

CA Client Automation

Desktop Management スクリプト言語

12.9



このドキュメント（組み込みヘルプシステムおよび電子的に配布される資料を含む、以下「本ドキュメント」）は、お客様への情報提供のみを目的としたもので、日本 CA 株式会社（以下「CA」）により随時、変更または撤回されることがあります。本ドキュメントは、CA が知的財産権を有する機密情報であり、CA の事前の書面による承諾を受けずに本書の全部または一部を複写、譲渡、変更、開示、修正、複製することはできません。

本ドキュメントで言及されている CA ソフトウェア製品のライセンスを受けたユーザは、社内でユーザおよび従業員が使用する場合に限り、当該ソフトウェアに関連する本ドキュメントのコピーを妥当な部数だけ作成できます。ただし、CA のすべての著作権表示およびその説明を当該複製に添付することを条件とします。

本ドキュメントを印刷するまたはコピーを作成する上記の権利は、当該ソフトウェアのライセンスが完全に有効となっている期間内に限定されます。いかなる理由であれ、上記のライセンスが終了した場合には、お客様は本ドキュメントの全部または一部と、それらを複製したコピーのすべてを破棄したことを、CA に文書で証明する責任を負います。

準拠法により認められる限り、CA は本ドキュメントを現状有姿のまま提供し、商品性、特定の使用目的に対する適合性、他者の権利に対して侵害のないことについて、黙示の保証も含めいかなる保証もしません。また、本ドキュメントの使用に起因して、逸失利益、投資損失、業務の中断、営業権の喪失、情報の喪失等、いかなる損害（直接損害か間接損害かを問いません）が発生しても、CA はお客様または第三者に対し責任を負いません。CA がかかる損害の発生の可能性について事前に明示に通告されていた場合も同様とします。

本ドキュメントで参照されているすべてのソフトウェア製品の使用には、該当するライセンス契約が適用され、当該ライセンス契約はこの通知の条件によっていかなる変更も行われません。

本書の制作者は CA および CA Inc. です。

「制限された権利」のもとでの提供：アメリカ合衆国政府が使用、複製、開示する場合は、FAR Sections 12.212、52.227-14 及び 52.227-19(c)(1)及び(2)、ならびに DFARS Section 252.227-7014(b)(3) または、これらの後継の条項に規定される該当する制限に従うものとします。

Copyright © 2014 CA. All rights reserved. 本書に記載されたすべての商標、商号、サービス・マークおよびロゴは、それぞれの各社に帰属します。

CA Technologies 製品リファレンス

このマニュアルセットで参照されている CA 製品は、以下のとおりです。

- CA Advantage® Data Transport® (CA Data Transport)
- CA Asset Intelligence
- CA APM (CA Asset Portfolio Management)
- CA Business Intelligence
- CA Common Services™
- CA DMM (CA Desktop Migration Manager)
- CA Embedded Entitlements Manager (CA EEM)
- CA MDM (CA Mobile Device Management)
- CA NSM (CA Network and Systems Management)
- CA Patch Manager
- CA Process Automation
- CA Service Desk Manager
- CA WorldView™

CA への連絡先

テクニカルサポートの詳細については、弊社テクニカルサポートの Web サイト (<http://www.ca.com/jp/support/>) をご覧ください。

目次

第 1 章：Desktop Management スクリプト言語	7
識別子.....	7
数値.....	7
コメント.....	8
エラー コード（Desktop Management スクリプト言語）	8
変数の宣言	11
ラベル.....	13
配列の宣言.....	13
式.....	14
特殊文字.....	15
ステートメント.....	16
代入文.....	16
条件文.....	17
関数呼び出し文.....	19
GOTO ステートメント	19
反復文.....	19
関数.....	22
共通設定関数.....	22
Intellisigs 関数.....	26
Intellisig に対する追加情報	46
表示関数.....	52
環境関数.....	60
ファイルの内容に関する関数.....	64
ファイル システム関数.....	82
アイコン関数.....	135
初期化ファイル（.INI）関数	143
MIF および Inv ファイル関数.....	154
その他の関数.....	170
ネットワーク関数.....	187
レジストリ操作関数.....	190
文字列操作関数.....	222
ユーザ定義関数.....	235
例：スクリプト関数からの DLL の使用	236

