

HP Connect-It

ソフトウェアバージョン : 3.81

プログラム用参考ガイド



法的制限事項

Copyrights

© Copyright 1994-2007 Hewlett-Packard Development Company, L.P.

権利の制限

機密コンピュータソフトウェア。

所持、使用、またはコピーには、HPが発行する有効なライセンスが必要です。

FAR 12.211および12.212に基づき、市販コンピュータソフトウェア、コンピュータソフトウェアドキュメンテーション、および市販品目の技術データは、ベンダの標準商用ライセンスに従って米国政府にライセンスされます。

保証

HP製品およびサービスに対する保証は、当該製品およびサービスに付属の明示的保証規定に記載されているものに限られます。

本書のいかなる内容も当該保証に新たに保証を追加するものではありません。

HPは本書の技術上または編集上の誤謬、欠落についての責任を負わないものとします。

ここに記載されている情報は、予告なしに変更されることがあります。

商標

- Adobe®, Adobe Photoshop® and Acrobat® are trademarks of Adobe Systems Incorporated.
- Corel® and Corel logo® are trademarks or registered trademarks of Corel Corporation or Corel Corporation Limited.
- Java™ is a US trademark of Sun Microsystems, Inc.
- Linux is a U.S. registered trademark of Linus Torvalds
- Microsoft®, Windows®, Windows NT® and Windows® XP are U.S. registered trademarks of Microsoft Corporation.
- Oracle® is a registered US trademark of Oracle Corporation, Redwood City, California.
- UNIX® is a registered trademark of The Open Group.

目次

1. はじめに	9
1. プログラミングの基本	11
変数の概要	11
制御構造	16
演算子	20
ファイル管理	24
2. 適用分野	29
3. 表記規則	31
記法	31
スクリプト中の「日付+時刻」の形式	32
スクリプト中の"Duration"型定数の形式	32
4. 定義	35
関数の定義	35
エラーコードの定義	35
5. 関数の型とパラメータ	37
型のリスト	37

関数の型	37
パラメータの型	38

II. 関数の説明 39

6. 関数の説明 41

Abs()	41
AppendOperand()	42
ApplyNewVals()	43
Asc()	43
Atn()	44
BasicToLocalDate()	45
BasicToLocalTime()	45
BasicToLocalTimeStamp()	46
Beep()	46
CDbl()	47
ChDir()	47
ChDrive()	48
Chr()	48
CInt()	49
CLng()	50
Cos()	51
CountOccurrences()	52
CountValues()	52
CSng()	53
CStr()	54
CurDir()	55
CVar()	55
Date()	56
DateAdd()	56
DateAddLogical()	57
DateDiff()	57
DateSerial()	58
DateValue()	59
Day()	60
EscapeSeparators()	60
ExeDir()	61
Exp()	62
ExtractValue()	62
FileCopy()	63
FileDateTime()	64
FileExists()	64
FileLen()	65

Fix()	66
FormatDate()	66
FormatResString()	67
FV()	68
GetEnvVar()	69
GetListItem()	70
GetXmlAttributeValue()	71
GetXmlElementValue()	71
Hex()	72
Hour()	73
InStr()	73
Int()	74
IPMT()	75
IsNumeric()	76
Kill()	77
LCase()	77
Left()	78
LeftPart()	79
LeftPartFromRight()	80
Len()	81
LocalToBasicDate()	82
LocalToBasicTime()	82
LocalToBasicTimeStamp()	83
LocalToUTCDate()	83
Log()	84
LTrim()	84
MakeInvertBool()	85
Mid()	86
Minute()	87
MkDir()	87
Month()	88
Name()	89
Now()	89
NPER()	90
Oct()	91
ParseDate()	92
ParseDMYDate()	93
ParseMDYDate()	93
ParseYMDDate()	94
PifCloseODBCDatabase()	95
PifCreateDynaMappableFromFmtName()	96
PifDateToTimezone()	97
PifExecODBCSql()	100
PifFirstInCol()	101
PifGetBlobSize()	103

PifGetDateVal()	103
PifGetDoubleVal()	104
PifGetElementChildName()	105
PifGetElementCount()	106
PifGetHexStringFromBlob()	107
PifGetInstance()	108
PifGetIntlStringVariantVal()	108
PifGetIntVal()	109
PifGetItemCount()	110
PifGetLongVal()	111
PifGetOpenSessionTime()	112
PifGetStringFromBlob()	112
PifGetStringVal()	113
PifGetVariantVal()	114
PifIgnoreCollectionMapping()	115
PifIgnoreDocumentMapping()	116
PifIgnoreDocumentReconc()	117
PifIgnoreNodeMapping()	118
PifIgnoreNodeReconc()	120
PifIgnoreSubDocumentReconc()	120
PifIsInMap()	121
PifLogInfoMsg()	122
PifLogWarningMsg()	123
PifMapValue()	124
PifMapValueContaining()	126
PifMapValueContainingEx()	129
PifMapValueEx()	131
PifNewQueryFromFmtName()	132
PifNewQueryFromXml()	134
PifNodeExists()	137
PifOpenODBCDatabase()	137
PifQueryClose()	139
PifQueryGetDateVal()	140
PifQueryGetDoubleVal()	141
PifQueryGetIntVal()	142
PifQueryGetLongVal()	143
PifQueryGetStringVal()	144
PifQueryNext()	145
PifRejectCollectionMapping()	147
PifRejectDocumentMapping()	148
PifRejectDocumentReconc()	149
PifRejectNodeMapping()	150
PifRejectNodeReconc()	151
PifRejectSubDocumentReconc()	152
PifScenarioPath()	153

PifSetDateVal()	153
PifSetDoubleVal()	154
PifSetLongVal()	155
PifSetNullVal()	156
PifSetPendingDocument()	157
PifSetStringVal()	158
PifStrVal()	160
PifUserFmtStrToVar()	160
PifUserFmtVarToStr()	162
PifWriteBlobInFile()	163
PifXMLDump()	164
PMT()	166
PPMT()	167
PV()	168
Randomize()	169
RATE()	170
RemoveRows()	171
Replace()	172
Right()	173
RightPart()	174
RightPartFromLeft()	175
RmAllInDir()	176
Rmdir()	177
Rnd()	178
RoundValue()	179
RTrim()	180
Second()	181
SetMaxInst()	181
SetSubList()	182
Sgn()	183
Shell()	184
Sin()	185
Space()	186
Sqr()	187
Str()	187
StrComp()	188
String()	189
SubList()	189
Tan()	191
Time()	192
Timer()	192
TimeSerial()	193
TimeValue()	194
ToSmart()	194
Trim()	195

UCase()	196
UnEscapeSeparators()	197
Union()	198
UTCToLocalDate()	199
Val()	199
WeekDay()	200
Year()	201
III. 索引	203
使用可能な関数 - 全関数のリスト	205
使用可能な関数 - Connect-It	209
使用可能な関数 - 組み込み	211

1 はじめに

1 プログラミングの基本

この章では、Connect-Itで使用できるBasic言語を使ったプログラミングの基本について説明します。プログラミングの経験があり、他の言語を使用したことがある方なら、この章の内容の大半はすでにご存じのはずです。ただし、Connect-ItのBasicでは従来の機能の一部が意図的に除外されたり制限されたりしているため、この章全体に目を通しておくことをお勧めします。

変数の概要

変数は、プログラムの実行中にデータを保管するために用いられます。変数は次の情報によって識別されます。

- 名前。変数に格納されている値を参照するために用いられます。
- 型。変数に格納できるデータの種類を指定します。

一般的に、変数には次の2種類があります。

- 配列。
- スカラ変数。これは配列以外のすべての変数です。

変数の宣言

変数を使用するためには、明示的に宣言する必要があります。宣言の構文は次のとおりです。

```
Dim <変数の名前> [As <変数の型>]
```



注意:

ConnectIt Basicでの変数の明示的な宣言は、Microsoft Visual Basicで*Option Explicit*を使用するのと同じです。

変数名は次の制約を満たす必要があります。

- 先頭は英大文字または英小文字
- 長さ40文字以下
- 名前に含まれる文字は、英字A-Z、a-z、数字0-9、下線文字 ("_")



注意:

アクセント付きの文字も使用できますが、使用しないことをお勧めします。

- 予約語は使用できません。例えば、Basicの関数や節の名前は予約語です。オプションのAs節は、変数の型を定義します。型は変数に格納できる情報の種類を指定します。使用可能なデータ型としては、*String*、*Integer*、*Variant*などがあります。

As節を省略した場合、変数は*Variant*型と見なされます。

単一宣言

単一宣言の場合、各宣言文は、次の例のように1つの変数を表します。

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

複合宣言

複合宣言の場合、各宣言文は、次の例のように、任意の数の変数を表すことができます。

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



注意:

すでに説明したように、変数の型を指定しない場合、デフォルトで変数は*Variant*と見なされます。したがって、上の例の2行目では、変数AとBは*Variant*型、Cは*Integer*型です。

データ型

下の表は、関数やパラメータで使用できるデータ型の一覧です。

型	説明
Integer	-32,768～+32,767の整数
Long	-2,147,483,647～+2,147,483,646の整数
Single	4バイトの浮動小数点数（単精度）
Double	8バイトの浮動小数点数（倍精度）
String	任意の文字が使用できる文字列
Date	日付または日付+時刻
Variant	任意の型を表現できる一般型

数値型

Connect-ItのBasic言語では、数値型としてInteger、Long、Single、Doubleを用意しています。数値型はVariantよりもメモリ使用量が少ないのが普通です。

変数に格納する値が常に整数（123など）であって小数（3.14など）でない場合、IntegerまたはLongと宣言するのが適しています。これらのデータ型の演算は、他の型の場合よりも高速で、メモリ使用量が少なくなります。特に、ループで使用するカウンタには、これらのデータ型が最適です。

変数に小数を格納する場合、SingleまたはDoubleとして宣言します。



注意:

浮動小数点数（SingleまたはDouble）には、丸め誤差が生じる可能性があります。

String型

変数に文字列だけを格納する場合は、Stringとして宣言します。

```
Dim MyString As String
```

これにより、この変数に文字列を格納し、専用の文字列処理関数を使ってその内容を操作できます。

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

デフォルトでは、String型の変数のサイズは可変です。文字列を格納するために割り当てられるサイズは、変数に代入されるデータのサイズに応じて変化します。ただし、次の構文を使ってString型の変数を宣言することもできます。

```
Dim <変数名> As String * <格納される文字列のサイズ>
```

次の例では、20文字を格納する変数を宣言しています。

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

この変数に20文字未満の文字列を格納した場合、指定したサイズになるまで文字列の末尾にスペースが追加されます。一方、20文字を超える文字列を格納した場合、21文字目以降は切り捨てられます。

Variant型

*Variant*型は一般的な型であり、他のすべての型の代わりとなります。各データ型と*Variant*との間の変換を意識する必要はありません。変換は次の例のように自動的に実行されます。

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

変換は自動的に行われますが、次の規則を守る必要があります。

- *Variant*に対して算術演算を実行する場合、内容は数値でなければなりません。数値は文字列で表されていてもかまいません。
- *Variant*に対して連結演算を行うには、+演算子でなく&演算子を使用します。*Variant*には、2つの特殊な値を格納できます。空値とNull値です。

空値

*Variant*に最初に値を代入する前には、空値が格納されています。この値は特別な値であり、0、空文字列、Null値のいずれでもありません。*Variant*に空値が格納されているかどうかを確認するには、次の例のようにBasic関数**IsEmpty()**を使用します。

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

空値を持つ*Variant*は式の中で使用できます。空値は状況に応じて値0または空文字列として扱われます。*Variant*に空値を再代入するには、次の例のようにキーワード**Empty**を使用します。

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

Null値

Null値は、データベース中で欠けている値や不明の値を示すために多く用いられます。この値には次のような特別な性質があります。

- Null値を含む式は、常にNull値を返します。これを式におけるNull値の伝搬といいます。式の一部でもNullの場合は、式全体がNullになります。
- 一般則として、関数のパラメータをNullに設定した場合、関数はNull値を返します。

データ配列

配列を使うと、変数の集合を1つの名前でも参照し、番号（添字）を使って各要素を一意に識別できます。配列の要素はすべて同じデータ型でなければなりません。例えば、*String*と*Double*の両方の値を含む配列は作成できません。*Variant*型を使えばこの制約を回避できます。

配列の宣言

配列は変数の集合です。

規則として、次の表現は次のように表されます。

- 配列の下限：最初の要素の添字



注意:

デフォルトでは、配列の下限は0です。

- 配列の上限：最後の要素の添字



注意:

配列の上限は、*Long*の最大値（2,147,483,646）を超えることはできません。

配列の宣言は、次のように変数の宣言と似ています。

```
Dim <配列の名前>(<配列の上限>) [As <配列内の変数のデータ型>]
```

例：

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

次の宣言を使って、配列の下限を指定することもできます。

```
Dim <配列の名前>(<配列の下限> To <配列の上限>) [As <配列内の変数のデータ型>]
```

例：

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

制限

Connect-It Basicの配列には、次の制限があります。

- 可変サイズの配列はサポートされません。特に、配列のサイズを実行時に変更することはできません。
- 多次元配列はサポートされません。

制御構造

制御構造は、その名前からわかるように、プログラムの実行を制御するためのものです。制御構造には次の2種類があります。

- 判定構造：特定の条件の結果としてプログラムの行き先を決定します。
- ループ構造：特定の条件に基づいてプログラムの一部分を繰り返します。

判定構造

判定構造は、テストの結果に基づいて条件付きで命令を実行します。次の判定構造が使用できます。

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

If...Then

この構造は、1つまたは複数の命令を条件付きで実行するために使用します。この構造の構文では、単一行の文と複数行の文が使用できます。単一行の文では、1つの命令しか実行できません。

```
If <条件> Then <命令>
```

```
If <条件> Then  
<命令群>  
End If
```

条件は通常は比較ですが、数値を返す任意の式が使用できます。この値は、Basicコードによって**True**または**False**と解釈されます。**False**は0に対応します。他のすべての値は**True**と見なされます。

条件が**True**に評価された場合、キーワード**Then**の後の命令または命令群が実行されます。

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

この構造は、複数の条件つき命令ブロックを定義するために使用します。**True**に評価された最初のブロックだけが実行されます。

```
If <条件1> Then  
<命令群1>  
ElseIf <条件2> Then  
<命令群2>  
...  
Else
```



```
<命令群N>  
End If
```

最初の条件がテストされ、結果が**False**に評価された場合、2番目の条件がテストされ、同様にしてどれかの条件が**True**に評価されるまで続きます。キーワード**Then**の後の命令セットが実行されます。

キーワード**Else**は省略可能です。これは、すべての条件が**False**に評価された場合に実行される命令群を定義します。

注意:

判定構造の**Elself**は、必要なだけいくつでもネストできます。ただし、常に同じ式を異なる値と比較し続ける場合は、判定構造の構文が必要以上に複雑で読みにくくなる可能性があります。このような場合、**Select...Case**型の判定構造を使用することをお勧めします。

