtractValue("you\,me", ",", "\") :' "you\,me"を返し、""を文字列に残します。
 MyStr=ExtractValue("you\,me", ",", "") :' "you\"を返し、"me"を文字列に残します。
 RetVal=""
```

FileCopy()

内部BASICシンタックス

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

説明

ファイルまたはフォルダをコピーします。

パラメータ

- *strSource*：コピーするファイルまたはディレクトリのフルパス
- *strDest*：コピー先のファイルまたはディレクトリのフルパス

戻りコード

- 0：成功
- 0以外：エラーコード

FileDateTime()

内部BASICシンタックス

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

説明

ファイルの時刻と日付を倍長整数型で返します。

パラメータ

- *strFileName*：処理するファイルのフルパス

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

FileExists()

内部BASICシンタックス

説明

ファイルが存在するかどうかテストします。

FileLen()

内部BASICシンタックス

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

説明

ファイルのサイズを返します。

パラメータ

- *strFileName*：処理するファイルのフルパス

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

Fix()

内部BASICシンタックス

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

説明

数値の整数部分を返します（負数の場合は、その値の正の方向の一番近い整数を返します）。

パラメータ

- *dValue*：整数部分を取得する数値

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

FormatResString()

内部BASICシンタックス

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

説明

文字列内の変数$1、$2、$3、$4、および$5を、*strParamOne*、*strParamTwo*、*strParamThree*、*strParamFour*、および*strParamFive*パラメータに渡された文字列にそれぞれ置き換えます。

パラメータ

- *strResString*：処理する文字列
- *strParamOne*：変数$1を置き換える文字列
- *strParamTwo*：変数$2を置き換える文字列
- *strParamThree*：変数$3を置き換える文字列
- *strParamFour*：変数$4を置き換える文字列
- *strParamFive*：変数$5を置き換える文字列

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

次の例は、

```
FormatResString("I$1he$2you$3", "you", "we", "they")
```

"Iyouheweyouthey"を返します。

FV()

内部BASICシンタックス

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

説明

定額の定期的な支払、および固定利率に基づいて計算した、実際の年間支払金額を返します。

パラメータ

- *dblRate*：支払日ごとの利率。例えば、年利が6%のローンの毎月の支払日の利率は、次のようになります。

0.06/12=0.005 または 0.5%

- *iNper*：総支払回数
- *dblPmt*：1回の支払金額。支払金額には一般的に元金と利子が含まれます。
- *dblPV*：実際に支払わなければならない金額（総額）
- *iType*：支払期限を示します。次のいずれかの値になります。
  - *0*：支払が後払い（期間内の最後）の場合
  - *1*：支払が先払い（期間内の初め）の場合

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

注

注意: RateとNperパラメータの計算には同じ単位の支払額を使用する必要があります。支払う金額（Pmtパラメータの金額）はマイナスの数値、受け取る総額はプラスの数値で表わされます。

GetListItem()

内部BASICシンタックス

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

説明

区切り文字で区切られている文字列の*lNb*番目の文字列を返します。

パラメータ

- *strFrom*：処理する文字列
- *strSep*：処理する文字列の区切り文字
- *lNb*：取得する文字列の位置
- *strEscChar*：エスケープ文字。この文字が区切り文字の前に付くと、その区切り文字は無視されます。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

次に2つの例を示します。

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

"is"を返します。

```
GetListItem("this%_is_a_test", "_", 2, "%")
```

"a"を返します。

Hex()

内部BASICシンタックス

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

説明

10進法のパラメータの16進法の値を返します。

パラメータ

- *dValue*：16進法の値を取得する10進法の数値

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

Hour()

内部BASICシンタックス

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

説明

*tmTime*パラメータの時間の部分を返します。

パラメータ

- *tmTime*：処理する「時刻+日付」形式のパラメータ

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

InStr()

内部BASICシンタックス

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

説明

文字列内で、検索する文字列が最初に見つかった文字の位置を返します。

パラメータ

- *iPosition*：検索の開始点。このパラメータは必ず指定する必要があります。65,535よりも小さい有効な正の整数でなければなりません。
- *strSource*：処理する文字列
- *strPattern*：検索する文字列

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

Int()

内部BASICシンタックス

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

説明

数値の整数部分を返します（負数の場合は、その値の負の方向の一番近い整数を返します）。

パラメータ

- *dValue*：整数部分を取得する数値

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

IPMT()

内部BASICシンタックス

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

説明

年間支払金額のうち、指定した支払日の利子の金額を返します。

パラメータ

- *dblRate*：支払日ごとの利率。例えば、年利が6%のローンの毎月の支払日の利率は、次のようになります。

0.06/12=0.005 または 0.5%

- *iPer*：計算する期間。１から*Nper*の数値の間で指定します。
- *iNper*：総支払回数
- *dblPV*：実際に支払わなければならない金額（総額）
- *dblFV*：支払が終了した時の差引残高または将来の支払用に確保する金額。一般に、特にローンを返済する場合は、このパラメータを「0」に設定します。実際、すべての支払を済ませるとローンの値はゼロになります。
- *iType*：支払期限を示します。次のいずれかの値になります。
- *0*：支払が後払い（期間内の最後）の場合
- *1*：支払が先払い（期間内の初め）の場合

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

注

注意: RateとNperパラメータの計算には同じ単位の支払額を使用する必要があります。支払う金額（Pmtパラメータの金額）はマイナスの数値、受け取る総額はプラスの数値で表わされます。

IsNumeric()

内部BASICシンタックス

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

説明

文字列に数値が含まれるかどうかを識別します。

パラメータ

- *strString*：解析する文字列

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

Kill()

内部BASICシンタックス

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

説明

ファイルを削除します。

パラメータ

- *strKilledFile*：処理するファイルのフルパス

戻りコード

- 0：成功
- 0以外：エラーコード

LCase()

内部BASICシンタックス

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

説明

文字列パラメータに含まれるすべての文字を小文字に変換して返します。

パラメータ

- *strString*：小文字に変換する文字列

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
' この例では、LtrimとRtrim関数を使って文字列変数から文字列の前後のスペースを削除します。
' また、Trim関数だけで前後のスペースを取り除く例も示します。
' ネストした関数と一緒にLcaseおよびUcase関数も使っています。

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->".
 ' Trimだけを使っても同じ結果が得られます。
 strTrimString = UCase(Trim(strString)) :' strTrimString = "<-TRIM->".
 RetVal= "|" & strTrimString & "|"
```

Left()