第 1 章 : Desktop Management スクリプト言語

このスクリプト言語は、エージェントでコマンドを実行するための共通手段です。このスクリプト言語は、エージェント コマンドを実行する **Desktop Management** コンポーネントの機能（たとえば、**Asset Management** エージェントでソフトウェアを管理できる機能）を備えたプログラミング ツールです。スクリプト コマンドを熟知すると、すべてのエージェント プラットフォームおよびオペレーティング システムで使用できます。

スクリプトのコア言語には、通常の制御文、式、および変数が含まれます。コア言語には、項目プロシージャで必要な一般的なタスクおよび特定のタスクを実行できる関数のライブラリがあります。

言語機能を使用することにより、さまざまなターゲット オペレーティング システム間で移植可能なスクリプトを作成できます。

識別子

識別子は、プログラムで使用する定数、変数、および関数です。識別子には、最大 **64** 文字を格納でき、先頭は文字またはアンダースコアである必要があります。識別子には、スペースおよびかっこを格納できません。識別子では、大文字小文字は区別されません。

例 :

```
MyVar  
MyString
```

数値

DMS では、32 ビットの整数が使用されます。数値は **-2147483647** から **2147483648** までの範囲になります。

コメント

コメントは、**REM** または一重引用符 (') で開始される行です。 **REM** または一重引用符を行の途中に挿入でき、その場合、行の残り部分は無視されます。

例：コメント

REM この行はコメントです。

'この行はコメントです。

Dim x as Integer ' この部分はコメントです

エラー コード (Desktop Management スクリプト言語)

DMS 言語エラー コードを以下に示します。

エラー コード	説明
1	構文エラーです。
2	無効な変数のインデックス タイプ
3	不明な変数タイプ
4	メモリ不足
5	変数名の重複
6	予想される変数名
7	不明な変数
8	不明な関数
9	変数が必要です
10	変数が初期化されていません
11	タイプ名の前に使用できるもの
12	不明な変数タイプ名
13	無効な変数名
14	スクリプト ファイルを開くことができません

エラーコード	説明
15	ラベル名の重複
16	IF .. THEN ブロックに THEN がありません
17	順番に沿った式が必要です
18	IF .. THEN ブロックに END IF がありません
19	WHILE ...END WHILE ブロック内に END WHILE がありません
20	REPEAT...UNTIL ブロック内に UNTIL がありません
21	FOR .. NEXT ブロックに TO がありません
22	FOR .. NEXT ブロックに NEXT がありません
23	関数定義の構文エラー
24	関数定義に END FUNCTION がありません
25	不明な EXIT 文タイプ
26	式内のデータ タイプの無効な組み合わせ
27	使用可能な整数または文字列式
28	無効な式の演算
29	データ タイプに互換性がありません
30	このデータ タイプには無効な演算です
31	ゼロで分割
32	基本タイプが必要です
33	必要な式
34	使用可能な変数参照
35	関数呼び出しの引数が一致しません
36	無効なルート変更
37	使用可能なプリプロセッサ変数名
38	プリプロセッサ式の構文エラー
39	プリプロセッサ構文エラー
40	対応する #IF がない #ELSE が見つかりました
41	対応する #IF がない #ELSEIF が見つかりました

エラーコード	説明
42	対応する #IF がない #ENDIF が見つかりました
43	無効なプリプロセッサ命令
44	不明なプリプロセッサ命令
45	無効な関数形式
46	ライブラリ ファイルをロードできません
47	関数を参照できません
48	使用可能な文字または文字列式
49	使用可能な直接の値
50	定数を評価できません
51	引数に無効な変数タイプがあります
52	終了していないテキスト スtring 定数
53	使用可能な文字定数
54	終了していない文字定数
55	式の構文エラー
56	スクリプト デバッグ セッションのユーザ ブレイク
57	スクリプト デバッグ セッションで終了したプログラム
58	無効な GOTO 先
59	使用可能なスクリプト記号
60	変数を評価できません
68	Exit リストをロードできません
69	クラスを登録できません
70	DLL をロードできません
73	無効な配列タイプ
74	無効な構造メンバタイプ
75	配列インデックスが範囲外です
76	配列サイズ フィールドがありません