Select...Case

この構造は、前の判定構造と同じ目的で使用されますが、一般的にコードがより読みやすくなります。**Select...Case**は構造の先頭で1つのテストを実行し、テスト結果を構造内の各**Case**に与えられた値と比較します。一致するものがあった場合、その**Case**に対応する命令セットが実行されます。

```
Select Case <テスト>  
[Case <値リスト1>  
<命令群1>  
[Case <値リスト2>  
<命令群2>  
...  
[Case Else  
<命令群n>  
End Select
```

各値リストは、カンマで区切った値のリストです。複数の**Case**キーワードにテスト結果と一致する値がある場合、最初に一致する**Case**に対応する命令セットだけが実行されます。

Case Elseキーワードに対応する命令セットは、**Case**キーワードの中に一致するものがなかった場合に実行されます。

ループ構造

ループ構造は、一連の命令を繰り返し実行するために使用します。次のループ構造が使用できます。

■ Do...Loop

■ For...Next

Do...Loop

この構造は、一連の命令を不定の回数実行するために使用します。ループは、条件が満たされた場合、あるいは満たされなかった場合に終了します。この条件は、**False** (0) または **True** (0以外) に評価される値または式です。

注意:

ループを強制的に終了するには、実行される命令の中で **Exit Do** キーワードを使用します。

この構造にはいくつかの変種がありますが、最も一般的なのは次のものです。

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

この場合、条件が最初に評価されます。結果が **True** の場合、命令群が実行され、プログラムは **Do While** キーワードに戻り、条件をもう一度評価し、以下同様に続きます。条件が **False** に評価されるとループは終了します。

次の例は、ループの1回の反復ごとに加算されるカウンタの値をテストします。カウンタが20に達するとループは終了します。

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

次の例は前の例を基にしていますが、カウンタの値が10の場合に **Exit Do** キーワードを使ってループを強制的に終了します。

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

このタイプの **Do...Loop** 構造では、命令群を実行する前に条件が評価されます。命令を実行した後で条件をテストしたい場合は、次の **Do...Loop** 構造を使用します。

```
Do  
<命令群>  
Loop While <条件>
```

 **注意:**

このタイプの構造では、命令群が少なくとも1回実行されることが保証されます。

前の2つの**Do...Loop** 構造は、条件が**True**である間反復します。条件が**False**である間反復したい場合は、次の構造のどれかを使用します。

```
Do Until <条件>
<命令群>
Loop

Do
<命令群>
Loop Until <条件>
```

この構造タイプを使うと、前の例は次のように書けます。

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

For...Next

この構造は、一連の命令を実行する回数が定義されていない場合に使用します。

Do...Loopと異なり、**For...Next**ループは、カウンタと呼ばれる変数を使用します。この変数の値は、反復ごとに加算あるいは減算されます。

 **注意:**

ループを強制的に終了するには、実行される命令の中で**Exit For**キーワードを使用します。

```
For <カウンタ> = <初期値> To <最終値> [Step <増分>]
<命令群>
Next [<カウンタ>]
```

 **重要項目:**

引数カウンタ、初期値、最終値、増分はすべて数値で表されます。



注意:

増分は正または負の値です。正の値の場合、初期値が最終値以下でないと命令群は実行されません。負の値の場合、初期値が最終値以上でないと命令群は実行されません。増分を指定しない場合、デフォルトで1に設定されます。

For...Nextループが実行されると、以下の動作が行われます。

- 1 カウンタが初期化され、初期値が格納されます。
- 2 Basicコードは、カウンタの値が最終値よりも大きいかどうかをテストします。結果が真の場合、プログラムはループを終了します。



注意:

増分が負の場合、Basicはカウンタの値が最終値よりも小さいかどうかをテストします。

- 3 命令群が実行されます。
 - 4 カウンタに1または指定された値が加算されます。
 - 5 動作2~4が繰り返されます。
- 次の動作は、1000以下のすべての偶数を加算します。

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

次の例は前の例を基にしていますが、カウンタの値が500のときに**Exit For**キーワードを使ってループを強制的に終了します。

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

演算子

演算子とは、単純な演算（加算、乗算など）を変数に対して実行したり、比較を行ったりするための記号です。演算子にはいくつかの種類があります。

- 代入演算子
- 算術演算子
- 関係演算子（比較演算子とも呼ぶ）

- 論理演算子

代入演算子

このタイプの演算子は、変数に値を代入するために使用します。Connect-It Basicの代入演算子は"="記号1つだけです。代入構文は次のとおりです。

```
<変数> = <値>
```

算術演算子

算術演算子は、変数の値を算術的に変更したり、2つの式の間で単純な算術演算を実行したりするために使用します。

+演算子

この演算子は、2つの値を加算します。構文は次のとおりです。

```
<結果> = <式1> + <式2>
```

 **注意:**

この演算子には、2つの数値の加算と、文字列の連結という2つの用途があります。あいまいさを避けるため、この演算子は数値の加算にのみ使用し、文字列の連結には&演算子を使用することをお勧めします。

-演算子

この演算子は、2つの値の減算を行うため、または値の符号を反転するため（単項演算子）に使用します。この演算子には2つの構文があります。

```
<結果> = <式1> - <式2>
```

または

```
- <式>
```

*演算子

この演算子は、2つの値を乗算します。構文は次のとおりです。

```
<結果> = <式1> * <式2>
```

/演算子

この演算子は、2つの値の除算を実行します。構文は次のとおりです。

```
<結果> = <式1> / <式2>
```

^演算子

この演算子は、べき乗演算を実行します。構文は次のとおりです。

```
<結果> = <式1> ^ <式2>
```

注意:

この構文で、式2が整数の場合、式1は負の値を取ることはできません。式が複数の指数演算を連続的に実行する場合、演算は左から右の順番で論理的に解釈されます。

Mod演算子

この演算子は、2つの値のユークリッド除算の余りを計算します。構文は次のとおりです。

```
<結果> = <式1> Mod <式2>
```

注意:

浮動小数点数は自動的に整数に丸められます。

次の例は4を返します（6.8は最も近い整数7に丸められます）。

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

関係演算子

関係演算子は、2つの値を比較します。次の表に関係演算子の一覧を示します。

演算子	名称	説明	構文
=	等価演算子	2つの値を比較し、等しいかどうかを確認します。	<式1> = <式2>
<	小なり演算子	ある値がもう1つの値より厳密に小さいかどうかをテストします。	<式1> < <式2>
<=	小なりまたは等しい演算子	ある値がもう1つの値より小さいかまたは等しいかどうかをテストします。	<式1> <= <式2>
>	大なり演算子	ある値がもう1つの値より厳密に大きいかどうかをテストします。	<式1> > <式2>

演算子	名称	説明	構文
>=	大なりまたは等しい演算子	ある値がもう1つの値より大きいかまたは等しいかどうかをテストします。	<式1> >= <式2>
<>	不等価演算子	ある値がもう1つの値と異なるかどうかをテストします。	<式1> <> <式2>

論理演算子

論理演算子は、複数の条件を評価するために使用します。

And演算子

この演算子は、2つの式の論理AND（両方の条件が真）を実行します。構文は次のとおりです。

```
<結果> = <式1> And <式2>
```

両方の式（オペランド）が`True`に評価された場合、結果は`True`です。どちらかの式が`False`に評価された場合、結果は`False`です。

Or演算子

この演算子は、2つの式の論理OR（どちらかの条件が真）を実行します。構文は次のとおりです。

```
<結果> = <式1> Or <式2>
```

どちらかの式が`True`に評価された場合、結果は`True`です。

Xor演算子

この演算子は、2つの式の論理XOR（どちらか一方の条件だけが真）を実行します。構文は次のとおりです。

```
<結果> = <式1> Xor <式2>
```

どちらか一方の式だけが`True`に評価された場合、結果は`True`です。

Not演算子

この演算子は、式の論理否定を実行します。構文は次のとおりです。

```
<結果> = Not <式1>
```

式が`True`に評価された場合、結果は`False`です。式が`False`に評価された場合、結果は`True`です。

演算子の優先順位

複数の演算子を組み合わせて使用する場合、式は次の優先順位に従って評価されます。演算子は下に行くほど優先順位が低くなります。

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

ファイル管理

Connect-It Basicでは、単純なファイル管理が可能です。最も一般的な操作（読み取り、書き込みなど）が標準で使用できます。

ファイルの概要

ファイルとは、プログラムが外部のオブジェクトを参照する方法です。ファイルは論理レコードの集合です。レコードは構造化されている場合もされていない場合もあり、プログラムはレコードに対して一群の基本的な操作（読み取り、書き込みなど）を実行できます。論理レコードは、1つの基本操作で扱うことができる最小のデータセットを表します。

Connect-Itで扱えるのは、いわゆるシーケンシャルファイルだけです。シーケンシャルファイルに対する主な操作は、次のレコードの読み取りと、ファイルの末尾への新規レコードの追加です。レコードの読み取りと書き込みを同時に行うことはできません。

読み取りの場合、カーソルはシーケンシャルファイルの最初の論理レコードに置かれます。読み取り操作を行うたびに、レコードがプログラムの内部領域（通常は変数）に転送され、カーソルはファイル中の次のレコードに進みます。読み取るレコードがまだ残っているかどうかを判定する操作（**EOF**節：End Of File）が用意されています。

書き込みの場合、シーケンシャルファイルは空であるか、カーソルがファイルの最後のレコードの後に置かれます。書き込み操作を行うたびに、プログラムの内部領域（通常は変数）に格納されたデータがファイル中のレコードに転送され、カーソルはそのレコードの後に移動します。



注意:

シーケンシャルファイルの最大の特徴の1つは、レコードが書き込まれた順序で読み取られることです。

ファイルのオープンとクローズ

Open節

これはファイル操作のための主要な節です。これにより、ファイルの読み取り、作成、書き込みが可能になります。構文は次のとおりです。

```
Open <ファイルのパス> For <モード> [Access <アクセスタイプ>] As [#]<ファイル番号>
```

この節のパラメータの詳細を次の表に示します。

パラメータ	説明
<ファイルのパス>	操作の対象となるファイルを指定する文字列。この文字列は、ファイルのフルパスを表すことができます。
<モード>	ファイルの処理モードを指定します。このパラメータは次のいずれかの値を取ります。 <ul style="list-style-type: none">■ <i>Input</i> : ファイルは読み取りモードで開かれます。■ <i>Output</i> : ファイルは書き込みモードで開かれます。ファイルがすでに存在する場合、既存の内容は上書きされます。■ <i>Append</i> : ファイルは書き込みモードで開かれます。ファイルがすでに存在し、既存の内容がある場合、新しい内容はファイルの末尾に追加されます。

パラメータ

<アクセスタイプ>

説明

開いたファイルに対して実行できる操作を指定します。ファイルが別のプロセスによって開かれており、指定したアクセスが許可されない場合、ファイルのオープンコマンドは失敗します。このパラメータは次の値のいずれかに設定できます。

- **Read** : ファイルは読み取り専用アクセス用に開かれます。
- **Write** : ファイルは書き込み専用アクセス用に開かれます。
- **Read Write** : ファイルは読み取り/書き込みモードで開かれます。このタイプのアクセスは、**Append**モードでのみ使用可能です。

<ファイル番号>

ファイルを識別する1~511の固有の番号。

FreeFile()関数を使って、使用可能な次のファイル番号を調べることができます。

注意:

次の点に注意してください。

- ファイルの読み書き操作を行うには、まず**Open**節でファイルを開いておく必要があります。
- **Append**、**Binary**、**Output**モードでは、参照したファイルが存在しない場合、ファイルが作成されます。
- **Binary**または**Input**モードでは、ファイルを閉じずに別の番号で開くことができます。**Append**または**Output**モードでは、ファイルを閉じないと別の番号で開くことはできません。

Close節

この節では、**Open()**節で開いたファイルを閉じることができます。構文は次のとおりです。

```
Close [<ファイルのリスト>]
```

オプションの<ファイルのリスト>引数には、1つまたは複数のファイル番号を指定します。このオプション引数の構文は次のとおりです。

```
[[#]<ファイル番号>],[#]<ファイル番号>]...
```

注意:

このパラメータを省略した場合、**Open()**節で開かれているすべてのアクティブなファイルが閉じられます。

ファイルからのデータの読み取り

ファイルからデータを読み取るには2つの節が使用できます。どちらの節を使用するかは、ファイルに対して指定したアクセスモードによって決まります。2つの節は次のとおりです。

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

Input節

この節は、*Binary*または*Input*モードで開かれたファイルから、指定した数の文字を読み取るために使用します。この節の構文は次のとおりです。

```
Input (<読み取る文字数>,[#]<ファイル番号>)
```

Line Input節

この節は、シーケンシャルファイルから1行分のデータを読み取り、*String*または*Variant*型の変数に格納するために使用します。この節の構文は次のとおりです。

```
Line Input #<ファイル番号>, <変数名>
```

重要項目:

この節は、キャリッジリターンまたはキャリッジリターン+改行が見つかるまで、文字を1つずつ読み取ります。

ファイルへのデータの書き込み

ファイルにデータを書き込むには、1つの節**Print**が使用できます。この節の構文は次のとおりです。

```
Print #<ファイル番号>, [<データ>]
```

注意:

Print節は、ラインフィード文字（ASCIIコード10）が見つかるまでデータをファイルの1行に書き込みます。次に例を示します。

```
Open "c:\test.txt" For Output As #1
Print #1, "This is a test" & Chr(10)
Print #1, "And now we can close the file..." & Chr(10)
Close #1
```

2 適用分野

この文書に記述された関数は、Connect-Itのすべてのスクリプトウィンドウで使用できます。

3 表記規則

この章では、本書で使用する記法と、一部の定数の形式について説明します。

記法

本書の例では、次の記法が使用されています。

□	角括弧はオプションのパラメータを示します。 角括弧はコマンドには入力しません。 例外：Basicスクリプトで、角括弧がデータベース中のデータのパスを次の形式で示す場合： <i>[Link.Link.Field]</i>
<>	かぎ括弧は、パラメータを通常言語で示します。括弧は入力しません。テキストを適切な情報に置き換えてください。
{ }	中括弧は、ノードの定義またはプロパティに対する複数行のスクリプトブロックを囲みます。
	縦棒は、中括弧内の使用可能なパラメータの組を区切ります。

次のテキストスタイルは特別な意味を持ちます。

Fixed width
characters

DOSコマンド、関数パラメータ、データフォーマット

例	コードまたはコマンドの例。
...	省略されたコードまたはコマンド。
<i>Object name</i>	フィールド、タブ、メニュー、ファイルの名前は太字で示されます。
注意:	重要な注記。
注記	

スクリプト中の「日付+時刻」の形式

スクリプト中で参照される日付は、ユーザが指定した表示オプションと無関係に、国際形式で表されます。

yyyy/mm/dd hh:mm:ss

例：

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

 注意:

ハイフン ("-") を日付の区切り文字として使用することもできます。

BasicとUnixの日付

BasicとUnixでは日付の扱いが異なります。

- Basicでは、日付は国際形式または浮動小数点数 ("Double"型) で表します。この場合、数値の整数部が1899年12月30日午前0時からの経過日数を表し、小数部が現在の日付の割合 (日の始まりからの経過秒数を86400で割った値) を表します。
- Unixでは、日付は長整数 ("Long"型) で表します。これは、1870年1月1日午前0時からの経過秒数をタイムゾーンと無関係に (UTC時で) 示します。