内部BASICシンタックス

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

説明

文字列の左端から、iNumberで指定した数の文字を返します。

パラメータ

- *strString*：処理する文字列
- *iNumber*：返す文字数

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim lWord, strMsg, rWord, iPos :' Declare variables.
 strMsg = "Left() Test."
 iPos = InStr(1, strMsg, " ") : ' スペースを探す
 lWord = Left(strMsg, iPos - 1) :' 左側の単語を取得
 rWord = Right(strMsg, Len(strMsg) - iPos) :' 右側の単語を取得
 strMsg=rWord+lWord :' 2つの単語を交換
 RetVal=strMsg
```

LeftPart()

内部BASICシンタックス

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

説明

*strSep*パラメータに指定されている区切り文字の左側の文字列を1つ取得します。
左から右に向かって区切り文字を探します。
*bCaseSensitive*パラメータを使って、大文字と小文字を区別することもできます。

パラメータ

- *strFrom*：処理する文字列
- *strSep*：処理する文字列の区切り文字

- *bCaseSensitive*：このパラメータを使って、大文字と小文字を区別する（=1）、または区別しない（=0）かを指定します。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

文字列"This_is_a_test"に、LeftPart、LeftPartFromRight、RightPart、およびRightPartFromLeft関数を使った例を示します。

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

"This"を返します。

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

"This_is_a"を返します。

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

"test"を返します。

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

"is_a_test"を返します。

LeftPartFromRight()

内部BASICシンタックス

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

説明

*strSep*パラメータに指定されている区切り文字の左側にある文字列を１つ取得します。

右から左に向かって区切り文字を探します。

*bCaseSensitive*パラメータを使って、大文字と小文字を区別することもできます。

パラメータ

- *strFrom*：処理する文字列
- *strSep*：処理する文字列の区切り文字
- *bCaseSensitive*：このパラメータを使って、大文字と小文字を区別する（=1）、または区別しない（=0）かを指定します。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

文字列"This_is_a_test"に、LeftPart、LeftPartFromRight、RightPart、およびRightPartFromLeft関数を使った例を示します。

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

"This"を返します。

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

"This_is_a"を返します。

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

"test"を返します。

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

"is_a_test"を返します。

Len()

内部BASICシンタックス

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

説明

文字列または可変型データに含まれる文字数を返します。

パラメータ

- *vValue*：処理する可変型データ

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim strTest as String
Dim iLength as Integer
 strTest = "Peregrine Systems"
 iLength = Len(strTest) : ' iLengthの値は17
 RetVal=iLength
```

LocalToBasicDate()

内部BASICシンタックス

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

説明

文字列形式の日付（Windowsのコントロールパネルに表示される形式）をBASIC形式の日付に変換します。

パラメータ

- *strDateLocal*：変換する文字列形式の日付

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

LocalToBasicTime()

内部BASICシンタックス

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

説明

文字列形式の時刻（Windowsのコントロールパネルに表示される形式）をBASIC形式の時刻に変換します。

パラメータ

- *strTimeLocal*：変換する文字列形式の時刻

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

LocalToBasicTimeStamp()

内部BASICシンタックス

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

説明

文字列形式の日付+時刻（Windowsのコントロールパネルに表示される形式）をBASIC形式の日付+時刻に変換します。

パラメータ

- *strTSLocal*：変換する文字列形式の日付+時刻

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

LocalToUTCDate()

内部BASICシンタックス

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

説明

「日付＋時刻」フォーマットの日付をUTCフォーマット（タイムゾーンに関係しない） へ変換します。

パラメータ

- *tmLocal*：「日付＋時刻」形式の日付

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

Log()

内部BASICシンタックス

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

説明

数値の自然対数を返します。

パラメータ

- *dValue*：対数を取得する数値

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

LTrim()

内部BASICシンタックス

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

説明

文字列の先頭のスペースをすべて取り除きます。

パラメータ

- *strString*：処理する文字列

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
' この例では、LtrimとRtrim関数を使って文字列変数から文字列の前後のスペースを削除します。
' また、Trim関数だけで前後のスペースを取り除く例も示します。
' ネストした関数と一緒にLcaseおよびUcase関数も使っています。

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->".
' Trimだけを使っても同じ結果が得られます。
strTrimString = UCase(Trim(strString)) :' strTrimString = "<-TRIM->".
RetVal= "|" & strTrimString & "|"
```

MakeInvertBool()

内部BASICシンタックス

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

説明

ブール値の逆の値を返します（0は1、他の値はすべて0になります）。

パラメータ

- *lValue*：処理する数値

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim MyValue
  MyValue=MakeInvertBool(0) : ' 1を返します。
  MyValue=MakeInvertBool(1) : ' 0を返します。
  MyValue=MakeInvertBool(254) : ' 0を返します。
```

Mid()

内部BASICシンタックス

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

説明

文字列内の一部分の文字列を返します。

パラメータ

- *strString*：処理する文字列
- *iStart*：*strString*から取り出す文字列の開始位置
- *iLen*：取り出す文字列の長さ

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim strTest as String
 strTest="One Two Three" : ' テスト文字列を定義
 strTest=Mid(strTest,5,3) :' strTest="Two"
 RetVal=strTest
```

Minute()

内部BASICシンタックス

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

説明

*tmTime*パラメータに含まれている時刻の分の部分を返します。

パラメータ

- *tmTime*：処理する「時刻+日付」形式のパラメータ

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim strMinute
 strMinute=Minute(Date())
 RetVal=strMinute : '現在の時刻の分の部分を返します。例えば、時刻が15:45:30の場合は、"45"を返します。
```

MkDir()

内部BASICシンタックス

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

説明

新しいディレクトリを作成します。

パラメータ

- *strMkDirectory*：作成するディレクトリのフルパス

戻りコード

- 0：成功
- 0以外：エラーコード

Month()

内部BASICシンタックス

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

説明

*tmDate*パラメータに含まれている日付の月の部分を返します。

パラメータ

- *tmDate*：処理する「時刻+日付」形式のパラメータ

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim strMonth
 strMonth=Month(Date())
 RetVal=strMonth : ' 現在の月を返します。
```

Name()

内部BASICシンタックス

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

説明

ファイルの名前を変更します。

パラメータ

- *strSource*：名前を変更するファイルのフルパス
- *strDest*：新しいファイル名

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

Now()

内部BASICシンタックス

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

説明

現在の日付と時刻を返します。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

NPER()

内部BASICシンタックス

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

説明

定額の定期的な支払、および一定利率に基づく、年間の支払回数を返します。

パラメータ

- *dblRate*：支払日ごとの利率。例えば、年利が6%のローンの毎月の支払日の利率は、次のようになります。

0.06/12=0.005 または 0.5%

- *dblPmt*：1回の支払金額。支払金額には一般的に元金と利子が含まれます。
- *dblPV*：実際に支払わなければならない金額（総額）
- *dblFV*：支払が終了したときの差引残高または将来の支払用に確保する金額。一般に、特にローンを返済する場合は、このパラメータを「0」に設定します。実際、すべての支払を済ませるとローンの値はゼロになります。
- *iType*：支払期限を示します。次のいずれかの値になります。
  - *0*：支払が後払い（期間内の最後）の場合
  - *1*：支払が先払い（期間内の初め）の場合

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

注

注意: 支払う金額（Pmtパラメータの金額）はマイナスの数値、受け取る総額はプラスの数値で表わされます。

Oct()

内部BASICシンタックス

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

説明

10進法のパラメータの８進法の値を返します。

パラメータ

- *dValue*：８進法の値を取得する数値

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

ParseDate()

内部BASICシンタックス

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

説明

文字列フォーマットの日付を、Basic日付オブジェクトへ変換します。

パラメータ

- *strDate*：文字列フォーマットの日付
- *strFormat*：文字列に含まれている日付のフォーマット。以下の値が使用可能です。
  - DD/MM/YY
  - DD/MM/YYYY
  - MM/DD/YY
  - MM/DD/YYYY
  - YYYY/MM/DD
  - Date：クライアントコンピュータの日付パラメータに基づいた日付
  - DateInter ：国際標準形式の日付

- *strStep*：このオプションパラメータには、文字列内で使用される日付の区切り文字が含まれます。許可される区切り文字は「\」と「-」です。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

ParseDMYDate()

内部BASICシンタックス

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

説明

次の形式の日付から、日付オブジェクト（BASIC形式）を返します。