エラー コード	説明
77	警告：関数「<関数名>」の宣言が複数検出されました。繰り返しは無視されます。
78	スクリプト ファイルがありません
79	ロケールに問題が検出されました
80	インタープリタはオペレータによって取り消されました
81	警告：マスクされた引用符 (¥") がスクリプトで検出されました。簡易引用符 (") が想定されます。
82	メモリが不足しています。DMS インタープリタは異常終了しました。
83	内部エラー
84	スクリプトが 64 K 行を超えて構成されています
85	インクルード ファイルが見つかりません

変数の宣言

変数を使用する前にスクリプトでその変数を宣言し、変数に値を割り当てる前にその変数を宣言します。

スクリプト変数を宣言するには、**DIM** キーワードを使用します。

変数の基本タイプおよびデフォルト値は以下のとおりです。

ブール値

unsigned char (False)

整数

long (0)

文字列

char * ("")

文字列

wchar_t (0)

バイト

unsigned char (0)

ワード

unsigned short (0)

短整数型

short (0)

例：

```
DIM MyIntegerVar as Integer
DIM MyStringVar as String
MyIntegerVar = 1234
MyStringVar = "This is a string."
```

```
DIM C1, C2, C3 as Char
C1 = 0x61
C2 = 0x0061
C3 = "a"
rem C1, C2, C3 - すべて同じ文字です
```

ラベル

ラベルは、以下の規則に準拠する識別子です。

- ラベルの開始や終了はコロン「:」で行います。
- 文字、数字、またはアンダースコアのみを使用します。
- GOTO 文内で使用して、プログラム内のあるポイントからラベル プレフィクスが付けられた文へプログラムの実行を移動します。

注：ラベルでは大文字小文字は区別されません。

例：

```
Dim Number As Integer
Number = 1
IF Number=1 Then GOTO end
PRINT "Number=0"
end:
```

配列の宣言

配列は、データ要素を順序よく配置したものです。 スクリプト インタープリタでは、1 次元配列（ベクトル）のみ処理できます。

例：

12 要素のベクトル。ここでは、最初の要素 **a[0]** には、値（数値）**444** が格納されています。

```
Dim ar[12] as integer
0 - 11
a[0] = 444
```

注：配列は基本タイプでのみ宣言することができますが、配列を使用してユーザ タイプを宣言することができます。

正しい例

```
DIM a[25] as Integer
DIM b[100] as Boolean
```

間違った例

```
TYPE T1
  A as Integer
  B as Char
END TYPE
```

```
DIM C[25] as T1
```

式

式は、演算子およびオペランドから構成されます。

演算子には、AND OR * / + - < > <= >= があります。

演算子に適用される優先順位の規則は、代数の規則に従います。つまり、すべての演算子には 2 つのオペランドが使用されます。

注： NOT 演算子はありませんが、事前定義された関数 NOT を使用できます。

オペランドが文字列型である場合、オペランドは + 演算子により連結されます。

```
DIM FirstName as String
DIM LastName as String
DIM FullName as String
DIM Space as String
FirstName="Mary"
LastName="Johnson"
Space=" "
FullName=FirstName+Space+LastName
```

式 `FirstName+Space+LastName` は「Mary Johnson」になり、したがって、文字列 `FullName` の内容は「Mary Johnson」になります。

+ 演算子を整数型のオペランドに使用する場合、スクリプト インタープリタでは、以下のようにオペランドが加算されます。

```
DIM X as Integer
DIM Y as Integer
DIM Z as Integer
X=2
Y=3
Z=X+Y
```