スクリプト中の"Duration"型定数の形式

スクリプト中では、経過時間は秒単位で記憶され、表現されます。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用します。

```
RetVal=259200
```

同様に、経過時間を計算する関数は秒単位の数値を返します。



会計計算の場合、Connect-Itは最も一般的な単純化を考慮します。この場合のみ、1年は12ヶ月、1ヶ月は30日と見なされます（すなわち、1年 = 360日）。

4 定義

この章では、いくつかの基本的な用語の定義をまとめます。

関数の定義

関数とは、一連の動作を実行し、ユーザに値を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内蔵のBasicで関数を呼び出すための構文の例を次に示します。

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

エラーコードの定義

関数が失敗した場合、エラーコードが返されます。

最後のエラーとそのコードの説明を調べるには、**Err.Description**と**Err.Number**を使用します。

エラーメッセージを意図的に発生させるには、**Err.Raise**関数を使用します。構文は次のとおりです。

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

次の表に、一般的なエラーコードを示します。

エラーコード	意味
12001	未定義エラー
12002	関数に対する無効なパラメータ
12003	無効なハンドルまたはオブジェクトが削除された
12004	使用可能なデータなし。このエラーは通常、クエリの実行時に発生します。クエリがデータを返さない場合、このエラーが発生します。
12006	無効な値 (パラメータに対する型が正しくないなど)
12009	廃止された、または実装されていない関数

5 関数の型とパラメータ

型のリスト

次の表は、関数またはパラメータで使用できる型の一覧です。

型	説明
Integer	-32,768～+32,767の整数
Long	-2,147,483,647～+2,147,483,646の整数
Single	4バイトの浮動小数点数（単精度）
Double	8バイトの浮動小数点数（倍精度）
String	任意の文字が使用できる文字列
Date	日付または日付+時刻
Variant	任意の型を表現できる一般型

関数の型

関数の型は、関数の戻り値の型に一致します。この点には十分な注意が必要です。プログラムのコンパイルエラーや実行時エラーの原因となる場合があるからです。

パラメータの型

関数で使用するパラメータにも型があり、関数が正しく動作するためには適切な型を使用する必要があります。関数の構文では、パラメータの前に型に基づくプレフィックスがつきます。次の表に、型と対応するプレフィックスの一覧を示します。

型	Basic構文で使用するプレフィックス
Integer	"i"
Long	"l"
Double	"d"
String	"str"
Date	"dt"
Variant	"v"

II 関数の説明

6 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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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** : サプ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、Connect-Itログにメッセージが書き込まれます。

Asc()

内部Basicシンタックス

Function Asc(strAsc As String) As Long

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

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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(IValue 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

- **IValue** : 変換する式

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

例

```
Dim dCalc as Double  
dCalc=Cos(2.79)  
RetVal=dCalc
```

注

 **注意:**

度からラジアンへの変換公式は次になります:

```
ラジアンでの角度 = (度での角度) * Pi / 180
```

CountOccurrences()

内部Basicシンタックス

Function CountOccurrences(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、Connect-Itログにメッセージが書き込まれます。

例

```
Dim MyStr
MyStr=CountOccurrences("you | me | you,me | you", "you", ","):"2"を返します。
MyStr=CountOccurrences("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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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** : 経過時間の追加先の日付
- **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、Connect-Itログにメッセージが書き込まれます。

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** : 経過時間の追加先の日付
- **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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

例

次の例では、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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

例

```
Dim MyStr
MyStr=EscapeSeparators("you | me | you,me | you", "| \", "\"):"you\ | me\ | yo
u\,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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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

- **psrData** : 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、Connect-Itログにメッセージが書き込まれます。

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 : 正常実行。
- 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、Connect-Itログにメッセージが書き込まれます。

FileExists()

内部Basicシンタックス

Function FileExists(strFileName As String) As Long

説明

ファイルが存在するかどうかテストします。

- 0 : ファイルが見つからない
- 1 : ファイルが見つかった

パラメータ

- **strFileName** : 存在するかテストするファイルの完全パス

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

例

```
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()

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

例

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

FormatDate()

内部Basicシンタックス

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

説明

strFormatパラメータの式に従って日付を書式化します。

パラメータ

- **tmFormat** : 書式化する日付
- **strFormat** : 書式化命令が含まれている式

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

例

次の例は、日付を書式化するコードの例です。

```
Dim MyDate
MyDate="2000/03/14"
RetVal=FormatDate(MyDate, "dddd d mmmm yyyy") : "Tuesday 14 March 2000"を返します。
```

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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。

GetEnvVar()

内部Basicシンタックス

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

説明

環境変数の値を返します。環境変数が存在しない場合は、空の値が返されます。

パラメータ

- **strVar** : 環境変数の名前
- **bExpand** : このブール型パラメータは、環境変数が1つまたは複数の別の環境変数を参照する場合にのみ有用です。この際、パラメータの値が1であると (デフォルト値)、参照される各変数は値に変換されます。それ以外の場合、変数はそのまま変換されません。

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

例

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

GetListItem()

内部Basicシンタックス

Function GetListItem(strFrom As String, strSep As String, INb 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**INb**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
- **INb** : 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、Connect-Itログにメッセージが書き込まれます。

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。

GetXmlAttributeValue()

内部Basicシンタックス

Function GetXmlAttributeValue(strVal As String) As String

説明

この関数は、与えられたXML要素値に対して無効な文字をエスケープします。

パラメータ

このパラメータは、エスケープする文字を指定します。

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

例

例えば、次の関数は単一引用符を有効な文字に変換します。

```
GetXmlAttributeValue("doesn't")
```

結果は次のとおりです。

```
"doesn't"
```

GetXmlElementValue()

内部Basicシンタックス

Function GetXmlElementValue(strElemContent As String) As String

説明

この関数は、与えられたXML属性値に対して無効な文字をエスケープします。

パラメータ

このパラメータは、エスケープする文字を指定します。

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

例

この関数は次のようにWHERE節で使用できます。

```
strLayer = strLayer + "<Where Path='ItemsUsed'>" + GetXmlElementValue("AssetTag like 'A%'" ) + "</Where>"
```

この関数のコメント付きの例については次を参照：[PifNewQueryFromFmtName\(\)](#) [献 132]

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

例

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

InStr()

内部Basicシンタックス

Function InStr(lPosition As Long, strSource As String, strPattern As String, bCaseSensitive As Long) 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

- **lPosition**: 検索の開始位置。このパラメータは必須であり、65,535以下の有効な自然数である必要があります。
- **strSource**: 処理する文字列
- **strPattern**: 検索する文字列
- **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、Connect-Itログにメッセージが書き込まれます。

例

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

注

 **注意:**

最初の一致する文字列の位置は必ず1です。検索対象の文字列が見つからない場合、関数は0を戻します。

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、Connect-Itログにメッセージが書き込まれます。

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）

- **dbIFV** : 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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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 : 正常実行。
- 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、Connect-Itログにメッセージが書き込まれます。

例

```
' 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()

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**: 戻される文字数。

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

例

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

LocalToDate()

内部Basicシンタックス

Function LocalToDate(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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

例

```
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()

内部Basicシンタックス

Function MakeInvertBool(IValue 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

- **IValue** : 処理する数値

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

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内から展開する文字列の開始位置。
- **ILen**: 展開する文字列の長さ。

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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 : 正常実行。
- 0以外 : エラーコード。

例

```
Dim lErr as Long
'c:\tmpディレクトリの作成
lErr = MkDir("c:\tmp")
```

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、Connect-Itログにメッセージが書き込まれます。

例

```
Dim lMonth as Long
lMonth=Month(Date())
RetVal=lMonth : '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、Connect-Itログにメッセージが書き込まれます。

例

```
Dim lErr as Long
' "C:\tmp\src.txt"を"D:\tmp\dst.txt"に名前変更
lErr = Name("C:\tmp\src.txt", "D:\tmp\dst.txt")
```

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

例

```
Dim dDate as Date  
dDate = ParseDMYDate("31/02/2003")
```

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、Connect-Itログにメッセージが書き込まれます。

例

```
Dim dDate as Date  
dDate = ParseMDYDate("02/31/2003")
```

ParseYMDDate()

内部Basicシンタックス

Function ParseYMDDate(strDate As String) As Date

説明

この関数は、(Basicで識別される)Dateオブジェクトをyyyy/mm/dd形式で返します。

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、Connect-Itログにメッセージが書き込まれます。

例

```
Dim dDate as Date  
dDate = ParseYMDDate("2003/02/31")
```

PifCloseODBCDatabase()

内部Basicシンタックス

Function PifCloseODBCDatabase(strDSN As String) As Long

説明

この関数はODBC接続を終了します。

パラメータ

- **strDSN** : ODBC接続のデータソース名。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

例

```
Dim iRet As Integer
iRet = PifOpenOdbcDatabase(PifStrVal("DSN"), PifStrVal("LOGIN"), PifStrVal("PWD"))

if iRet = 0 Then
Dim strQuery As String
strQuery = "SELECT a.AssetName, a.FullName FROM Asset a"
iRet = PifCreateDynaMapTable(PifStrVal("DSN"), "Assets", strQuery, FALSE)
End If

Dim strRes as String
if iRet = 0 Then
strRes = PifMapValue([FullName], "Assets", 1, "", 1)
End If

If iRet = 0 and strRes = "" Then
iRet = PifExecODBCSql(PifStrVal("DSN"), "INSERT INTO Asset(AssetName, AssetFullName)
```

PifCreateDynaMappableFromFmtName()

内部Basicシンタックス

Function PifCreateDynaMappableFromFmtName(strCntrName As String, strFmtName As String, strLayer As String, strMappableName As String, strKeyName As String, bCreateOnce As Long) As Long

説明

この関数は、セッション開始時に動的マップテーブルを作成するために使用します。

パラメータ

- **strCntrName** : コネクタ名。
- **strFmtName** : マップテーブルの内容を定義する作成するドキュメントタイプ。
- **strLayer** : ドキュメントタイプに対して使用されるWhere節。
- **strMappableName** : 作成するマップテーブルの名前。
- **strKeyName** : 作成するマップテーブルのキーとして使用される列名。
列名をキーとして指定しない場合、マップテーブルの作成に使用されたクエリの最初の属性が用いられます。

例えば、ポートフォリオ項目の識別子を使用する場合、構文は次のようになります。

```
Portfolio.AssetTag
```

マップテーブルが現在のドキュメントテーブルとして用いられるため、ドキュメントタイプの名前を指定する必要はありません。

- **bCreateOnce**:
 - **0** : マップテーブルは関数が呼び出されるたびに作成されます。
 - **1** : マップテーブルはセッション全体に対して作成されます。

戻りコード

- **0** : 正常実行。
- **0以外** : エラーコード。

注

関連項目：

- [PifMapValueEx\(\)](#) [献 131]
- [PifMapValueContainingEx\(\)](#) [献 129]

PifDateToTimezone()

内部Basicシンタックス

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

説明

この関数は、指定されたタイムゾーンで表された日付（日付と時刻）を、別のタイムゾーンに変換します。必要な場合、夏時間設定が考慮されます。

指定されたタイムゾーンが存在しない場合、エラーメッセージが返されます。

 **注意:**

夏時間のないGMT 0日付を指定するには、'UTC'タイムゾーン（協定世界時）を使用します。

パラメータ

- **tmSrc**：このパラメータは、変換する日付を指定します。
- **strTmznSrc**：このパラメータには、変換前のタイムゾーンの名前を指定します。
- **bWithDayLightSavingSrc**：このパラメータの値に応じて、この関数は変換前のタイムゾーンの夏時間設定を考慮（=1）または無視（=0）します。
- **strTmznDst**：このパラメータには、変換後のタイムゾーンの名前を指定します。
- **bDayLightSavingDst**：このパラメータの値に応じて、この関数は変換後のタイムゾーンの夏時間設定を考慮（=1）または無視（=0）します。

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

例

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

は次の値を返します：

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

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

は次の値を返します：

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

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

は次の値を返します：

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

注

タイムゾーンのリストと説明：

名前	説明
Dateline Standard Time	(GMT-12:00) エニウェトク、クウェジェリン
Samoa Standard Time	(GMT-11:00) ミッドウェー諸島、サモア
Hawaiian Standard Time	(GMT-10:00) ハワイ
Alaskan Standard Time	(GMT-09:00) アラスカ
Pacific Standard Time	(GMT-08:00) 太平洋標準時 (米国、カナダ)、ティファナ
US Mountain Standard Time	(GMT-07:00) アリゾナ
Mountain Standard Time	(GMT-07:00) 山地標準時 (米国、カナダ)
Central America Standard Time	(GMT-06:00) 中央アメリカ
Central Standard Time	(GMT-06:00) 中部標準時 (米国、カナダ)
Mexico Standard Time	(GMT-06:00) メキシコシティ
Canada Central Standard Time	(GMT-06:00) サスカチエワン
SA Pacific Standard Time	(GMT-05:00) ボゴタ、リマ、キト
Eastern Standard Time	(GMT-05:00) 東部標準時 (米国、カナダ)
US Eastern Standard Time	(GMT-05:00) インディアナ (東部)
Atlantic Standard Time	(GMT-04:00) 大西洋標準時 (カナダ)

名前	説明
SA Western Standard Time	(GMT-04:00) カラカス、ラパス
Pacific SA Standard Time	(GMT-04:00) サンティアゴ
Newfoundland Standard Time	(GMT-03:30) ニューファンドランド
E. South America Standard Time	(GMT-03:00) ブラジリア
SA Eastern Standard Time	(GMT-03:00) ブエノスアイレス、ジョージタウン
Greenland Standard Time	(GMT-03:00) グリーンランド
Mid-Atlantic Standard Time	(GMT-02:00) 中部大西洋岸諸州
Azores Standard Time	(GMT-01:00) アゾレス諸島
Cape Verde Standard Time	(GMT-01:00) カーボベルデ諸島
Greenwich Standard Time	(GMT) カサブランカ、モンロビア
GMT Standard Time	(GMT) グリニッジ平均時：ダブリン、エディンバラ、リスボン、ロンドン
UTC	(GMT) 協定世界時
W. Europe Standard Time	(GMT+01:00) アムステルダム、ベルリン、ベルン、ローマ、ストックホルム、ウィーン
Central Europe Standard Time	(GMT+01:00) ベオグラード、プラティスラバ、ブダペスト、リュブリャナ、プラハ
Romance Standard Time	(GMT+01:00) ブリュッセル、コペンハーゲン、マドリード、パリ
Central European Standard Time	(GMT+01:00) サラエボ、スコピエ、ソフィア、ピリニウス、ワルシャワ、ザグレブ
W. Central Africa Standard Time	(GMT+01:00) 西中央アフリカ
GTB Standard Time	(GMT+02:00) アテネ、イスタンブール、ミンスク
E. Europe Standard Time	(GMT+02:00) ブカレスト
Egypt Standard Time	(GMT+02:00) カイロ
South Africa Standard Time	(GMT+02:00) ハラーレ、プレトリア
FLE Standard Time	(GMT+02:00) ヘルシンキ、リガ、タリン
Jerusalem Standard Time	(GMT+02:00) エルサレム
Arabic Standard Time	(GMT+03:00) バグダード
Arab Standard Time	(GMT+03:00) クウェート、リヤド
Russian Standard Time	(GMT+03:00) モスクワ、サンクトペテルブルク、ボルゴグラード
E. Africa Standard Time	(GMT+03:00) ナイロビ
Iran Standard Time	(GMT+03:30) テヘラン
Arabian Standard Time	(GMT+04:00) アブダビ、マスカット
Caucasus Standard Time	(GMT+04:00) バクー、トビリシ、イエレバン
Afghanistan Standard Time	(GMT+04:30) カブール
Ekaterinburg Standard Time	(GMT+05:00) エカテリンブルク
West Asia Standard Time	(GMT+05:00) イスラマバード、カラチ、タシケント
India Standard Time	(GMT+05:30) カルカッタ、チェンナイ、ボンベイ、ニューデリー
Nepal Standard Time	(GMT+05:45) カトマンドゥー
N. Central Asia Standard Time	(GMT+06:00) アルマトゥイ、ノボシビルスク