```
dd/mm/yyyy
```

パラメータ

- *strDate*：文字列として保存されている日付

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

ParseMDYDate()

内部BASICシンタックス

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

説明

次の形式の日付から、日付オブジェクト（BASIC形式）を返します。

mm/dd/yyyy

パラメータ

- *strDate*：文字列として保存されている日付

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

ParseYMDDate()

内部BASICシンタックス

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

説明

yyyy/mm/dd形式の日付の文字列を、BASICの日付型変数に変換します。

パラメータ

- *strDate*：文字列として保存されている日付

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

PifFirstInCol()

内部BASICシンタックス

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

説明

*strChildCond*に指定した検索条件に基づいて、コレクション内の最初の項目の番号を返します。

パラメータ

- *strPathCol*：検索するコレクションへのパス
- *strChildCond*：検索条件
-

注意: nがコレクション内の要素の数の場合は、iStartCountは、0からn-1の間の数値でなければなりません。

iStartCount：検索開始インデックス

戻りコード

- 0：成功
- 0以外：エラーコード

例

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

Do
```

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

PifGetBlobSize()

内部BASICシンタックス

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

説明

Blobオブジェクトのサイズを返します。

パラメータ

- *strPath*：コレクション内のBlobオブジェクトのフルパス

戻りコード

フルパスで識別されたBlobオブジェクトのサイズを返します。

例

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

PifGetElementChildName()

内部BASICシンタックス

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

説明

ソースドキュメントタイプの識別されたノードのN番目の子要素の名前を返します。

パラメータ

- *strPath*：操作に関連するノードのフルパス
- *iItem*：子要素の名前を取得する場合、その子要素の番号

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

以下の例はノード「MyDocument.MyNode」の3番目の子要素の名前を返します。

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

PifGetElementCount()

内部BASICシンタックス

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

説明

ソースドキュメントタイプの識別されたノードの数を返します。

パラメータ

- *strPath*：操作に関連するノードのフルパス

戻りコード

関数は、パスが*strPath*で指定されているノードの子要素の数を返します。

例

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

PifGetInstance()

内部BASICシンタックス

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

説明

コレクション内で現在処理している要素の番号を返します。最初に処理される要素の番号は「0」です。

戻りコード

処理される要素の番号を文字列として返します。

例

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

PifGetItemCount()

内部BASICシンタックス

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

説明

パスにより識別されたコレクション内の要素の数を返します。

パラメータ

- *strPath*：コレクションのフルパス

戻りコード

コレクションが存在しない場合は、0（ゼロ）を返します。

PifIgnoreDocumentMapping()

内部BASICシンタックス

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

説明

この関数を使って、処理するドキュメントを１つ飛ばすことができます。
*strMsg*パラメータの情報メッセージをログに送るように設定できます。

パラメータ

- *strMsg*：ログに送る文字列。情報をログに記録する時だけ指定します。

戻りコード

- 0：成功
- 0以外：エラーコード

例

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

PifIgnoreNodeMapping()

内部BASICシンタックス

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

説明

この関数を使って、ドキュメント内のノードを１つ飛ばすことができます。
*strMsg*パラメータの情報メッセージをログに送るように設定できます。>

パラメータ

- *strMsg*：ログに送る文字列。情報をログに記録する時だけ指定します。

戻りコード

- 0：成功
- 0以外：エラーコード

例

```
If [Comment] = "" Then
 PifIgnoreNodeMapping("現在のノードは処理されていません")
Else
 RetVal = [Comment]
End If
```

注

注意: ノードが処理されないため、ログに送られたメッセージは、処理を飛ばしたノードの親ノードに書き込まれます。

PifIsInMap()

内部BASICシンタックス

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

説明

キーワードがマップテーブルにあるかどうかをテストします。大文字／小文字を区別して検索できます。

パラメータ

- *strKey*：キーワード
- *strMaptable*：検索するマップテーブルの名前
-
  - *0*：大文字と小文字を区別しません。
  - *1*：大文字と小文字を区別します。

  bCaseSensitive：このパラメータを使って、検索で大文字と小文字を区別するかを指定できます。

戻りコード

- 0：キーワードはありません。
- 1：キーワードが見つかりました。

例

```
If PifIsInMap("CAT_PC", "MainAsset") Then
  RetVal = 1
Else
  RetVal = 0
End If
```

PifLogInfoMsg()

内部BASICシンタックス

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

説明

*strMsg*パラメータの情報メッセージをログに送ります。

パラメータ

- *strMsg*：ログに送るメッセージを含んでいる文字列

戻りコード

- 0：成功
- 0以外：エラーコード

例

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

PifLogWarningMsg()

内部BASICシンタックス

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

説明

*strMsg*パラメータの警告メッセージをログに送ります。

パラメータ

- *strMsg*：ログに送る警告メッセージを含んでいる文字列

戻りコード

- 0：成功
- 0以外：エラーコード

例

```
If [Brand] = "" Then
  PifLogWarningMsg("不明なメーカ")
Else
  RetVal = [Brand]
End if
```

PifMapValue()

内部BASICシンタックス

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

説明

マップテーブルの１つの要素の値を返します。*strKey*、*strMapTable*、および*iPos*を使って、処理する要素を特定します。

大文字／小文字を区別して検索できます。

パラメータ

- *strKey*：検索する要素を含んでいるマップテーブルの行を特定するためのキーワード
- *strMaptable*：検索するマップテーブルの名前
- *iPos*：取得する値の要素を含む列の番号
- *strDefault*：要素が見つからなかった場合に返されるデフォルトの値
-
  - *0*：大文字と小文字を区別しません。
  - *1*：大文字と小文字を区別します。

  bCaseSensitive：このパラメータを使って、検索で大文字と小文字を区別するかを指定できます。

戻りコード

目的の要素が見つからない場合に、見つかった文字列または*strDefault*パラメータに指定されているデフォルトの文字列を返します。

例

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

PifMapValueContaining()

内部BASICシンタックス

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

説明

マップテーブルの１つの要素の値を返します。*strKey*、*strMapTable*、および*iPos*を使って、処理する要素を特定します。

大文字／小文字を区別して検索できます。

PifMapValue関数と異なり、*strKey*パラメータにキーワードのスーパーセットを指定できます。

例えば、*strKey*に"Monitor"を指定すると、キーワード"Cat_Monitor"が付いた要素を検索できます。

パラメータ

- *strKey*：検索する要素を含むマップテーブルの行を特定するためのキーワードまたはキーワードの部分集合
- *strMaptable*：検索するマップテーブルの名前
- *iPos*：取得する値の要素を含む列の番号
- *strDefault*：要素が見つからなかった場合に返されるデフォルトの値
- *bCaseSensitive*：このパラメータを使って、検索で大文字と小文字を区別するかを指定できます。
  - *0*：大文字と小文字を区別しません。
  - *1*：大文字と小文字を区別します。