名前	説明
Central Asia Standard Time	(GMT+06:00) アスタナ、ダッカ
Sri Lanka Standard Time	(GMT+06:00) スリ・ジャヤワルダナプラ
Myanmar Standard Time	(GMT+06:30) ラングーン
SE Asia Standard Time	(GMT+07:00) バンコク、ハノイ、ジャカルタ
North Asia Standard Time	(GMT+07:00) クラスノヤルスク
China Standard Time	(GMT+08:00) 北京、重慶、香港、ウルムチ
North Asia East Standard Time	(GMT+08:00) イルクーツク、ウランバートル
Malay Peninsula Standard Time	(GMT+08:00) クアラルンプール、シンガポール
W. Australia Standard Time	(GMT+08:00) パース
Taipei Standard Time	(GMT+08:00) 台北
Tokyo Standard Time	(GMT+09:00) 大阪、札幌、東京
Korea Standard Time	(GMT+09:00) ソウル
Yakutsk Standard Time	(GMT+09:00) ヤクーツク
Cen. Australia Standard Time	(GMT+09:30) アデレード
AUS Central Standard Time	(GMT+09:30) ダーウィン
E. Australia Standard Time	(GMT+10:00) ブリスベーン
AUS Eastern Standard Time	(GMT+10:00) キャンベラ、メルボルン、シドニー
West Pacific Standard Time	(GMT+10:00) グアム、ポートモレスビー
Tasmania Standard Time	(GMT+10:00) ホバート
Vladivostok Standard Time	(GMT+10:00) ウラジオストク
Central Pacific Standard Time	(GMT+11:00) マガダン、ソロモン諸島、ニューカレドニア
New Zealand Standard Time	(GMT+12:00) オークランド、ウェリントン
Fiji Standard Time	(GMT+12:00) フィジー、カムチャッカ、マーシャル諸島
Tonga Standard Time	(GMT+13:00) ヌクアロファ

PifExecODBCSql()

内部Basicシンタックス

Function PifExecODBCSql(strDSN As String, strSqlQuery As String) As Variant

説明

この関数は、ODBCデータベースに対してSQLクエリを実行します。

パラメータ

- **strDSN** : 接続するODBCデータソースの名前。
- **strSqlQuery** : ODBCデータベースに対して実行するSQLクエリ。

戻りコード

この関数は、クエリの最初の結果を返します。クエリに一致する結果がない場合はNULL値を返します。

例

```
Dim iRet As Integer
iRet = PifOpenOdbcDatabase(PifStrVal("DSN"), PifStrVal("LOGIN"), PifStrVal("PWD"))

if iRet = 0 Then
Dim strQuery As String
strQuery = "SELECT a.AssetName, a.FullName FROM Asset a"
iRet = PifCreateDynaMapTable(PifStrVal("DSN"), "Assets", strQuery, FALSE)
End If

Dim strRes as String
if iRet = 0 Then
strRes = PifMapValue([FullName], "Assets", 1, "", 1)
End If

If iRet = 0 and strRes = "" Then
iRet = PifExecODBCSql(PifStrVal("DSN"), "INSERT INTO Asset(AssetName, AssetFullName)
```

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

この関数は、**strChildCond**に指定された条件に基づいて、コレクション中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のパス。
- **strChildCond** : 要素の検索条件。条件は次の形式です :

<パス> = <値>

ここで:

- <パス>はコレクション内の要素のパス
- <値>はConnect-Itと同じロケールの文字列で与えられた値

■  **注意:**

nをコレクションの要素数とすると、**iStartCount**は0~n-1の範囲内であればなりません。

iStartCount : 検索開始インデックス。

戻りコード

- 0 : 正常実行。
- 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 <> 0 Then
iToTal = iToTal + 1
End If
Loop Until iIndex = 0
```

PifGetBlobSize()

内部Basicシンタックス

Function PifGetBlobSize(strPath As String) As Long

説明

この関数は、BLOBオブジェクトのサイズを返します。

パラメータ

- **strPath** : このパラメータは、コレクション中のBLOBオブジェクトのフルパスを指定します。

戻りコード

この関数は、フルパスで識別されるBLOBオブジェクトのサイズを返します。

例

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

PifGetDateVal()

内部Basicシンタックス

Function PifGetDateVal(strPath As String, bCkeckNodeExists As Long) As Date

説明

この関数は、strPathパラメータで定義された値を取り出して、データ型基本変数に格納します。

パラメータ

strPath: 日付型ノード型ドキュメントを指定します。

bCkeckNodeExists: この関数は、返されたデータにデータノードが存在するかどうかを確認するのに使用します。

戻りコード

この関数は、日付型フォーマットに変換された値を返します。

例

```
Dim DateVal As Date
DateVal = PifGetDateVal("field containing a date")
RetVal = DateVal
```

PifGetDoubleVal()

内部Basicシンタックス

Function PifGetDoubleVal(strPath As String, bCkeckNodeExists As Long) As Double

説明

この関数は、strPathパラメータで定義された値を取り出して倍精度実数に変換します。

パラメータ

strPath: 変換するデータノードのパスです。

bCkeckNodeExists: この関数は、返されたデータにデータノードが存在するかどうかを確認するのに使用します。

戻りコード

この関数は、倍精度型数値に変換された値を返します。

例

```
RetVal = PifGetDoubleVal("field containing a double type number")
```

PifGetElementChildName()

内部Basicシンタックス

**Function PifGetElementChildName(strPath As String, iltem As Long)
As String**

説明

この関数は、ソースドキュメントタイプの指定されたノードのn番目のサブ要素の名前を返します。

パラメータ

- **strPath** : このパラメータは、操作の対象となるノードのフルパスを指定します。
- **iltem** : このパラメータは、名前を取得したいサブ要素の数を指定します。

戻りコード

この関数は、要素の名前を返します。

注意:

パラメータ**iltem**が負の値であるかノード中のサブ要素の数よりも大きい場合、トラッキング行にエラーが記録され、関数は空文字列を返します。

例

次のソースドキュメントタイプに対して :



```
pifGetElementChildName("FileInfo",0)
```

は"FileName"を返します。

```
pifGetElementChildName("FileInfo",1)
```

は"ModificationDate"を返します。

```
pifGetElementChildName("FileInfo",2)
```

は空文字列を返し、トラッキング行にエラーを記録します。

PifGetElementCount()

内部Basicシンタックス

Function PifGetElementCount(strPath As String) As Long

説明

この関数は、ソースドキュメントタイプの指定されたノードのサブ要素の数を返します。

パラメータ

- **strPath** : このパラメータは、操作の対象となるノードのフルパスを指定します。

戻りコード

この関数は、パスが**strPath**で指定されるノードのサブ項目の数を返します。

例

次のソースドキュメントタイプに対して：



次のスクリプト：

```
Dim iChildCount as Integer
iChildCount = PifGetElementCount("FileInfo")
```

は2を返します。**FileInfo**要素には**FileName**と**ModificationDate**の2個のサブ要素があるからです。

PifGetHexStringFromBlob()

内部Basicシンタックス

Function PifGetHexStringFromBlob(strPath As String, bCkeckNodeExists As Long) As Long

説明

この関数は、ソースドキュメントのバイナリロングオブジェクト（BLOB）を16進文字列に格納します。バイナリオブジェクトの各ビットが16進形式（2文字）で文字列に格納されます。例えば、10進値"27"の16進表現は"1B"です。

パラメータ

- **strPath** : このパラメータは、ソースドキュメント中のバイナリ要素（BLOB）のフルパスを指定します。
- **bCkeckNodeExists** : このパラメータは、ソースドキュメントにバイナリ要素が見つからない場合にエラーを記録する（=TRUE）かしない（=FALSE）かを指定します。このパラメータのデフォルト値はFALSEです。

戻りコード

この関数は、バイナリ要素の16進表現を返します。

例

一部のLDAPサーバでは、エントリがバイナリ要素"ObjectGUID"によって一意に定義されます。下のスクリプトは、このバイナリ値を宛先コネクタの文字列内部にインポートし、それを更新キーとして使用します。

```
RetVal = PifGetHexStringFromBlob("ObjectGUID", TRUE)
```

PifGetInstance()

内部Basicシンタックス

Function PifGetInstance() As String

説明

コレクションからコレクションへのマッピングでは、この関数はコレクションで現在処理されている要素の番号を返します。最初に処理される要素が番号"0"です。

戻りコード

処理されている要素の番号が文字列で返されます。

例

次のコレクションマッピング (**Hardware.Bus.IdentificationCard**への) に対して：



次のスクリプト：

```
Dim strMyElement as String  
strMyElement= "Item #" & PifGetInstance
```

は次の値を返します：**Item #0**、**Item #1**、**Item #2**、など

PifGetIntlStringVariantVal()

内部Basicシンタックス

Function PifGetIntlStringVariantVal(strPath As String, bCkeckNodeExists As Long) As String

説明

この関数は、**strPath**パラメータで定義された値を取り出して国際文字列に変換します。

パラメータ

strPath: 変換するデータノードのパスです。

bCkeckNodeExists: この関数は、返されたデータにデータノードが存在するかどうかを確認するのに使用します。

戻りコード

この関数は、国際文字列フォーマットに変換された値を返します。

例

```
Dim strTest As String
strTest = PifGetIntlStringVariantVal("int64")
PifLogInfoMsg(CStr([int64]))
RetVal = strTest
```

この関数は、64ビット整数値を文字列フォーマットに変換し、その結果を**strTest**に保存します。ログに表示される値は281478209994880です。これはBasic関数(CStr)で倍精度値として処理される2.8147820999488e+014に相当します。

注

この関数は、64ビット整数をサポートしないBasicエンジンに関する既知の問題を回避するために使用されます。

PifGetIntVal()

内部Basicシンタックス

Function PifGetIntVal(strPath As String, bCkeckNodeExists As Long) As Long

説明

この関数は、**strPath**パラメータで定義された値を取り出して整数に変換します。

パラメータ

strPath: 変換するデータノードのパスです。

bCkeckNodeExists: この関数は、返されたデータにデータノードが存在するかどうかを確認するのに使用します。

戻りコード

この関数は、整数に変換された値を返します。

例

```
Dim IValue As Int
IValue = PifGetIntVal("field containing an integer")
RetVal = IValue
```

PifGetItemCount()

内部Basicシンタックス

Function PifGetItemCount(strPath As String) As Long

説明

この関数は、完全に識別されたコレクションの要素数を返します。

パラメータ

- **strPath** : コレクションのフルパス。

戻りコード

この関数は、コレクションの要素数を返します。

コレクションが存在しない場合、この関数は"0"を返します。

例

次のドキュメントタイプに対して：



スクリプト：

```
Dim iCount As Integer  
iCount = PifGetItemCount("Hardware.Bus.IdentifiedCard")
```

は**Hardware.Bus.IdentifiedCard**コレクションのサブ要素数を返します。例えば、下のドキュメントの場合、スクリプトは"8"を返します。



PifGetLongVal()

内部Basicシンタックス

Function PifGetLongVal(strPath As String, bCkeckNodeExists As Long) As Long

説明

この関数は、**strPath**パラメータで定義された値を取り出して倍長整数に変換します。

パラメータ

strPath: 変換するデータノードのパスです。

bCkeckNodeExists: この関数は、返されたデータにデータノードが存在するかどうかを確認するのに使用します。

戻りコード

この関数は、整数に変換された値を返します。

例

```
Dim IValue As Long
IValue = PifLongVal("field containing a long integer")
RetVal = IValue
```

PifGetOpenSessionTime()

内部Basicシンタックス

Function PifGetOpenSessionTime() As Date

説明

この関数は、セッション開始時の日付と時刻を返します。

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

例

```
pifLogInfoMsg("Session start time is: " & pifGetSessionStartTime & "")
```

PifGetStringFromBlob()

内部Basicシンタックス

Function PifGetStringFromBlob(strPath As String, bCkeckNodeExists As Long) As Long

説明

この関数は、blobからstrPathパラメータで定義されている値を取得し、それを文字列に変換します。

パラメータ

strPath: 変換するデータノードのパスです。

bCkeckNodeExists: この関数は、返されたデータにデータノードが存在するかどうかを確認するのに使用します。

戻りコード

この関数は、文字列に変換された値を返します。

例

```
Dim StrValue As String
StrValue = PifGetStringFromBlob("field containing a blob")
RetVal = StrValue
```

PifGetStringVal()

内部Basicシンタックス

Function PifGetStringVal(strPath As String, bCkeckNodeExists As Long) As String

説明

この関数は、strPathパラメータで定義された値を取り出して文字列に変換します。

パラメータ

strPath: 変換するデータノードのパスです。

bCkeckNodeExists: この関数は、返されたデータにデータノードが存在するかどうかを確認するのに使用します。

戻りコード

この関数は、文字列フォーマットに変換された値を返します。

例

```
Dim StrValue As String
StrValue = PifGetStringVal("amComputer.LogicalDrives(3).Name")
RetVal = StrValue
```

コレクションアイテムのフィールドの値を取得するのに使用する構文を以下に挙げます。

```
PifGetStringVal(a.b(index).c)
```

ここで、**c**はフィールド、**b**がコレクションで、**index**は対象の番号です。
amComputer.LogicalDrivesコレクションのNameフィールドの一例を以下に挙げます。

```
PifGetStringVal(amComputer.LogicalDrives(3).Name)
```

この構文では、コレクションにある3番目の論理ディスクドライブ (LogicalDrive(3)) の名前が取得されます。

PifGetVariantVal()

内部Basicシンタックス

```
Function PifGetVariantVal(strPath As String, bCkeckNodeExists As Long) As Variant
```

説明

この関数は、strPathパラメータで定義された値を取り出してバリエーションに変換します。

パラメータ

strPath: 変換するデータノードのパスです。

bCkeckNodeExists: この関数は、返されたデータにデータノードが存在するかどうかを確認するのに使用します。

戻りコード

この関数は、バリエーションに変換された値を返します。

例

```
strTest = RetVal = PifGetVariantVal("field containing a variant")
```

PifIgnoreCollectionMapping()

内部Basicシンタックス

Function PifIgnoreCollectionMapping(strMsg As String) As Long

説明

この関数は、コレクションからコレクションへのマッピングでのみ、コレクションを無視するために使用します。

なお、**PifIgnoreNodeMapping()** 関数を使えば、単にコレクションの現在の要素を無視することができます。

strMsg パラメータで指定した情報メッセージをログファイルに出力できます。

パラメータ

- **strMsg** : オプションのパラメータ。ログに出力される文字列。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

例

次の例では、要素の1つで`quantity`ノードの値が'0'に設定されている場合、コレクションのすべての要素が無視されます。

```
if [root.item(pifGetInstance).quantity] = 0 then  
pifIgnoreCollectionMapping  
end if
```

PifIgnoreNodeMapping()との比較として、次の例では、コレクションの要素のうち、`quantity`ノードの値が'0'に設定されているものだけが無視されます。

```
if [root.item(pifGetInstance).quantity] = 0 then
pifIgnoreNodeMapping
end if
```

PifIgnoreDocumentMapping()

内部Basicシンタックス

Function PifIgnoreDocumentMapping(strMsg As String) As Long

説明

この関数は、ドキュメントを無視するために使用します。

strMsg パラメータで指定した情報メッセージをログに出力できます。

生成されたドキュメントはドキュメントログに記録されますが、次のコネクタには伝達されません。

パラメータ

- **strMsg** : オプションのパラメータ。ログに出力される文字列。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

例

例えば、**BarCode** フィールドが空であるドキュメントを無視したいが、**AssetTag** フィールドの値はトラッキング行に記録したい場合、次のスクリプトを**BarCode** ノードに対して使用できます。

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

このスクリプトを下のドキュメントに対して使用すると：



次のドキュメントがドキュメントログに記録されます（このドキュメントは次のコネクタには伝達されません）。



さらに、トラッキング行に情報メッセージが記録されます。

PiflgnoreDocumentReconc()

内部Basicシンタックス

Function PiflgnoreDocumentReconc(strMsg As String) As Long

説明

この関数は、照合更新操作でドキュメントの無視に使用されます。挿入や更新は実行されません。**strMsg**パラメータに格納されているメッセージをログに表示できます。



注意:

この関数は非パラレル化モードでのみ使用できます。

パラメータ

- **strMsg** : このパラメータは、ログに表示するメッセージを指定します。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

注



注意:

この関数は更新スクリプトでのみ使用できます。

PiflgnoreNodeMapping()

内部Basicシンタックス

Function PiflgnoreNodeMapping(strMsg As String) As Long

説明

この関数は、ドキュメント中のノードをスキップするために使用します。
strMsg パラメータで指定した情報メッセージをログに出力できます。



警告:

この関数はドキュメントのルートノードに対しては使用できません。ドキュメント全体を無視するには、 **PiflgnoreDocumentMapping** 関数を使用します。

パラメータ

- **strMsg** : オプションのパラメータ。ログに出力される文字列。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

例

次のスクリプトは、**[Comment]**フィールドが空の場合に、現在のノードを無視し、メッセージをログに記録します。

```
If [Comment] = "" Then  
PiflgnoreNodeMapping("The current node was not processed")  
Else
```

```
RetVal = [Comment]
End If
```



例えば、マッピングソースドキュメントが次のものの場合:



PifignoreNodeMapping()関数を使用しないと、次のドキュメントが生成され
ます。



PifignoreNodeMapping()を使用すると、次のドキュメントが生成されます。



注

 **注意:**

ノードは処理されないため、ログに出力されるメッセージはスキップされるノ
ードの親に書き込まれます。

PifIgnoreNodeReconc()

内部Basicシンタックス

Function PifIgnoreNodeReconc(strMsg As String) As Long

説明

この関数は、照合更新操作で現在のノード（サブドキュメント、コレクション、構造体など）の無視に使用されます。挿入や更新は実行されません。**strMsg**パラメータに格納されているメッセージをログに表示できます。

パラメータ

- **strMsg** : このパラメータは、ログに表示するメッセージを指定します。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

注



注意:

この関数はConnect-It更新スクリプトでのみ使用できます。

PifIgnoreSubDocumentReconc()

内部Basicシンタックス

Function PifIgnoreSubDocumentReconc(strMsg As String) As Long

説明

この関数は、フィールド型要素が選択されている場合に適用されます。この関数を照合更新操作で使用すると、関数の適用先と同じレベルにあるすべての要素と

すべてのサブ要素（リンク、構造体、コレクションなど）が無視されます。挿入や更新は実行されません。**strMsg**パラメータで指定したメッセージをログに表示できます。

パラメータ

- **strMsg** : このパラメータは、ログに表示するメッセージを指定します。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

注

注意:

この関数は、フィールドが選択されている際、Connect-It照合更新スクリプトでのみ使用できます（つまり、リンク、構造体、コレクション型要素には適用されません）。

与えられたドキュメント型のフォーマットが以下の場合、

```
Struct1
-Field1
-Field1b
-Struct2
C-Field2
```

フィールド1b、構造体2、およびフィールド2は無視されます。

PiflsInMap()

内部Basicシンタックス

Function PiflsInMap(strKey As String, strMappable As String, bCaseSensitive As Long) As Long

説明

この関数は、キーワードがマップテーブル中に存在するかどうかをテストします。検索では大文字と小文字を区別することもできます。

パラメータ

- **strKey** : キーワード。
 - **strMappable** : 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ない。
 - **1** : 大文字と小文字を区別する。
- bCaseSensitive** : このパラメータは、検索で大文字と小文字を区別するかどうかを指定します。

戻りコード

- **0** : キーワードが見つからない。
- **1** : キーワードが見つかった。

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 パラメータで指定された情報メッセージをログに出力します。

パラメータ

- **strMsg** : ログに出力するメッセージを表す文字列。

戻りコード

- 0 : 正常実行。
- 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 パラメータで指定された警告メッセージをログに出力します。

パラメータ

- **strMsg** : ログに出力する警告メッセージを表す文字列。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

例

```
If [Brand] = "" Then
PifLogWarningMsg("Unknown brand")
Else
RetVal = [Brand]
End if
```

PifMapValue()

内部Basicシンタックス

Function PifMapValue(strKey As String, strMappable As String, iPos As Long, strDefault As String, bCaseSensitive As Long, bLogErrIfOutOfRange As Long, bLogNewEntries As Long) As String

説明

マップテーブル中の要素の値を返します。要素は**strKey**、**strMapTable**、**iPos** パラメータによって一意に識別されます。
検索では大文字と小文字を区別することもできます。

パラメータ

- **strKey** : 要素を含むマップテーブルの行を識別するキーワード。
- **strMappable** : 検索するマップテーブルの名前。
- **iPos** : 値を取得したい要素を含む列の番号。

注意:

このパラメータは任意の正の数です。0が最初の列を表します。

- **strDefault** : 要素が見つからなかった場合に返されるデフォルト値。
- **bCaseSensitive** : このパラメータは、検索で大文字と小文字を区別するかどうかを指定します。
 - **0** : 大文字と小文字を区別しない。
 - **1** : 大文字と小文字を区別する（デフォルト値）。

- **bLogErrIfOutOfRange** : このパラメータは、列番号 (**iPos**パラメータ) が負であるか、マップテーブル中の列の数よりも大きい場合に、イベントログにエラーを記録する (=TRUE) かしない (=FALSE) かを指定します。



注意:

デフォルトでは、このパラメータはTRUEに設定されます。

- **bLogNewEntries** : このパラメータは、見つかったキーのうちまだマップテーブルで宣言されていないものに対するマップテーブルを、シナリオに隣接する新しいファイルに作成します。このファイルの名前は、シナリオの名前の前に `_NewMapKeys.mpt` を付けたものです。



注意:

シナリオに関連するマップテーブルは自動的に更新されません。マップテーブルを更新するには、作成されたファイルを編集し、新しい情報をコピーしてマップテーブルファイルに貼り付けます。

- **0** (デフォルト値) : マップテーブルファイルの自動作成を無効にします。
- **1** : マップテーブルファイルの自動作成を有効にします。

戻りコード

この関数は、見つかった文字列を返します。要素が見つからなかった場合は、デフォルト文字列 (**strDefault**パラメータの内容) を返します。

例

例えば、次のマップテーブルの場合 :



次のスクリプト :

```
pifMapValue("PRINTER", "TypeCategory", 1, "")
```

は**CAT_PRINTER**を返します。

次のスクリプト :

```
pifMapValue("Printer", "TypeCategory", 1, "")
```

はデフォルト値を返します。検索では大文字と小文字が区別されるので、検索された要素は見つかりません。

次のスクリプト：

```
pifMapValueContaining("Print", "TypeCategory", 1, 1, 1 "")
```

はデフォルト値を返し、イベントログにエラーを保存し、マップテーブルファイルを作成します。

注



注意:

この関数は、次のワイルドカード文字を正しく処理します。

- ? : 任意の文字を置き換えます。
- * : 任意の数の文字を置き換えます。

PifMapValueContaining()

内部Basicシンタックス

```
Function PifMapValueContaining(strKey As String, strMappable As String, iPos As Long, strDefault As String, bCaseSensitive As Long, bLogErrIfOutOfRange As Long, bLogNewEntries As Long) As String
```

説明

この関数は、マップテーブル中の要素の値を返します。要素は**strKey**、**strMappable**、**iPos**パラメータによって一意に識別されます。

検索では大文字と小文字を区別することもできます。

PifMapValue関数との違いは、**strKey**パラメータが検索するキーワードのサブセットでもかまわないことです。

例えば、**strKey**の値が"Monitor"の場合、キーワード"Cat_Monitor"を持つ要素が見つかります。

パラメータ

- **strKey** : 要素を含むマップテーブルの行を識別するキーワードまたはキーワードのサブセット。
- **strMappable** : 検索するマップテーブルの名前。

- **iPos** : 値を取得したい要素を含む列の番号。



注意:

このパラメータは任意の正の数です。0が最初の列を表します。

- **strDefault** : 要素が見つからなかった場合に返されるデフォルト値。
 - **0** : 大文字と小文字を区別しない。
 - **1** : 大文字と小文字を区別する (デフォルト値)。
- **bCaseSensitive** : このパラメータは、検索で大文字と小文字を区別するかどうかを指定します。
- **bLogErrIfOutOfRange** : このパラメータは、列番号 (**iPos**パラメータ) が負であるか、マップテーブル中の列の数よりも大きい場合に、イベントログにエラーを記録する (=TRUE) かしない (=FALSE) かを指定します。



注意:

デフォルトでは、このパラメータはTRUEに設定されます。

- **bLogNewEntries** : このパラメータは、見つかったキーのうちまだマップテーブルで宣言されていないものに対するマップテーブルを、シナリオに隣接する新しいファイルに作成します。このファイルの名前は、シナリオの名前の前に `_NewMapKeys.mpt` を付けたものです。



注意:

シナリオに関連するマップテーブルは自動的に更新されません。マップテーブルを更新するには、作成されたファイルを編集し、新しい情報をコピーしてマップテーブルファイルに貼り付けます。

- **0** (デフォルト値) : マップテーブルファイルの自動作成を無効にします。
- **1** : マップテーブルファイルの自動作成を有効にします。

戻りコード

この関数は、見つかった文字列を返します。要素が見つからなかった場合は、デフォルト文字列 (**strDefault**パラメータの内容) を返します。

例

例えば、次のマップテーブルの場合：



次のスクリプト：

```
pifMapValueContaining("PRINT", "TypeCategory", 1, "")
```

は**CAT_PRINTER**を返します。



注意:

StrKeyパラメータに一致するキーがマップテーブルに複数ある場合、最初に見つかったキーが返されます。

次のスクリプト：

```
pifMapValueContaining("Print", "TypeCategory", 1, "")
```

はデフォルト値を返します。検索では大文字と小文字が区別されるので、要素は見つかりません。

次のスクリプト：