戻りコード

見つかった文字列、または*strDefault*パラメータに指定されているデフォルトの文字列を返します。

例

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

PifNewQueryFromFmtName()

内部BASICシンタックス

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

説明

リソースが生成するドキュメントのリスト内で事前に定義されたドキュメントタイプ上に、クエリを作成します。

パラメータ

- *strCntrName*：リソースの名前（クエリの実行先リソース）
- *strFmtName*：（生成用ドキュメントタイプとして事前に定義された）ドキュメントタイプの識別子
- *strLayer*：AQLクエリのWHERE句

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

PifNewQueryFromXml()

内部BASICシンタックス

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

説明

リソースにクエリを作成します。*strQuery*パラメータで、ドキュメントタイプ全体をXML形式で定義する必要があります。処理（AQLのクエリ句）は、*strQuery*パラメータによりXML形式で定義されます。

パラメータ

- *strCntrName*：リソースの名前（クエリの実行先リソース）
- *strQuery*：クエリの実行先のドキュメントタイプ（属性、構造体、コレクション）を定義するXMLドキュメント
- *strLayer*クエリの処理（WHERE句、ORDER BY句）を定義するXMLドキュメント

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

PifNodeExists()

内部BASICシンタックス

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

説明

フルパスでノードを特定し、そのノードが生成されるドキュメント内に存在するかどうかを確認します。

パラメータ

- *strPath*：処理するノードのフルパス

戻りコード

次のいずれかの値を返します。

- *0*：ノードは存在しません。
- *1*：ノードが存在します。

例

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

PifQueryClose()

内部BASICシンタックス

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

説明

クエリを閉じ、クエリが使用する内部リソースを解放します。

パラメータ

- *lQueryHandle*：PifNewQueryFromFmtName()関数またはPifNewQueryFromXml()関数で作成されたクエリ上のハンドル

戻りコード

- 0：成功
- 0以外：エラーコード

PifQueryGetDateVal()

内部BASICシンタックス

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

説明

現在のドキュメントのノードの値（Date形式）を返します。現在のドキュメントは、PifQueryNext()関数でセットされたクエリカーソルの、セット先のドキュメントです。

パラメータ

- *lQueryHandle*：PifNewQueryFromFmtName()関数とPifNewQueryFromXml()関数で作成されたクエリのハンドル
- *strPath*：値の取得元の現在のドキュメントノードのパス

戻りコード

識別されるノードの値

PifQueryGetDoubleVal()

内部BASICシンタックス

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

説明

現在のドキュメントのノードの値（Double形式）を返します。現在のドキュメントは、PifQueryNext()関数でセットされたクエリカーソルの、セット先のドキュメントです。

パラメータ

- *lQueryHandle*：PifNewQueryFromFmtName()関数またはPifNewQueryFromXml()関数で作成されたクエリのハンドル
- *strPath*：値の取得元の現在のドキュメントノードのパス

戻りコード

識別されるノードの値

PifQueryGetIntVal()

内部BASICシンタックス

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

説明

現在のドキュメントのノードの値（Integer整数形式）を返します。現在のドキュメントは、PifQueryNext()関数でセットされたクエリカーソルの、セット先のドキュメントです。

パラメータ

- *lQueryHandle*：**PifNewQueryFromFmtName()関数またはPifNewQueryFromXml()関数で作成されたクエリのハンドル**
- *strPath*：**値の取得元の現在のドキュメントノードのパス**

戻りコード

識別されるノードの値

PifQueryGetLongVal()

内部BASICシンタックス

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

説明

現在のドキュメントのノードの値（Long形式）を返します。現在のドキュメントは、PifQueryNext()関数でセットされたクエリカーソルの、セット先のドキュメントです。

パラメータ

- *lQueryHandle*：PifNewQueryFromFmtName()関数またはPifNewQueryFromXml()関数で作成されたクエリのハンドル
- *strPath*：値の取得元の現在のドキュメントノードのパス

戻りコード

識別されるノードの値

例

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

PifQueryGetStringVal()

内部BASICシンタックス

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

説明

現在のドキュメントのノードの値（文字列形式）を返します。現在のドキュメントは、PifQueryNext()関数でセットされたクエリカーソルの、セット先のドキュメントです。

パラメータ

- *lQueryHandle*：PifNewQueryFromFmtName()関数またはPifNewQueryFromXml()関数で作成されたクエリのハンドル
- *strPath*：値の取得元の現在のドキュメントノードのパス

戻りコード

識別されるノードの値

例

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

PifQueryNext()

内部BASICシンタックス

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

説明

クエリカーソルを次の結果へセットします。PifNewQueryFromFmtName()またはPifNewQueryFromXml()関数の呼出し後に、カーソルは最初のドキュメントにはセットされません。ユーザはPifQueryNext()関数を呼び出し、次の関数と共に現在のドキュメントの値へアクセスします：

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

パラメータ

- *lQueryHandle*：PifNewQueryFromFmtName()関数または PifNewQueryFromXml()関数で作成されたクエリ上のハンドル

戻りコード

- 0：成功
- 0以外：エラーコード

注

検索するデータがなくなると関数はエラーを返します（エラーコード-2003）。

PifRejectDocumentMapping()

内部BASICシンタックス

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

説明

ドキュメントを拒否します。拒否されたドキュメントは、次のコネクタに転送されません。*strMsg*パラメータの情報メッセージをログに送るように設定できます。

パラメータ

- *strMsg*：ログに送る文字列。情報をログに記録する時だけ指定します。

戻りコード

- 0：成功
- 0以外：エラーコード

例

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

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

PifRejectNodeMapping()

内部BASICシンタックス

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

説明

ノードを拒否します。*strMsg*パラメータの情報メッセージをログに送るように設定できます。

パラメータ

- *strMsg*：ログに送る文字列。情報をログに記録する時だけ指定します。

戻りコード

- 0：成功
- 0以外：エラーコード

例

```
If [Location.Name] = "" Then
  PifRejectNodeMapping(PifStrVal("UNKNOWN_LOCATION"))
End If
```

PifSetDateVal()

内部BASICシンタックス

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

説明

ターゲットドキュメント内でノードの値を固定化します。ノードがない場合は、関数がノードを作成します。

パラメータ

- *strPath*：操作に関連するノードのフルパス
- *dttVal*：ノードに割り当てる値（日付）

戻りコード

- 0：成功
- 0以外：エラーコード

例

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

PifSetDoubleVal()

内部BASICシンタックス

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

説明

ターゲットドキュメント内でノードの値を固定化します。ノードがない場合は、関数がノードを作成します。

パラメータ

- *strPath*：操作に関連するノードのフルパス
- *dVal*：ノードに割り当てる値（倍精度：Double）

戻りコード

- 0：成功
- 0以外：エラーコード

例

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

PifSetLongVal()