```
pifMapValueContaining("Print", "TypeCategory", 1, 1, 1 "")
```

はデフォルト値を返し、イベントログにエラーを保存し、マップテーブルファイルを作成します。

注



注意:

この関数は、次のワイルドカード文字を正しく処理します。

- ? : 任意の文字を置き換えます。
- * : 任意の数の文字を置き換えます。

PifMapValueContainingEx()

内部Basicシンタックス

Function PifMapValueContainingEx(strKey As String, strMappable As String, strColName As String, strDefault As String, bCaseSensitive As Long, bLogErrIfOutOfRange As Long, bLogNewEntries As Long) As String

説明

この関数は、マップテーブル中の要素の値を返します。要素はパラメータ **strKey**、**strMapTable**、**strColName**によって一意に識別されます。

要素の検索では、大文字と小文字を区別することもできます。

PifMapContaining関数と異なり、**strColName**パラメータは列の位置でなく名前を定義します。

この関数は次の場合に使用できます。

- マップテーブルがフォーマットから作成された場合（列名がフォーマットの要素名に対応）。
- 列名がマップテーブルエディタを使ってマップテーブル中に手で定義された場合。

パラメータ

- **strKey** : マップテーブルで定義されたキーの1つを含む式。
- **strMappable** : 検索するマップテーブルの名前。
- **strColName** : 取得したい値を持つ要素を含む列の名前。
- **strDefault** : 要素が見つからなかった場合に返されるデフォルト値。
 - **0** : 検索で大文字と小文字を区別しない。
 - **1** (デフォルト値) : 検索で大文字と小文字を区別する。
- **bCaseSensitive** : このパラメータは、検索で大文字と小文字を区別するかどうかを指定します。
- **bLogErrIfOutOfRange** : このパラメータは、列番号 (**iPos**パラメータ) が負であるか、マップテーブル中の列の数よりも大きい場合に、イベントログにエラーを記録する (=TRUE) かしない (=FALSE) かを指定します。

 **注意:**

デフォルトでは、このパラメータはTRUEに設定されます。

- **blogNewEntries** : このパラメータは、見つかったキーのうちまだマップテーブルで宣言されていないものに対するマップテーブルを、シナリオに隣接する新しいファイルに作成します。このファイルの名前は、シナリオの名前の前に **_NewMapKeys.mpt** を付けたものです。

 **注意:**

シナリオに関連するマップテーブルは自動的に更新されません。マップテーブルを更新するには、作成されたファイルを編集し、新しい情報をコピーしてマップテーブルファイルに貼り付けます。

- **0** (デフォルト値) : マップテーブルファイルの自動作成を無効にします。
- **1** : マップテーブルファイルの自動作成を有効にします。

戻りコード

この関数は、見つかった文字列を返します。要素が見つからなかった場合は、デフォルト文字列 (**strDefault** パラメータの内容) を返します。

注

 **注意:**

この関数は、次のワイルドカード文字を正しく処理します。

- **?** : 任意の文字に対応します。
- ***** : 任意の数の文字に対応します。

フォーマットから作成されたマップテーブルまたはマップテーブルエディタを使って手動で作成されたマップテーブルがない場合は、**PifMapViewContaining** または **PifMapView** 関数を使用します。

PifMapValueEx()

内部Basicシンタックス

Function PifMapValueEx(strKey As String, strMappable As String, strColName As String, strDefault As String, bCaseSensitive As Long, bLogErrIfOutOfRange As Long, bLogNewEntries As Long) As String

説明

この関数は、マップテーブル中の要素の値を返します。要素はパラメータ**strKey**、**strMapTable**、**strColName**によって一意に識別されます。**PIFMAPVALUE**関数との違いは、**strColName**パラメータが列の番号でなく名前を指定することです。

要素の検索では、大文字と小文字を区別することもできます。

この関数は次の場合に使用できます。

- マップテーブルがフォーマットから作成された場合（列名がフォーマットの要素名に対応）。
- 列名がマップテーブルエディタを使ってマップテーブル中に手動で定義された場合。

パラメータ

- **strKey** : 要素を含むマップテーブルの列を識別するキーワード。
- **strMappable** : 検索するマップテーブルの名前。
- **strColName** : 取得したい値を持つ要素を含む列の名前。
- **strDefault** : 要素が見つからなかった場合に返されるデフォルト値。
- **bCaseSensitive** : このパラメータは、検索で大文字と小文字を区別するかどうかを指定します。
 - **0** : 検索で大文字と小文字を区別しない。
 - **1** (デフォルト値) : 検索で大文字と小文字を区別する。
- **bLogErrIfOutOfRange** : このパラメータは、列番号 (**iPos**パラメータ) が負であるか、マップテーブル中の列の数よりも大きい場合に、イベントログにエラーを記録する (=TRUE) かしない (=FALSE) かを指定します。



注意:

デフォルトでは、このパラメータはTRUEに設定されます。

- **bLogNewEntries** : このパラメータは、見つかったキーのうちまだマップテーブルで宣言されていないものに対するマップテーブルを、シナリオに隣接する新しいファイルに作成します。このファイルの名前は、シナリオの名前の前に `_NewMapKeys.mpt` を付けたものです。



注意:

シナリオに関連するマップテーブルは自動的に更新されません。マップテーブルを更新するには、作成されたファイルを編集し、新しい情報をコピーしてマップテーブルファイルに貼り付けます。

- **0** (デフォルト値) : マップテーブルファイルの自動作成を無効にします。
- **1** : マップテーブルファイルの自動作成を有効にします。

戻りコード

この関数は、見つかった文字列を返します。要素が見つからなかった場合は、デフォルト文字列 (**strDefault** パラメータの内容) を返します。

注



注意:

この関数は、次のワイルドカード文字を正しく処理します。

- `?` : 任意の文字に対応します。
- `*` : 任意の数の文字に対応します。

フォーマットから作成されたマップテーブルまたはマップテーブルエディタを使って手動で作成されたマップテーブルがない場合は、**PifMapViewContaining** または **PifMapView** 関数を使用します。

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

この関数は、リソースによって生成されるドキュメントのリストにあらかじめ定義されたドキュメントタイプに対するクエリを作成します。

パラメータ

- **strCntrName** : このパラメータは、リソース（クエリの実行対象）の名前を指定します。
- **strFmtName** : このパラメータは、ドキュメントタイプの識別子（生成されるドキュメントタイプとしてあらかじめ定義されたもの）を指定します。
- **strLayer** : このパラメータは、生成されるドキュメントタイプに対する生成ディレクティブを定義するために使用します。
生成ディレクティブは次のいずれかです。
 - コネクタに対する正しい構文を使用した**WHERE**節
 - コネクタに対して適用される生成ディレクティブ（**Order By**節など）を定義するXML記述。生成ディレクティブのXML構文は、**PifNewQueryFromXML**のドキュメントに記述されています。



注意:

ドキュメントタイプ**strFmtName**に生成ディレクティブがすでに含まれる場合、そのディレクティブは**strLayer**パラメータに指定されたものとマージされます。2つのパラメータで同じディレクティブに対して異なる値が指定された場合、**strLayer**で定義された値が用いられます。

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

例

次の例は、**strLayer**パラメータに使用する単純なWhere節を示します。

```
dim hQuery as long
dim iRc as long

hQuery = pifNewQueryFromFmtName("Asset Management", "amEmplDept",
, "Name like '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)
```

注

関連項目：

- PifNewQueryFromXml() [献 134]
- PifQueryClose() [献 139]
- PifQueryNext() [献 145]
- PifQueryGetDateVal() [献 140]
- PifQueryGetDoubleVal() [献 141]
- PifQueryGetIntVal() [献 142]
- PifQueryGetLongVal() [献 143]
- PifQueryGetStringVal() [献 144]

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

この関数は、リソースに対するクエリを作成します。ドキュメントタイプは **strQuery** パラメータによってXMLで完全に定義されている必要があります。処理（AQLクエリ節）は **strLayer** パラメータによってXMLで定義されます。

パラメータ

- **strCntrName**：このパラメータは、リソース（クエリの実行対象）の名前を指定します。

- **strQuery** : このパラメータは、クエリを実行するドキュメントタイプ（属性、構造、コレクション）を定義するXMLドキュメントを指定します。
- **strLayer** : このパラメータは、クエリに対する生成ディレクティブ（Where節、Order By節など）を定義するXMLドキュメントを指定します。

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

例

```
strLayer = "<Directives>"
strLayer = strLayer + "<Where>Name = " + GetXmlElementValue([Name])
+ "</Where>"
strLayer = strLayer + "<OrderBy>BarCode</OrderBy>"
strLayer = strLayer + "<Where Path='ItemsUsed'>AssetTag like 'A%'</Where>"
strLayer = strLayer + "</Directives>"
```

注

この例では、生成するドキュメントと実行するクエリの節の両方を指定するXMLドキュメントを作成します。すべてのXMLドキュメントと同様、これは有効でなければなりません。予約された文字（<, &など）はGetXmlElementValue() [献 71]およびGetXmlAttributeValue() [献 71]関数を使ってエスケープする必要があります。

注意:

strLayerが単純なAQL Where節を指定する場合、GetXmlElementValue() [献 71]を使用する必要はありません。

上の説明では、生成するドキュメントのタイプがわかっており、クエリに単純なWHERE節が含まれると仮定しています。下の例は、生成するドキュメントのタイプがわからない場合にこの関数を使用する方法を示します。これはまた、他のAQL節の使用方法も示します。

```
dim hQuery as long
dim strQuery as string
dim strLayer as string
dim iRc as long

strQuery = "<STRUCTURE Name='amEmplDept'>"
```

```

strQuery = strQuery + "<ATTRIBUTE Name='Name'/>"
strQuery = strQuery + "<ATTRIBUTE Name='BarCode'/>"
strQuery = strQuery + "<COLLECTION Name='ItemsUsed'>"
strQuery = strQuery + "<ATTRIBUTE Name='AssetTag'/>"
strQuery = strQuery + "</COLLECTION>"
strQuery = strQuery + "</STRUCTURE>"

strLayer = "<Directives>"
strLayer = strLayer + "<Where>Name = 'Taltekt'</Where>"
strLayer = strLayer + "<OrderBy>BarCode</OrderBy>"
strLayer = strLayer + "<Where Path='ItemsUsed'>AssetTag like 'A%'</Where>"
strLayer = strLayer + "</Directives>"

hQuery = pifNewQueryFromXml("Asset Management", strQuery, strLayer)

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)

```

注意:

この例では、生成するドキュメントと実行するクエリの節の両方を指定するXMLドキュメントを作成します。すべてのXMLドキュメントと同様、これは有効でなければなりません。予約された文字（<、&など）は、対応するエンティティ（<、&など）に置き換える必要があります。

関連項目：

- PifQueryClose() [献 139]
- PifNewQueryFromFmtName() [献 132]
- PifQueryNext() [献 145]
- PifQueryGetDateVal() [献 140]
- PifQueryGetDoubleVal() [献 141]
- PifQueryGetIntVal() [献 142]
- PifQueryGetLongVal() [献 143]
- PifQueryGetStringVal() [献 144]

PifNodeExists()

内部Basicシンタックス

Function PifNodeExists(strPath As String) As Long

説明

フルアクセスパスで識別されるノードが、生成されたドキュメント中に存在するかどうかをテストします。

パラメータ

- **strPath** : 操作の対象とな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** : ノードが存在しない場合
- **1** : ノードが存在する場合

例

```
If PifNodeExists("Hardware.Peripherals.Printer") = 0 Then  
PifIgnoreNodeMapping  
End If
```

PifOpenODBCDatabase()

内部Basicシンタックス

Function PifOpenODBCDatabase(strDSN As String, strLogin As String, strPwd As String) As Long

説明

この関数はODBCデータベースとの接続をオープンします。

パラメータ

- **strDSN** : 接続するODBCデータソースの名前。
- **strLogin** : ODBCデータベースへの接続に使用するログイン名。
- **strPwd** : ODBCデータベースへの接続に使用するログイン名 (**strLogin**) に対応するパスワード。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

例

```
Dim iRet As Integer
iRet = PifOpenOdbcDatabase(PifStrVal("DSN"), PifStrVal("LOGIN"), PifStrVal("PWD"))

if iRet = 0 Then
Dim strQuery As String
strQuery = "SELECT a.AssetName, a.FullName FROM Asset a"
iRet = PifCreateDynaMapTable(PifStrVal("DSN"), "Assets", strQuery, FALSE)
End If

Dim strRes as String
if iRet = 0 Then
strRes = PifMapValue([FullName], "Assets", 1, "", 1)
End If

If iRet = 0 and strRes = "" Then
iRet = PifExecODBCSql(PifStrVal("DSN"), "INSERT INTO Asset(AssetName, AssetFullName)
```

注

注意:

データソースとの接続がすでに存在する場合、この関数は別の接続を作成しません。

PifQueryClose()

内部Basicシンタックス

Function PifQueryClose(IQueryHandle As Long) As Long

説明

この関数は、クエリをクローズし、クエリが使用している内部リソースを解放します。

パラメータ

- **IQueryHandle** : このパラメータは、**PifNewQueryFromFmtName()**または**PifNewQueryFromXml()**関数で作成されたクエリのハンドルを指定します。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

例

```
dim hQuery as long
dim iRc as long

hQuery = pifNewQueryFromFmtName("Asset Management", "amEmplDept", "Name like '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)
```

注

関連項目：

- PifNewQueryFromXml() [献 134]
- PifNewQueryFromFmtName() [献 132]
- PifQueryNext() [献 145]
- PifQueryGetDateVal() [献 140]
- PifQueryGetDoubleVal() [献 141]
- PifQueryGetIntVal() [献 142]
- PifQueryGetLongVal() [献 143]
- PifQueryGetStringVal() [献 144]

PifQueryGetDateVal()

内部Basicシンタックス

Function PifQueryGetDateVal(IQueryHandle As Long, strPath As String, bPathMustExist As Long) As Date

説明

この関数は、現在のドキュメントのノードの値を（Dateで）返します。現在のドキュメントとは、**PifQueryNext()**関数でクエリのカーソルが設定されているものです。

パラメータ

- **IQueryHandle**：このパラメータは、**PifNewQueryFromFmtName()**または**PifNewQueryFromXml()**関数で作成されたクエリのハンドルを指定します。
- **strPath**：このパラメータは、値を取得したい現在のドキュメントのノードのパスを指定します。
- **bPathMustExist**：このパラメータは、**strPath**パラメータで指定されたパスから値を読み取れなかった場合に、イベントログにエラーを記録する（=TRUE）かしない（=FALSE）かを指定します。

戻りコード

指定されたノードの値

注

関連項目：

- `PifNewQueryFromXml()` [献 134]
- `PifNewQueryFromFmtName()` [献 132]
- `PifQueryNext()` [献 145]
- `PifQueryClose()` [献 139]
- `PifQueryGetDoubleVal()` [献 141]
- `PifQueryGetIntVal()` [献 142]
- `PifQueryGetLongVal()` [献 143]
- `PifQueryGetStringVal()` [献 144]

PifQueryGetDoubleVal()

内部Basicシンタックス

Function PifQueryGetDoubleVal(IQueryHandle As Long, strPath As String, bPathMustExist As Long) As Double

説明

この関数は、現在のドキュメントのノードの値を（Doubleで）返します。現在のドキュメントとは、**PifQueryNext()**関数でクエリのカーソルが設定されているものです。

パラメータ

- **IQueryHandle**：このパラメータは、**PifNewQueryFromFmtName()**または**PifNewQueryFromXml()**関数で作成されたクエリのハンドルを指定します。
- **strPath**：このパラメータは、値を取得したい現在のドキュメントのノードのパスを指定します。
- **bPathMustExist**：このパラメータは、**strPath**パラメータで指定されたパスから値を読み取れなかった場合に、イベントログにエラーを記録する（=TRUE）かしない（=FALSE）かを指定します。

戻りコード

指定されたノードの値

注

関連項目：

- `PifNewQueryFromXml()` [献 134]
- `PifNewQueryFromFmtName()` [献 132]
- `PifQueryNext()` [献 145]
- `PifQueryGetDateVal()` [献 140]
- `PifQueryClose()` [献 139]
- `PifQueryGetIntVal()` [献 142]
- `PifQueryGetLongVal()` [献 143]
- `PifQueryGetStringVal()` [献 144]

`PifQueryGetIntVal()`

内部Basicシンタックス

Function `PifQueryGetIntVal(IQueryHandle As Long, strPath As String, bPathMustExist As Long) As Long`

説明

この関数は、現在のドキュメントのノードの値を（Integerで）返します。現在のドキュメントとは、**`PifQueryNext()`**関数でクエリのカーソルが設定されているものです。

パラメータ

- **`IQueryHandle`**：このパラメータは、**`PifNewQueryFromFmtName()`**または**`PifNewQueryFromXml()`**関数で作成されたクエリのハンドルを指定します。
- **`strPath`**：このパラメータは、値を取得したい現在のドキュメントのノードのパスを指定します。
- **`bPathMustExist`**：このパラメータは、**`strPath`**パラメータで指定されたパスから値を読み取れなかった場合に、イベントログにエラーを記録する（=TRUE）かしない（=FALSE）かを指定します。

戻りコード

指定されたノードの値

注

関連項目：

- `PifNewQueryFromXml()` [献 134]
- `PifNewQueryFromFmtName()` [献 132]
- `PifQueryNext()` [献 145]
- `PifQueryGetDateVal()` [献 140]
- `PifQueryGetDoubleVal()` [献 141]
- `PifQueryClose()` [献 139]
- `PifQueryGetLongVal()` [献 143]
- `PifQueryGetStringVal()` [献 144]

PifQueryGetLongVal()

内部Basicシンタックス

Function PifQueryGetLongVal(IQueryHandle As Long, strPath As String, bPathMustExist As Long) As Long

説明

この関数は、現在のドキュメントのノードの値を（Longで）返します。現在のドキュメントとは、**PifQueryNext()**関数でクエリのカーソルが設定されているものです。

パラメータ

- **IQueryHandle**：このパラメータは、**PifNewQueryFromFmtName()**または**PifNewQueryFromXml()**関数で作成されたクエリのハンドルを指定します。
- **strPath**：このパラメータは、値を取得したい現在のドキュメントのノードのパスを指定します。
- **bPathMustExist**：このパラメータは、**strPath**パラメータで指定されたパスから値を読み取れなかった場合に、イベントログにエラーを記録する（=TRUE）かしない（=FALSE）かを指定します。

戻りコード

指定されたノードの値

例

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

注

関連項目：

- PifNewQueryFromXml() [献 134]
- PifNewQueryFromFmtName() [献 132]
- PifQueryNext() [献 145]
- PifQueryGetDateVal() [献 140]
- PifQueryGetDoubleVal() [献 141]
- PifQueryGetIntVal() [献 142]
- PifQueryClose() [献 139]
- PifQueryGetStringVal() [献 144]

PifQueryGetStringVal()

内部Basicシンタックス

Function PifQueryGetStringVal(IQueryHandle As Long, strPath As String, bPathMustExist As Long) As String

説明

この関数は、現在のドキュメントのノードの値を（Stringで）返します。現在のドキュメントとは、**PifQueryNext()**関数でクエリのカーソルが設定されているものです。

パラメータ

- **IQueryHandle**：このパラメータは、**PifNewQueryFromFmtName()**または**PifNewQueryFromXml()**関数で作成されたクエリのハンドルを指定します。

- **strPath** : このパラメータは、値を取得したい現在のドキュメントのノードのパスを指定します。
- **bPathMustExist** : このパラメータは、**strPath**パラメータで指定されたパスから値を読み取れなかった場合に、イベントログにエラーを記録する (=TRUE) かしない (=FALSE) かを指定します。

戻りコード

指定されたノードの値

注

関連項目 :

- PifNewQueryFromXml() [献 134]
- PifNewQueryFromFmtName() [献 132]
- PifQueryNext() [献 145]
- PifQueryGetDateVal() [献 140]
- PifQueryGetDoubleVal() [献 141]
- PifQueryGetIntVal() [献 142]
- PifQueryGetLongVal() [献 143]
- PifQueryClose() [献 139]

PifQueryNext()

内部Basicシンタックス

Function PifQueryNext(IQueryHandle As Long) As Long

説明

この関数は、次の結果にクエリカーソルを設定します。カーソルは、**PifNewQueryFromFmtName()**または**PifNewQueryFromXml()**関数を呼び出した後で最初のドキュメントに設定されません。次のいずれかの関数で現在のドキュメントの値にアクセスするには、ユーザが**PifQueryNext()**関数を呼び出す必要があります。

- **PifQueryGetStringVal()**
- **PifQueryGetDateVal()**
- **PifQueryGetDoubleVal()**

- **PifQueryGetLongVal()**
- **PifQueryGetIntVal()**

パラメータ

- **IQueryHandle** : このパラメータは、**PifNewQueryFromFmtName()**または**PifNewQueryFromXml()**関数で作成されたクエリのハンドルを指定します。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

例