内部BASICシンタックス

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

説明

ターゲットドキュメント内でノードの値を固定化します。ノードがない場合は、関数がノードを作成します。

パラメータ

- *strPath*：操作に関連するノードのフルパス
- *lVal*：ノードに割り当てる値（倍長整数）

戻りコード

- 0：成功
- 0以外：エラーコード

例

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

PifSetStringVal()

内部BASICシンタックス

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

説明

ターゲットドキュメント内でノードの値を固定化します。ノードがない場合は、関数がノードを作成します。

パラメータ

- *strPath*：操作に関連するノードのフルパス
- *strVal*：ノードに割り当てる値（文字列）

戻りコード

- 0：成功
- 0以外：エラーコード

例

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

PifStrVal()

内部BASICシンタックス

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

説明

*strId*パラメータのIDに関連付けられている文字列を返します。

パラメータ

- *strID*：取得する文字列のID

戻りコード

IDが見つからない場合は、空の文字列を返し、Connect-Itのログファイルにエラーを書込みます。IDが見つかった場合は、そのIDに関連付けられている文字列を返します。

例

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

PifUserFmtStrToVar()

内部BASICシンタックス

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

説明

文字列を既製フォーマット（ユーザフォーマット）に基づいて、Connect-Itのインタフェース内のウィザードを使って処理します。既製フォーマットの属性に応じて日付型または数値型の数を返します。

パラメータ

- *strData*：処理する文字列
- *strUserFmtName*：既製フォーマット（ユーザフォーマット）名

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

以下の2つの既製フォーマットがあるとします。

- ソース = "yyy'-'mm'-'dd"
- ターゲット = "'dddd' 'dd' 'mmmm' 'yyyy'"

この場合のスクリプトは以下の通りです。

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

これは[date_modified]= 2001-05-30 用に"木曜日30 5月 2001" という値を返します。

注

注意: 既製フォーマットに関する詳細は、Connect-Itの『ユーザガイド』を参照してください。

PifUserFmtVarToStr()

内部BASICシンタックス

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

説明

既製フォーマット（ユーザフォーマット）に基づいて変数を処理し、文字列を返します。

パラメータ

- *vData*：関数に処理される変数
- *strUserFmtName*：既製フォーマット（ユーザフォーマット）名

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

以下の2つの既製フォーマットがあるとします。

- ソース = "yyy'-'mm'-'dd"
- ターゲット = "'dddd' 'dd' 'mmmm' 'yyyy'"

この場合のスクリプトは以下の通りです。

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

これは[date_modified]= 2001-05-30 用に "木曜日 30 5月 2001" という値を返します。

注

注意: 既製フォーマットに関する詳細は、Connect-Itの『ユーザガイド』を参照してください。

PMT()

内部BASICシンタックス

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

説明

定額の定期的な支払、および一定利率に基づいて計算した、年間支払金額を返します。

パラメータ

- *dblRate*：支払日ごとの利率。例えば、年利が6%のローンの毎月の支払日の利率は、次のようになります。

```
0.06/12=0.005 または 0.5%
```

- *iNper*：総支払回数
- *dblPV*：実際に支払わなければならない金額（総額）
- *dblFV*：支払が終了した時の差引残高または将来の支払用に確保する金額。一般に、特にローンを返済する場合は、このパラメータを「0」に設定します。実際、すべての支払を済ませるとローンの値はゼロになります。
- *iType*：支払期限を示します。次のいずれかの値になります。
  - *0*：支払が後払い（期間内の最後）の場合
  - *1*：支払が先払い（期間内の初め）の場合

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

注

注意: Rate およびNperパラメータの計算には同じ単位の支払金額を使用する必要があります。支払う金額（Pmtパラメータの金額）はマイナスの数値、受け取る総額はプラスの数値で表わされます。

PPMT()

内部BASICシンタックス

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

説明

定額の定期的な支払い、および一定利率に基づき、指定した支払日に返済する元金の金額を返します。

パラメータ

- *dblRate*：支払日ごとの利率。例えば、年利が6%のローンの毎月の支払日の利率は、次のようになります。

  0.06/12=0.005 または 0.5%

- *iPer*：計算する期間。１から*Nper*の数値の間で指定します。
- *iNper*：総支払回数
- *dblPV*：実際に支払わなければならない金額（総額）
- *dblFV*：支払が終了した時の差引残高または将来の支払用に確保する金額。一般に、特にローンを返済する場合は、このパラメータを「0」に設定します。実際、すべての支払を済ませるとローンの値はゼロになります。
- *iType*：支払期限を示します。次のいずれかの値になります。
  - *0*：支払が後払い（期間内の最後）の場合
  - *1*：支払が先払い（期間内の初め）の場合

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

注

注意: Rate およびNperパラメータの計算には同じ単位の支払金額を使用する必要があります。支払う金額（Pmtパラメータの金額）はマイナスの数値、受け取る総額はプラスの数値で表わされます。

PV()

内部BASICシンタックス

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

説明

定額の定期的な支払い、および固定利率に基づいて計算した、実際の年間支払総額を返します。

パラメータ

- *dblRate*：支払日ごとの利率。例えば、年利が6%のローンの毎月の支払日の利率は、次のようになります。

0.06/12=0.005 または 0.5%

- *iNper*：総支払回数
- *dblPmt*：1回の支払金額。支払金額には一般的に元金と利子が含まれます。
- *dblFV*：支払が終了した時の差引残高または将来の支払用に確保する金額。一般に、特にローンを返済する場合は、このパラメータを「0」に設定します。実際、すべての支払を済ませるとローンの値はゼロになります。
- *iType*：支払期限を示します。次のいずれかの値になります。
  - *0*：支払が後払い（期間内の最後）の場合
  - *1*：支払が先払い（期間内の初め）の場合

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

注

注意: Rate およびNperパラメータの計算には同じ単位の支払金額を使用する必要があります。支払う金額（Pmtパラメータの金額）はマイナスの数値、受け取る総額はプラスの数値で表わされます。

Randomize()

内部BASICシンタックス

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

説明

乱数発生関数を初期化します。

パラメータ

- *lValue*：特定の新しい初期値を指定して乱数発生関数であるRnd関数を初期化するときに使います。このパラメータを省略すると、システムクロックからの値が初期値として使われます。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim MyNumber
 Randomize
```

```
MyNumber= Int((10*Rnd)+1) : ' 1 - 10の乱数値を返します。
RetVal=MyNumber
```

RATE()

内部BASICシンタックス

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

説明

年間支払金額のうちの１回分の支払金額の利率を返します。

パラメータ

- *iNper*：総支払回数
- *dblPmt*：１回の支払金額。支払いには一般的に元金と利子が含まれます。
- *dblFV*：支払が終了した時の差引残高または将来の支払用に確保する金額。一般に、特にローンを返済する場合は、このパラメータを「0」に設定します。実際、すべての支払を済ませるとローンの値はゼロになります。
- *dblPV*：実際に支払わなければならない金額（総額）
- *iType*：支払期限を示します。次のいずれかの値になります。
  - *0*：支払が後払い（期間内の最後）の場合
  - *1*：支払が先払い（期間内の初め）の場合
- *dblGuess*：１回の支払の利率の推定値

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

注

注意: 支払う金額（Pmtパラメータの値）はマイナスの数値、受け取る総額はプラスの数値になります。この関数は、Guessパラメータに指定した値から開始して、繰り返し計算を実行します。20回繰り返しても結果が出ない場合は、この関数は無効となります。

RemoveRows()

内部BASICシンタックス

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

説明

*strRowNames*パラメータに指定されている行をリストから削除します。

この関数は、［ListBox］コントロールタイプの値を処理する時に役立ちます。このタイプのコントロールの値は、次の文字で区切られています。

- パイプ文字（|）は、列を区切ります。
- コンマ（,）は、行を区切ります。
- 各行の終わりには、等号（=）とその後に固有のIDが付いています。

パラメータ

- *strList*：処理する［ListBox］コントロールの値が含まれている文字列
- *strRowNames*：削除する行のID。IDが複数ある場合は、カンマで区切ります。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim MyStr
 MyStr=RemoveRows("a1|a2=a0,b1|b2=b0", "a0,c0") :' "b1|b2=b0"を返します。
 RetVal=MyStr
```

Replace()

内部BASICシンタックス

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

説明

*strData*パラメータの文字列に含まれる*strOldPattern*パラメータの文字列をすべて*strNewPattern*パラメータの文字列で置き換えます。*bCaseSensitive*パラメータを使って、検索する*strOldPattern*パラメータの文字列の大文字／小文字を区別できます。

パラメータ

- *strData*：置換される文字列を含んでいる文字列
- *strOldPattern*：*strData*パラメータに含まれている検索の対象となる文字列。
- *strNewPattern*：検索した文字列を置換する文字列
- *bCaseSensitive*：このパラメータを使って、大文字と小文字を区別する（=1）、または区別しない（=0）かを指定します。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim MyStr
 MyStr=Replace("youmeyoumeyou", "you", "me",0) :' "mememememe"を返します。
 MyStr=Replace("youmeyoumeyou", "You", "me",1) :' "youmeyoumeyou"を返しま
```

```
す。
 MyStr=Replace("youmeYoumeyou", "You", "me",1) :' "youmememeyou"を返します。
 RetVal=""
```

Right()

内部BASICシンタックス

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

説明

文字列の右端から*iNumber*で指定した数の文字を返します。

パラメータ

- *strString*：処理する文字列
- *iNumber*：返す文字数

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim lWord, strMsg, rWord, iPos :' Declare variables.
 strMsg = "Left() Test."
 iPos = InStr(1, strMsg, " ") : ' スペースを探す
 lWord = Left(strMsg, iPos - 1) : ' 左側の単語を取得
 rWord = Right(strMsg, Len(strMsg) - iPos) : ' 右側の単語を取得
 strMsg=rWord+lWord : ' 2つの単語を交換
 RetVal=strMsg
```

RightPart()

内部BASICシンタックス

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

説明

*strSep*パラメータに指定されている区切り文字の右側の文字列を１つ取得します。
右から左に向かって区切り文字を探します。
*bCaseSensitive*パラメータを使って、大文字と小文字を区別することもできます。

パラメータ

- *strFrom*：処理する文字列
- *strSep*：処理する文字列の区切り文字
- *bCaseSensitive*：このパラメータを使って、大文字と小文字を区別する（=1）、または区別しない（=0）かを指定します。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

文字列"This_is_a_test"に、LeftPart、LeftPartFromRight、RightPart、およびRightPartFromLeft関数を使った例を示します。

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

"This"を返します。

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

"This_is_a"を返します。

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

"test"を返します。

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

"is_a_test"を返します。

RightPartFromLeft()

内部BASICシンタックス

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

説明

*strSep*パラメータに指定されている区切り文字の右側の文字列を１つ取得します。

左から右に向かって区切り文字を探します。

*bCaseSensitive*パラメータを使って、大文字と小文字を区別することもできます。

パラメータ

- *strFrom*：処理する文字列
- *strSep*：処理する文字列の区切り文字
- *bCaseSensitive*：このパラメータを使って、大文字と小文字を区別する（=1）、または区別しない（=0）かを指定します。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

文字列"This_is_a_test"に、LeftPart、LeftPartFromRight、RightPart、およびRightPartFromLeft関数を使った例を示します。

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

"This"を返します。

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

"This_is_a"を返します。

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

"test"を返します。

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

"is_a_test"を返します。

RmDir()

内部BASICシンタックス

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

説明

既存のディレクトリを１つ削除します。

パラメータ

- *strRmDirectory*：削除するディレクトリのフルパス

戻りコード

- 0：成功
- 0以外：エラーコード

Rnd()

内部BASICシンタックス

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

説明

乱数を含んでいる値を返します。

パラメータ

- *dValue*：関数の実行モードを定義する場合に使うパラメータ
  - 0より小さい値：関数を実行するたびに同じ数値を発生します。
  - 0より大きい値：次に発生する乱数
  - 0：直前に発生した乱数
  - 省略：次に発生する乱数

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim MyNumber
 Randomize
 MyNumber= Int((10*Rnd)+1) : ' 1 - 10のいずれかの値を乱数として返します。
 RetVal=MyNumber
```

注

注意: この関数を呼び出す前に、パラメータなしでRandomize関数を使い、乱数発生関数を初期化する必要があります。

RTrim()

内部BASICシンタックス

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

説明

文字列の末尾に含まれるスペースをすべて取り除きます。

パラメータ

- *strString*：処理する文字列

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
' この例では、LtrimとRtrim関数を使って文字列変数から文字列の前後のスペースを削除します。
' また、Trim関数だけで前後のスペースを取り除く例も示します。
' ネストした関数と一緒にLcaseおよびUcase関数も使っています。

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->".
 ' Trimだけを使っても同じ結果が得られます。
 strTrimString = UCase(Trim(strString)) :' strTrimString = "<-TRIM->".
 RetVal= "¦" & strTrimString & "¦"
```

Second()

内部BASICシンタックス

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

説明

*tmTime*パラメータの時刻の秒の部分の数値を返します。

パラメータ

- *tmTime*：処理する「時刻+日付」形式のパラメータ

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim strSecond
 strSecond=Second(Date())
RetVal=strSecond : '現在の時刻の秒の部分を返します。例えば、時刻が15:45:30の場合は、"30"を返します。
```

SetSubList()

内部BASICシンタックス

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

説明

［ListBox］コントロールのサブリストの値を定義します。

パラメータ

- *strValues*：処理する［ListBox］コントロールの値が含まれている文字列
- *strRows*：*strValues*パラメータの文字列に追加する値またはその文字列に含まれている文字を置換する値の一覧。各値をパイプ文字（|）で区切ります。処理する行は、等号（=）記号の右側にあるIDで識別されます。不明な行は処理されません。
- *strRowFormat*：サブリストの書式化命令。各命令をパイプ文字（|）で区切ります。このパラメータには、次の文字を使います。
  - 1は、サブリストの最初の列の情報を示します。
  - i -jは、列のグループを定義します。
  - ハイフン（-）は、すべての列を処理することを示します。
  - 不明な列の値は返しません。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim MyStr
 MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "A2|A1=a0, B2|B1=b0", "2|1")
: ' "A1|A2|a3=a0,B1|B2|b3=b0,c1|c2|c3=c0"を返します。
 MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "Z2=*,B2=b0", "2") : '
"a1|Z2|a3=a0,b1|B2|b3=b0,c1|Z2|c3=c0"を返します。
 MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "B5|B6|B7=b0,C5|C6,C7=c0",
"5-7") : ' "a1|a2|a3=a0,b1|b2|b3||B5|B6|B7=b0,c1|c2|c3||C5|C6|C7=c0"を返します。
 MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "B1|B2|B3|B4=b0", "-") : '
"a1|a2|a3=a0,B1|B2|B3|B4=b0,c1|c2|c3=c0"を返します。
 MyStr=SubList("A|B|C,D|E|F", "X=*", "2") : ' "A|X|C,D|X|F"を返します。
 RetVal=""
```