```
dim hQuery as long
dim iRc as long

hQuery = pifNewQueryFromFmtName("Asset Management", "amEmplDept"
, "Name like '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)
```

注

この関数は、参照するデータが残っていない場合にはエラーを返します（エラーコード-2003）。

関連項目：

- [PifNewQueryFromXml\(\)](#) [献 134]
- [PifNewQueryFromFmtName\(\)](#) [献 132]

- PifQueryClose() [献 139]
- PifQueryGetDateVal() [献 140]
- PifQueryGetDoubleVal() [献 141]
- PifQueryGetIntVal() [献 142]
- PifQueryGetLongVal() [献 143]
- PifQueryGetStringVal() [献 144]

PifRejectCollectionMapping()

内部Basicシンタックス

Function PifRejectCollectionMapping(strMsg As String) As Long

説明

この関数は、コレクションからコレクションへのマッピングでのみ、コレクションのすべての要素を拒否するために使用します。

なお、**PifRejectNodeMapping()** 関数を使えば、単にコレクションの現在のノードを拒否することができます。

strMsgパラメータで指定したメッセージをログファイルに出力できます。

注意:

この関数を使用することは、ドキュメントの部分的拒否と同等です。部分的拒否は、マッピングボックスのプロセスレポートで読み取ることができます。

パラメータ

- **strMsg** : このオプションのパラメータは、ログファイルに書き込むメッセージを指定します。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

例

次の例では、要素の1つで`quantity`ノードの値が'-1'に設定されている場合、コレクションのすべての要素が拒否されます。

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

PifRejectNodeMapping()との比較として、次の例では、コレクションの要素のうち、`quantity`ノードの値が'-1'に設定されているものだけが拒否されます。

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

注

関連項目：

- `PifRejectNodeMapping()` [献 150]
- `PifRejectDocumentMapping()` [献 148]

PifRejectDocumentMapping()

内部Basicシンタックス

Function PifRejectDocumentMapping(strMsg As String) As Long

説明

この関数は、ドキュメントを拒否するために使用します。

strMsgパラメータで指定した情報メッセージをログに出力できます。

ドキュメントは次のコネクタに送信されません。

パラメータ

- **strMsg**：オプションのパラメータ。ログに出力される文字列。

戻りコード

- 0：正常実行。

- 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
```

PifRejectDocumentReconc()

内部Basicシンタックス

Function PifRejectDocumentReconc(strMsg As String) As Long

説明

この関数は、照合更新操作でドキュメントの拒否に使用されます。挿入や更新は実行されません。**strMsg**パラメータに格納されているメッセージをログに表示できます。

 **注意:**

この関数は非パラレル化モードでのみ使用できます。

パラメータ

- **strMsg** : このパラメータは、ログに表示するメッセージを指定します。

戻りコード

- 0 : 正常実行。
- 0以外：エラーコード。

注

注意:

この関数はConnect-It更新スクリプトでのみ使用できます。

関連項目：

- PifRejectNodeReconc() [献 151]
- PifRejectSubDocumentReconc() [献 152]

PifRejectNodeMapping()

内部Basicシンタックス

Function PifRejectNodeMapping(strMsg As String) As Long

説明

この関数は、ドキュメントの現在のノードを拒否するために使用します。

strMsg パラメータで指定した情報メッセージをログに出力できます。

注意:

この関数を使用することは、ドキュメントの部分的拒否と同等です。部分的拒否は、マッピングボックスのプロセスレポートで読み取ることができます。

パラメータ

- **strMsg** : オプションのパラメータ。ログに出力される文字列。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

例

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

注



警告:

この関数はドキュメントのルートノードに対しては使用できません。ドキュメント全体を拒否するには、 **PifRejectDocumentMapping** 関数を使用します。

PifRejectNodeReconc()

内部Basicシンタックス

Function PifRejectNodeReconc(strMsg As String) As Long

説明

この関数は、照合更新操作で現在のノード（サブドキュメント、コレクション、構造体など）の拒否に使用されます。挿入や更新は実行されません。**strMsg**パラメータに格納されているメッセージをログに表示できます。

パラメータ

- **strMsg** : このパラメータは、ログに表示するメッセージを指定します。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

注



注意:

この関数はConnect-It更新スクリプトでのみ使用できます。

PifRejectSubDocumentReconc()

内部Basicシンタックス

Function PifRejectSubDocumentReconc(strMsg As String) As Long

説明

この関数は、照合更新操作でサブドキュメントの拒否に使用されます。挿入や更新は実行されません。**strMsg**パラメータに格納されているメッセージをログに表示できます。

パラメータ

- **strMsg** : このパラメータは、ログに表示するメッセージを指定します。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

注



注意:

この関数は更新スクリプトでのみ使用できます。

PifScenarioPath()

内部Basicシンタックス

Function PifScenarioPath() As String

説明

この関数は、シナリオファイルのフルパスを返します。シナリオがまだ保存されていない場合、この関数は空値を返します。

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

例

```
Dim strFile
strFile = pifScenarioPath & "Format.xml"

Open strFile For Output As #1
Print #1, pifXmlDump
Close #1
```

PifSetDateVal()

内部Basicシンタックス

Function PifSetDateVal(strPath As String, dtfVal As Date) As Long

説明

この関数は、ターゲットドキュメントのノードの値を設定するために使用します。ノードが存在しない場合、この関数はノードを作成します。

パラメータ

- **strPath** : このパラメータは、操作の対象となるノードのフルパスを指定します。
- **dtVal** : このパラメータは、ノードに割り当てる値 (Date) を指定します。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

例

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

注

関連項目 :

- [PifSetDoubleVal\(\)](#) [献 154]
- [PifSetLongVal\(\)](#) [献 155]
- [PifSetNullVal\(\)](#) [献 156]
- [PifSetStringVal\(\)](#) [献 158]

PifSetDoubleVal()

内部Basicシンタックス

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

説明

この関数は、ターゲットドキュメントのノードの値を設定するために使用します。ノードが存在しない場合、この関数はノードを作成します。

パラメータ

- **strPath** : このパラメータは、操作の対象となるノードのフルパスを指定します。
- **dVal** : このパラメータは、ノードに割り当てる値 (Double) を指定します。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

例

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

注

関連項目 :

- PifSetDateVal() [献 153]
- PifSetLongVal() [献 155]
- PifSetNullVal() [献 156]
- PifSetStringVal() [献 158]

PifSetLongVal()

内部Basicシンタックス

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

説明

この関数は、ターゲットドキュメントのノードの値を設定するために使用します。ノードが存在しない場合、この関数はノードを作成します。

パラメータ

- **strPath** : このパラメータは、操作の対象となるノードのフルパスを指定します。
- **lVal** : このパラメータは、ノードに割り当てる値 (Long) を指定します。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

例

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

注

関連項目 :

- [PifSetDoubleVal\(\)](#) [献 154]
- [PifSetDateVal\(\)](#) [献 153]
- [PifSetNullVal\(\)](#) [献 156]
- [PifSetStringVal\(\)](#) [献 158]

PifSetNullVal()

内部Basicシンタックス

Function PifSetNullVal(strPath As String) As Long

説明

この関数は、ターゲットドキュメントのノードに値'NULL'を設定するために使用します。ノードが存在しない場合、この関数はノードを作成します。

パラメータ

- **strPath** : このパラメータは、操作の対象となるノードのフルネームを指定します。このパラメータが省略されているか空の場合、現在のノードが選択されます。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

例

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

注

関連項目 :

- PifSetDoubleVal() [献 154]
- PifSetLongVal() [献 155]
- PifSetDateVal() [献 153]
- PifSetStringVal() [献 158]

PifSetPendingDocument()

内部Basicシンタックス

Function PifSetPendingDocument(strMsg As String) As Long

説明

この関数を使用して、照合更新操作中、ドキュメントを保留します。ドキュメントは挿入も更新もされません。**strMsg**パラメータに格納されているメッセージをログに表示できます。



注意:

この関数は非パラレル化モードでのみ使用できます。

パラメータ

- **strMsg** : このパラメータは、ログに表示するメッセージを指定します。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

例

```
If vNewVal >= vOldVal Then
RetVal = vNewVal
Else
RetVal = vOld
PifSetPendingDocument(PifStrVal("RECONC_PROPOSAL_NOT_VALIDATED"))
End If
```

注



注意:

この関数は更新スクリプトでのみ使用できます。

PifSetStringVal()

内部Basicシンタックス

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

説明

この関数は、ターゲットドキュメントのノードの値を設定するために使用します。ノードが存在しない場合、この関数はノードを作成します。

パラメータ

- **strPath** : このパラメータは、操作の対象となるノードのフルパスを指定します。
- **strVal** : このパラメータは、ノードに割り当てる値（文字列）を指定します。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

例

```
Dim str as String
Dim lRet as Long
str = "100.10.1.1"
lRet = PifSetStringVal("ipaddress", str)
```

注意:

戻りコードの変数への代入は必須です（そうしないとコンパイルエラーが発生します）。

ここで、**c**はフィールド、**b**がコレクションで、**index**は対象の番号です。

amComputer.LogicalDrivesコレクションのNameフィールドの一例を以下に挙げます。

```
PifSetStringVal(amComputer.LogicalDrives(3).Name)
```

この構文では、コレクションにある3番目の論理ディスクドライブ（LogicalDrive(3)）の名前が設定されます。

注

関連項目 :

- PifSetDoubleVal() [献 154]
- PifSetLongVal() [献 155]
- PifSetNullVal() [献 156]

- PifSetDateVal() [献 153]

PifStrVal()

内部Basicシンタックス

Function PifStrVal(strID As String) As String

説明

strIDパラメータで指定された識別子に対応する文字列を返します。

パラメータ

- **strID** : 読み取る文字列の識別子。

戻りコード

識別子が見つからない場合、この関数は空文字列を返し、Connect-Itログにエラーを書き込みます。識別子が見つかった場合、この関数は対応す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

この関数は、Connect-Itのグラフィカルインタフェースのウィザードであらかじめ定義されたフォーマットに従って文字列を処理し、定義されたフォーマットの性質に応じて日付または数値型の値を返します。

パラメータ

- **strData** : このパラメータは、処理する文字列を指定します。
- **strUserFmtName** : このパラメータは、定義済みフォーマットの名前を指定します。

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

例

次の2つの定義済みフォーマットがあるとします。

- `origin = "yyyy'-mm'-dd"`
- `destination = "dddd' dd' mmmm' yyyy"`

この場合、次のスクリプト：

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

は、[date_modified]= 2001-05-30に対して値"Thursday, May 30, 2001"を返します。

注

 **注意:**

定義済みフォーマットの詳細については、『Connect-It User's Guide』を参照してください。

関連項目：

- `PifUserFmtVarToStr()` [献 162]

PifUserFmtVarToStr()

内部Basicシンタックス

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

説明

この関数は、定義済みフォーマットに従ってVariantを処理し、文字列を返します。

パラメータ

- **vData** : このパラメータは、関数が処理するVariantを指定します。
- **strUserFmtName** : このパラメータは、定義済みフォーマットの名前を指定します。

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

例

次の2つの定義済みフォーマットがあるとします。

- `origin = "yyyy'-'mm'-'dd"`
- `destination = "dddd' 'dd' 'mmmm' 'yyyy"`

この場合、次のスクリプト：

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

は、[date_modified]= 2001-05-30に対して値"Thursday, May 30, 2001"を返します。

注



注意:

定義済みフォーマットの詳細については、『Connect-It User's Guide』を参照してください。

関連項目：

- `PifUserFmtStrToVar()` [献 160]

PifWriteBlobInFile()

内部Basicシンタックス

**Function PifWriteBlobInFile(strPath As String, strFileName As String)
As Long**

説明

この関数は、バイナリ要素（BLOB）をファイルに書き込むために使用します。ファイルが存在しない場合は作成されます。ファイルがすでに存在する場合は上書きされます。

パラメータ

- **strPath**：このパラメータは、バイナリ要素（BLOB）のソースドキュメント中でのフルパスを指定します。
- **strFileName**：このパラメータは、バイナリオブジェクトが保存されるファイルのフルパスを指定します。

戻りコード

- 0：正常実行。
- 0以外：エラーコード。

例

下の画面キャプチャは、社員の名前と写真を記録したソースドキュメントを示します。



次のスクリプトは、各社員の写真をc:\bitmapフォルダに保存します。ファイル名は社員の名前です。

```
Dim lErr As Long
Dim strFileName As String

strFileName = "C:\bitmap\" & ['contact.name'] & ".bmp"
lErr = PifWriteBlobInFile("portrait.attachment", strFileName)
```