Sgn()

内部BASICシンタックス

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

説明

数値の記号を表わす値を返します。

パラメータ

- *dValue*：記号を取得する数値

戻りコード

次のいずれかの値を返します。

- 1：0より大きい数値を示します。
- 0：0を示します。
- -1：0より小さい数値を示します。

例

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

Shell()

内部BASICシンタックス

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

説明

実行可能プログラムを起動します。

パラメータ

- *strExec*：起動する実行可能ファイルのフルパス

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

Sin()

内部BASICシンタックス

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

説明

数値のサインを返します。単位はラジアンです。

パラメータ

- *dValue*：サインを取得する数値

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

Space()

内部BASICシンタックス

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

説明

*iSpace*パラメータに指定されている数のスペース（空白文字）を挿入した文字列を作成します。

パラメータ

- *iSpace*：文字列に挿入するスペースの数

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim MyString
' 10個のスペースを返します。
 MyString = Space(10)
' ２つの文字列の間に10個のスペースを挿入します。
 MyString = "Space" & Space(10) &  "inserted"
RetVal=MyString
```

注

注意: この関数は、文字列を書式化したり、固定長の文字列から日付を削除する時に使います。

Sqr()

内部BASICシンタックス

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

説明

数値の平方根を返します。

パラメータ

- *dValue*：平方根を取得する数値

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

Str()

内部BASICシンタックス

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

説明

数値を文字列に変換します。

パラメータ

- *strValue*：文字列に変換する数値

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

StrComp()

内部BASICシンタックス

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

説明

２つの文字列を比較します。

パラメータ

- *strString1*：最初の文字列
- *strString2*：２つめの文字列
- *iOptionCompare*：比較のタイプ。バイナリの比較には「0」、テキストの比較には「１」を設定します。

戻りコード

- -1：*strString1*は*strString2*よりも大きい。
- 0：*strString1*は*strString2*と等しい。
- 1：*strString1*は*strString2*よりも小さい。

String()

内部BASICシンタックス

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

説明

*strString*文字を*iCount*回繰り返した文字列を返します。

パラメータ

- *iCount*：文字を繰り返す回数
- *strString*：繰り返す文字

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

SubList()

内部BASICシンタックス

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

説明

［ListBox］コントロールの値の文字列に含まれている値一覧のサブリストを返します。

パラメータ

- *strValues*：処理する［ListBox］コントロールの値が含まれている文字列
- *strRows*：サブリストに含める行のID。各IDをカンマ（,）で区切って指定します。特定のワイルドカード文字を使用できます。
  - （*）は、サブリスト内のすべての行を含むことを示します。
  - 不明なIDを指定した場合は、サブリストの空の値が返されます。
- *strRowFormat*：サブリストの書式化命令。各命令をパイプ文字（|）で区切ります。このパラメータには、次の文字を使います。
  - 1は、取得するサブリストのリストの最初の列の情報を示します。
  - 0は、取得するサブリストのリストの行のIDを示します。
  - アスタリスク（*）は、行のIDを除くすべての列の情報を示します。
  - 不明な列の値は返しません。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim MyStr
 MyStr=SubList("a1¦a2¦a3=a0,b1¦b2¦b3=b0,c1¦c2¦c3=c0", "a0,b0,a0", "3¦2¦3") : '
"a3¦a2¦a3,b3¦b2¦b3,a3¦a2¦a3"を返します。
 MyStr=SubList("a1¦a2¦a3=a0,b1¦b2¦b3=b0,c1¦c2¦c3=c0", "*", "*¦0") :'Returns
"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") : '
"a1¦a2¦a3=a0,b1¦b2¦b3=b0,c1¦c2¦c3=c0"を返します。
 MyStr=SubList("a1¦a2¦a3=a0,b1¦b2¦b3=b0,c1¦c2¦c3=c0", "*", "999=0") : '
"=a0,=b0,=c0"を返します。
 MyStr=SubList("a1¦a2¦a3=a0,b1¦b2¦b3=b0,c1¦c2¦c3=c0", "z0", "*=0") : ' ""を返しま
す。
 MyStr=SubList("a1¦a2¦a3=a0,b1¦b2¦b3=b0,c1¦c2¦c3=c0", "*", "=1") : ' "=a1,=b1,=c1"を
返します。
 MyStr=SubList("A¦B¦C,D¦E¦F", "*", "2=0") : ' "B,E"を返します。
 RetVal=""
```

Tan()

内部BASICシンタックス

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

説明

数値のタンジェントを返します。単位はラジアンです。

パラメータ

- *dValue*：タンジェントを取得する数値

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

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

Time()

内部BASICシンタックス

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

説明

現在の時刻を返します。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

Timer()

内部BASICシンタックス

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

説明

午前0時0分から経過した秒数を返します。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

TimeSerial()

内部BASICシンタックス

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

説明

iHour、*iMinute*、*iMinute*パラメータの形式に従って時刻を返します。

パラメータ

- *iHour*：時間
- *iMinute*：分
- *iSecond*：秒

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

各パラメータにはそれぞれ、日、月、または年を表す数式を使用できます。

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

例えば、上記の例は次の値を返します。

```
3:50:00
```

パラメータの値が予想される範囲（分ならば0 - 59、時間ならば0 - 24など）以外の値の場合は、次のレベルのパラメータに繰り上げられます。つまり、*iMinute*に75を入力すると、１時間と15分に解釈されます。

次の例は、

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

次の値を返します。

```
16:50:45
```

TimeValue()

内部BASICシンタックス

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

説明

「日付＋時刻」の値の時刻の部分を返します。

パラメータ

- *tmTime*：「日付＋時刻」形式の日付

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

次の例は、

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

次の値を返します。

```
15:00:00
```

ToSmart()

内部BASICシンタックス

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

説明

ソース文字列の各語の始めを大文字にします。

パラメータ

- *strString*：処理するソース文字列

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

Trim()

内部BASICシンタックス

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

説明

先頭と末尾のスペースを削除した文字列を返します。

パラメータ

- *strString*：処理する文字列

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
' この例では、LtrimとRtrim関数を使って文字列変数から文字列の前後のスペースを削除します。
' また、Trim関数だけで前後のスペースを取り除く例も示します。
' ネストした関数と一緒にLcaseおよびUcase関数も使っています。

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->".
' Trimだけを使っても同じ結果が得られます。
strTrimString = UCase(Trim(strString)) :' strTrimString = "<-TRIM->".
RetVal= "|" & strTrimString & "|"
```

UCase()

内部BASICシンタックス

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

説明

文字列に含まれるすべての小文字を大文字に変換して返します。

パラメータ

- *strString*：大文字に変換する文字列

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
' この例では、LtrimとRtrim関数を使って文字列変数から文字列の前後のスペースを削除します。
' また、Trim関数だけで前後のスペースを取り除く例も示します。
' ネストした関数と一緒にLcaseおよびUcase関数も使っています。

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->".
 ' Trimだけを使っても同じ結果が得られます。
```

```
strTrimString = UCase(Trim(strString)) :' strTrimString = "<-TRIM->".
RetVal= "¦" & strTrimString & "¦"
```

UnEscapeSeparators()

内部BASICシンタックス

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

説明

１つの文字列からすべてのエスケープ文字を削除します。

パラメータ

- *strSource*：処理する文字列
- *strEscChar*：削除するエスケープ文字

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim MyStr
 MyStr=UnEscapeSeparators("you \¦me \¦you \¦", " \¦") :' "you¦me¦you¦"を返します。
RetVal=""
```

Union()

内部BASICシンタックス

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

説明

区切り文字で区切られている２つの文字列を合体します。重複する文字列は削除されます。

パラメータ

- *strListOne*：最初の文字列
- *strListTwo*：２番目の文字列
- *strSeparator*：各文字列の区切り文字
- *strEscChar*：エスケープ文字。この文字が区切り文字の前に付くと、その区切り文字は無視されます。

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim MyStr
 MyStr=Union("a1¦a2,b1¦b2", "a1¦a3,b1¦b2", ",", "\") :' "a1¦a2,b1¦b2,a1¦a3"を返します。
 MyStr=Union("a1¦a2,b1¦b2", "a1¦a3\,b1¦b2", ",", "\") :' "a1¦a2,b1¦b2,a1¦a3\,b1¦b2"を返します。
 RetVal=""
```

UTCToLocalDate()

内部BASICシンタックス

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

説明

UTCフォーマットの日付（タイムゾーンに関係しない）「日付＋時刻」型の日付に変換します。

パラメータ

- *tmUTC*：UTCフォーマットの日付

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

Val()

内部BASICシンタックス

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

説明

数値を表す文字列を倍精度型に変換します。

パラメータ

- *strString*：変換する文字列

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

例

```
Dim strYear
Dim dYear as Double
 strYear=Year(Date())
dYear=Val(strYear)
 RetVal=dYear :'Returns the current year
```

WeekDay()

内部BASICシンタックス

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

説明

*tmDate*パラメータの日付の曜日の部分を返します。

パラメータ

- *tmDate*：処理する「時刻+日付」形式のパラメータ

戻りコード

「1」は日曜日、「2」は月曜日、…、「7」は土曜日のように、週の曜日に対応した数値を返します。

例

```
Dim strWeekDay
 strWeekDay=WeekDay(Date())
 RetVal=strWeekDay : ' 曜日を返します。
```

Year()

内部BASICシンタックス

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

説明

*tmDate*パラメータの日付の年の部分を返します。

パラメータ

- *tmDate*：処理する「時刻+日付」形式のパラメータ

戻りコード

エラーの場合は、Connect-Itのログファイルにエラーメッセージを書込みます。

III. 索引

6 使用可能な関数 - 機能： すべて

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

- CSng
- CStr
- CurDir
- CVar
- Date
- DateAdd
- DateAddLogical
- DateDiff
- DateSerial
- DateValue
- Day
- EscapeSeparators
- ExeDir
- Exp
- ExtractValue
- FileCopy
- FileDateTime
- FileExists
- FileLen
- Fix
- FormatResString
- FV
- GetListItem
- Hex
- Hour
- InStr
- Int
- IPMT
- IsNumeric
- Kill
- LCase
- Left
- LeftPart
- LeftPartFromRight
- Len
- LocalToBasicDate

- LocalToBasicTime
- LocalToBasicTimeStamp
- LocalToUTCDate
- Log
- LTrim
- MakeInvertBool
- Mid
- Minute
- MkDir
- Month
- Name
- Now
- NPER
- Oct
- ParseDate
- ParseDMYDate
- ParseMDYDate
- ParseYMDDate
- PifFirstInCol
- PifGetBlobSize
- PifGetElementChildName
- PifGetElementCount
- PifGetInstance
- PifGetItemCount
- PifIgnoreDocumentMapping
- PifIgnoreNodeMapping
- PifIsInMap
- PifLogInfoMsg
- PifLogWarningMsg
- PifMapValue
- PifMapValueContaining
- PifNewQueryFromFmtName
- PifNewQueryFromXml
- PifNodeExists
- PifQueryClose
- PifQueryGetDateVal

- PifQueryGetDoubleVal
- PifQueryGetIntVal
- PifQueryGetLongVal
- PifQueryGetStringVal
- PifQueryNext
- PifRejectDocumentMapping
- PifRejectNodeMapping
- PifSetDateVal
- PifSetDoubleVal
- PifSetLongVal
- PifSetStringVal
- PifStrVal
- PifUserFmtStrToVar
- PifUserFmtVarToStr
- PMT
- PPMT
- PV
- Randomize
- RATE
- RemoveRows
- Replace
- Right
- RightPart
- RightPartFromLeft
- RmDir
- Rnd
- RTrim
- Second
- SetSubList
- Sgn
- Shell
- Sin
- Space
- Sqr
- Str
- StrComp

- String
- SubList
- Tan
- Time
- Timer
- TimeSerial
- TimeValue
- ToSmart
- Trim
- UCase
- UnEscapeSeparators
- Union
- UTCToLocalDate
- Val
- WeekDay
- Year

7 使用可能な関数 - 機能： PIF

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

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

8 使用可能な関数 - 機能： 組み込み

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

- CSng
- CStr
- CurDir
- CVar
- Date
- DateSerial
- DateValue
- Day
- EscapeSeparators
- ExeDir
- Exp
- ExtractValue
- FileCopy
- FileDateTime
- FileExists
- FileLen
- Fix
- FormatResString
- FV
- GetListItem
- Hex
- Hour
- InStr
- Int
- IPMT
- IsNumeric
- Kill
- LCase
- Left
- LeftPart
- LeftPartFromRight
- Len
- LocalToBasicDate
- LocalToBasicTime
- LocalToBasicTimeStamp
- LocalToUTCDate

- Log
- LTrim
- MakeInvertBool
- Mid
- Minute
- MkDir
- Month
- Name
- Now
- NPER
- Oct
- ParseDate
- ParseDMYDate
- ParseMDYDate
- ParseYMDDate
- PMT
- PPMT
- PV
- Randomize
- RATE
- RemoveRows
- Replace
- Right
- RightPart
- RightPartFromLeft
- RmDir
- Rnd
- RTrim
- Second
- SetSubList
- Sgn
- Shell
- Sin
- Space
- Sqr
- Str

- StrComp
- String
- SubList
- Tan
- Time
- Timer
- TimeSerial
- TimeValue
- ToSmart
- Trim
- UCase
- UnEscapeSeparators
- Union
- UTCToLocalDate
- Val
- WeekDay
- Year