注意:

- **contact.name**要素にはドット (".") が含まれます。このため、これを参照するには引用符で囲む必要があります (['contact.name'])。
- 関数がエラーコードを返す場合、関数の戻りコードを必ず変数に代入する必要があります (例: **lErr = PifWriteBlobInFile()**)。そうしないと、スクリプトは無効になります。

注

この関数は次の場合にエラーを返します。

- ファイルのパスが有効でない場合
- ソース要素のパスがソースドキュメント中に存在しない場合
- ソース要素がバイナリ要素でない場合

PifXMLDump()

内部Basicシンタックス

Function PifXMLDump(strPath As String) As String

説明

この関数は、ソースドキュメントの要素（およびそのすべてのサブ要素）の内容をXMLフォーマットで文字列にダンプするために使用します。

パラメータ

- **strPath** : このパラメータは、操作の対象となるソースドキュメントの要素のフルパスを指定します。

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

例

最初の例は、下の要素の内容を**PifXMLDump()**関数で保存する方法を示します。



"field1"フィールドに関連付けられた次のスクリプトを使用します。

```
RetVal = PifXMLDump("")
```

下の画面キャプチャは、スクリプト実行の結果を示します。



"field1"には次のXMLドキュメントが格納されます。

```
<?xml version="1.0" encoding="ISO-8859-1"?>  
<Machine>  
<FileInfo>  
<FileName>F:\ConnectIt4016\datakit\idd\Frankfurt01.fsf</FileName>  
</FileInfo>  
</Machine>
```

FileInfo要素の構造だけを保存するには、次のスクリプトを使用します。

```
RetVal = PifXMLDump("FileInfo")
```

この場合、フィールドに格納されるXMLドキュメントは次のようになります。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<FileInfo>
<FileName>F:\ConnectIt4016\datakit\idd\Frankfurt01.fsf</FileName>
</FileInfo>
```

PMT()

内部Basicシンタックス

Function PMT(dblRate As Double, iNper As Long, dbIPV As Double, dbIFV 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
- **dbIPV** : 実際に支払わなければならない金額 (総額)
- **dbIFV** : 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、Connect-Itログにメッセージが書き込まれます。

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, dbIPV As Double, dbIFV 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
- **dbIPV** : 実際に支払わなければならない金額 (総額)
- **dbIFV** : 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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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(IValue As Long)

説明

乱数発生関数を初期化します。

パラメータ

- **IValue** : 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、Connect-Itログにメッセージが書き込まれます。

例

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

注

次も参照してください:

- [Rnd\(\)](#) [献 178]

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

例

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

注

次も参照してください:

- [SubList\(\)](#) [献 189]
- [SetSubList\(\)](#) [献 182]
- [ApplyNewVals\(\)](#) [献 43]

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。デフォルトではこのパラメータは1に設定されています。

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

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。
```

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**: 戻される文字数。

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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。デフォルトではこのパラメータは1に設定されています。

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

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。

RmAllInDir()

内部Basicシンタックス

Function RmAllInDir(strRmDirectory As String, bStopIfError As Long) As Long

説明

この関数は、フォルダ内の全アイテム(ファイルとフォルダ)を削除します。フォルダ自体は削除しません。

パラメータ

- **strRmDirectory**: このパラメータには、操作に関係するフォルダのフルパスが入ります。
- **bStopIfError**: このパラメータが1に設定されている場合、ファイルやフォルダの削除が行えない際に削除操作を中断します。0に設定されている場合、操作は継続され次のファイルやフォルダに移ります。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

例

```
RetVal = RmAllInDir("c:\files\test", 1)
```

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 : 正常実行。
- 0以外 : エラーコード。

例

```
RetVal = RmDir("c: mp")
```

注

 **注意:**

削除するディレクトリは空である必要があります。空でない場合、関数は実行されません

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、Connect-Itログにメッセージが書き込まれます。

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、**Randomize**関数をパラメータなしで使用し、乱数ジェネレータを初期化する必要があります。

次も参照してください:

- [Randomize\(\)](#) [献 169]

RoundValue()

内部Basicシンタックス

Function RoundValue(dValue As Double, iDigits As Long) As Double

説明

この関数は、**iDigits**パラメータで指定した小数点以下の桁数で、数値の丸め値を計算します。

パラメータ

- **dValue**: このパラメータには、丸め演算の対象となる数値が入ります。
- **iDigits**: このパラメータには、丸め演算を行う小数点位置の数値が入ります。

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

例

次の例:

```
RetVal = RoundValue(1.2568, 2)
```

は、次の値を戻します:

```
1.26
```

次の例:

```
RetVal = RoundValue(1.2568, 0)
```

は、次の値を戻します:

```
1
```

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、Connect-Itログにメッセージが書き込まれます。

例

```
' 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()

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、Connect-Itログにメッセージが書き込まれます。

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。
```

SetMaxInst()

内部Basicシンタックス

Function SetMaxInst(lMaxInst As Long) As Long

説明

Basicスクリプトが実行できる命令の最大数を設定します。デフォルトでは命令の数は10000個に制限されています。

パラメータ

- **IMaxInst** : スクリプトが実行できる命令の最大数。

戻りコード

- 0 : 正常実行。
- 0以外 : エラーコード。

注



注意:

IMaxInstパラメータを「0」に設定すると、スクリプトが実行できる命令の数は無制限になります。

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、Connect-Itログにメッセージが書き込まれます。

例

```
Dim MyStr
MyStr=SetSubList("a1 | a2 | a3=a0,b1 | b2 | b3=b0,c1 | c2 | c3=c0", "A2 | A1=a0, B
2 | B1=b0", "2 | 1") : "A1 | A2 | a3=a0,B1 | B2 | b3=b0,c1 | c2 | c3=c0"を返します。
MyStr=SetSubList("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=SetSubList("a1 | a2 | a3=a0,b1 | b2 | b3=b0,c1 | c2 | c3=c0", "B5 | B6 | B7=b
0,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=SetSubList("a1 | a2 | a3=a0,b1 | b2 | b3=b0,c1 | c2 | c3=c0", "B1 | B2 | B3 | B
4=b0", "-") : "a1 | a2 | a3=a0,B1 | B2 | B3 | B4=b0,c1 | c2 | c3=c0"を返します。
MyStr=SetSubList("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, bShowWindow As Long, bBackground As Long) 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
- **bShowWindow** : パラメータが1に設定されている場合(デフォルト値)、プログラム起動時にコマンドボックスが表示されます。0に設定されている場合、コマンドボックスは表示されません。
- **bBackground** : このパラメータが1に設定されている場合(デフォルト値)、関数はプログラムの実行完了を待ってからユーザに制御を返します(同期実行)。0に設定されている場合、プログラムは非同期に実行されます。

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

例

```
Dim MyId
MyId=Shell("C:\WinNT\explorer.exe")
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、Connect-Itログにメッセージが書き込まれます。

例

```
Dim dCalc as Double
dCalc=Sin(2.79)
RetVal=dCalc
```

注

 **注意:**

度からラジアンへの変換公式は次になります:

```
ラジアンでの角度 = (度での角度) * Pi / 180
```

Space()

内部Basicシンタックス

Function Space(iCount 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

- **iCount**: 文字列に挿入する空白文字の個数。

戻りコード

エラーの場合、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

例

```
Dim dNumber as Double  
dNumber=Cos(2.79)  
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 : strString**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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

例

```
Dim dCalc as Double  
dCalc=Tan(2.79)  
RetVal=dCalc
```

注

 **注意:**

度からラジアンへの変換公式は次になります:

```
ラジアンでの角度 = (度での角度) * Pi / 180
```

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、Connect-Itログにメッセージが書き込まれます。

例

```
RetVal = Time()
```

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、Connect-Itログにメッセージが書き込まれます。

例

```
RetVal = Timer()
```

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

例

次の例:

```
RetVal = ToSmart ("hello world")
```

は次の値を戻します:

```
Hello World
```

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、Connect-Itログにメッセージが書き込まれます。

例

```
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()

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、Connect-Itログにメッセージが書き込まれます。

例

```
' 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()

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

例

```
RetVal = UTCToLocaldate([DateTime])
```

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、Connect-Itログにメッセージが書き込まれます。

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、Connect-Itログにメッセージが書き込まれます。

例

```
Dim strYear  
strYear=Year(Date())  
RetVal=strYear : '現在の年を返します。
```

III 索引

使用可能な関数 - 全関数のリスト

- **Abs**
- **AppendOperand**
- **ApplyNewVals**
- **Asc**
- **Atn**
- **BasicToLocalDate**
- **BasicToLocalTime**
- **BasicToLocalTimeStamp**
- **Beep**
- **CDbl**
- **ChDir**
- **ChDrive**
- **Chr**
- **Clnt**
- **CLng**
- **Cos**
- **CountOccurrences**
- **CountValues**
- **CSng**
- **CStr**
- **CurDir**
- **CVar**
- **Date**
- **DateAdd**
- **DateAddLogical**
- **DateDiff**
- **DateSerial**
- **DateValue**
- **Day**
- **EscapeSeparators**
- **ExeDir**
- **Exp**
- **ExtractValue**
- **FileCopy**
- **FileDateTime**
- **FileExists**
- **FileLen**
- **Fix**
- **FormatDate**
- **FormatResString**

- **FV**
- **GetEnvVar**
- **GetListItem**
- **GetXmlAttributeValue**
- **GetXMLElementValue**
- **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**
- **PifCloseODBCDatabase**
- **PifCreateDynaMappableFromFmtName**
- **PifDateToTimezone**
- **PifExecODBCSql**
- **PifFirstInCol**
- **PifGetBlobSize**
- **PifGetDateVal**
- **PifGetDoubleVal**
- **PifGetElementChildName**
- **PifGetElementCount**
- **PifGetHexStringFromBlob**
- **PifGetInstance**
- **PifGetIntlStringVariantVal**
- **PifGetIntVal**
- **PifGetItemCount**
- **PifGetLongVal**
- **PifGetOpenSessionTime**
- **PifGetStringFromBlob**
- **PifGetStringVal**
- **PifGetVariantVal**
- **PifIgnoreCollectionMapping**
- **PifIgnoreDocumentMapping**
- **PifIgnoreDocumentReconc**
- **PifIgnoreNodeMapping**
- **PifIgnoreNodeReconc**
- **PifIgnoreSubDocumentReconc**
- **PifIsInMap**
- **PifLogInfoMsg**
- **PifLogWarningMsg**
- **PifMapValue**
- **PifMapValueContaining**
- **PifMapValueContainingEx**
- **PifMapValueEx**
- **PifNewQueryFromFmtName**

- **PifNewQueryFromXml**
- **PifNodeExists**
- **PifOpenODBCDatabase**
- **PifQueryClose**
- **PifQueryGetDateVal**
- **PifQueryGetDoubleVal**
- **PifQueryGetIntVal**
- **PifQueryGetLongVal**
- **PifQueryGetStringVal**
- **PifQueryNext**
- **PifRejectCollectionMapping**
- **PifRejectDocumentMapping**
- **PifRejectDocumentReconc**
- **PifRejectNodeMapping**
- **PifRejectNodeReconc**
- **PifRejectSubDocumentReconc**
- **PifScenarioPath**
- **PifSetDateVal**
- **PifSetDoubleVal**
- **PifSetLongVal**
- **PifSetNullVal**
- **PifSetPendingDocument**
- **PifSetStringValue**
- **PifStrVal**
- **PifUserFmtStrToVar**
- **PifUserFmtVarToStr**
- **PifWriteBlobInFile**
- **PifXMLDump**
- **PMT**
- **PPMT**
- **PV**
- **Randomize**
- **RATE**
- **RemoveRows**
- **Replace**
- **Right**
- **RightPart**
- **RightPartFromLeft**
- **RmAllInDir**
- **Rmdir**
- **Rnd**
- **RoundValue**
- **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**

使用可能な関数 - Connect-It

- **GetXmlAttributeValue**
- **GetXmlElementValue**
- **PifCloseODBCDatabase**
- **PifCreateDynaMappableFromFmtName**
- **PifDateToTimezone**
- **PifExecODBCSql**
- **PifFirstInCol**
- **PifGetBlobSize**
- **PifGetDateVal**
- **PifGetDoubleVal**
- **PifGetElementChildName**
- **PifGetElementCount**
- **PifGetHexStringFromBlob**
- **PifGetInstance**
- **PifGetIntlStringVariantVal**
- **PifGetIntVal**
- **PifGetItemCount**
- **PifGetLongVal**
- **PifGetOpenSessionTime**
- **PifGetStringFromBlob**
- **PifGetStringVal**
- **PifGetVariantVal**
- **PifIgnoreCollectionMapping**
- **PifIgnoreDocumentMapping**
- **PifIgnoreDocumentReconc**
- **PifIgnoreNodeMapping**
- **PifIgnoreNodeReconc**
- **PifIgnoreSubDocumentReconc**
- **PifIsInMap**
- **PifLogInfoMsg**
- **PifLogWarningMsg**
- **PifMapValue**
- **PifMapValueContaining**
- **PifMapValueContainingEx**
- **PifMapValueEx**
- **PifNewQueryFromFmtName**
- **PifNewQueryFromXml**
- **PifNodeExists**
- **PifOpenODBCDatabase**
- **PifQueryClose**

- **PifQueryGetDateVal**
- **PifQueryGetDoubleVal**
- **PifQueryGetIntVal**
- **PifQueryGetLongVal**
- **PifQueryGetStringVal**
- **PifQueryNext**
- **PifRejectCollectionMapping**
- **PifRejectDocumentMapping**
- **PifRejectDocumentReconc**
- **PifRejectNodeMapping**
- **PifRejectNodeReconc**
- **PifRejectSubDocumentReconc**
- **PifScenarioPath**
- **PifSetDateVal**
- **PifSetDoubleVal**
- **PifSetLongVal**
- **PifSetNullVal**
- **PifSetPendingDocument**
- **PifSetStringVal**
- **PifStrVal**
- **PifUserFmtStrToVar**
- **PifUserFmtVarToStr**
- **PifWriteBlobInFile**
- **PifXMLDump**

使用可能な関数 - 組み込み

- **Abs**
- **AppendOperand**
- **ApplyNewVals**
- **Asc**
- **Atn**
- **BasicToLocalDate**
- **BasicToLocalTime**
- **BasicToLocalTimeStamp**
- **Beep**
- **CDbl**
- **ChDir**
- **ChDrive**
- **Chr**
- **Clnt**
- **CLng**
- **Cos**
- **CountOccurrences**
- **CountValues**
- **CSng**
- **CStr**
- **CurDir**
- **CVar**
- **Date**
- **DateAdd**
- **DateAddLogical**
- **DateDiff**
- **DateSerial**
- **DateValue**
- **Day**
- **EscapeSeparators**
- **ExeDir**
- **Exp**
- **ExtractValue**
- **FileCopy**
- **FileDateTime**
- **FileExists**
- **FileLen**
- **Fix**
- **FormatDate**
- **FormatResString**

- **FV**
- **GetEnvVar**
- **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**
- **RmAllInDir**
- **Rmdir**
- **Rnd**
- **RoundValue**
- **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**