変数 **Z** には、整数値 5 が格納されます。

式の結果型は、オペランドの最大の型になります。

Byte+Word = Word

Char+String = String

特殊文字

C 言語などの一部のプログラミング言語では、円記号（「¥」）を使用して特殊文字を区別します。 **Desktop Management** スクリプトでは、円記号は、エスケープ記号としてサポートされませんが、通常の文字としてサポートされ、パス名およびファイル処理に使用できます。 特殊文字を入力するには、その文字の **16** 進値をコード化します。

たとえば、**Windows** 環境で文字列に改行を追加するには、スクリプトに以下の行を追加します。

```
Dim LF, CR As Char
Dim NL, str As String
LF = 0x0A
CR = 0x0D
NL = CR + LF
.
.
.
str = str + CRLF;
```

UNIX 環境では、CR は廃止されており、さらに、前の例における NL 初期化が以下のように置き換えられます。

NL = LF

頻繁に使用されるコードを以下に示します。

タブレーター (HT) : 0x09
改行 (LF) : 0x0A
キャリッジ リターン (CR) : 0x0D

Desktop Management スクリプトでは、NEWLINE\$ という名前の改行の定数が用意されています。インタプリタでは、正しい初期化および実行されている環境により決定されるこの定数の値が確認されます。定数は、スクリプト内でのみ使用します。上記の例は、以下のようになります。

```
Dim str As String
.
.
.
str = str + NEWLINE$
```

ステートメント

文は、スクリプトで実行するアクションを記述するものです。

代入文

代入文は、最も基本的なスクリプト言語文です。

例：

```
DIM x as Integer
DIM y as Integer
DIM z as Integer
x=1
y=x+z
```

条件文

スクリプト言語には、条件文の 1 タイプ - IF..THEN..ELSE 文があります。この文の構文は、以下のとおりです。

```
IF <expression> THEN
    <statement block>
[ELSE
    <statement block>]
ENDIF
```

文全体が 1 行に収まる場合は、ENDIF なしの条件文を使用できます。

```
IF <expression> THEN <statement block>
```

式の結果は、整数型である必要があります。

- 式が TRUE（ゼロ以外の値）である場合、THEN に続く文が実行されます。
- 式が FALSE（ゼロ値）であり、ELSE 部分がある場合、ELSE に続く文が実行されます。
- ELSE 部分がない場合、プログラムの実行は ENDIF から継続されます。

注：ENDIF の代わりに End If を使用できます。

条件文の別の形式として、ネストされた条件文があります。以下に示す最初のコード例のようなネストされた条件文の代わりに、2 番目の例のような条件文を作成できます。

```
DIM os As String
.
.
.
If os = "AIX" Then
    aixRun()
Else
    if os = "HPUX" Then
        rc = hpRun()
    Else
        if os = "SOLARIS(sparc)" Then
            solarisRun()
        else
            print("unknown os type: " + os)
        End If
    End If
End If
.
.
.
```

2 番目の例：

```
DIM os As String
.
.
.
If os = "AIX" Then
    aixRun()
Elseif os = "HPUX" Then
    hpRun()
Elseif os = "SOLARIS(sparc)" Then
    solarisRun()
else
    print("unknown os type: " + os)
End If
.
.
.
```

例：

```
DIM ProgName as String
DIM ProgPath as String
ProgPath="C:\AVIRUS"
IF InStr(ossystem,"DOS") THEN
ProgName=ProgPath + "FINDVIRU.EXE"
ELSE
ProgName=ProgPath + "WFINDVIR.EXE"
ENDIF
end:
```

関数呼び出し文

関数呼び出しを文として使用でき、この場合、関数の結果は無視されます。

例：

```
Print("message: Starting section 2")
```

GOTO ステートメント

GOTO 文を使用して、プログラム内のあるポイントから別のポイントへプログラムの実行制御を移動できます。**GOTO** 文の後ろに、処理が継続される部分のラベル名が続く必要があります。

GOTO 文を使用して、ループ（**FOR**、**WHILE**、**REPEAT**）、関数ブロック、または **IF** (**ELSE**) ブロックの外側から内側にジャンプすることはできません。

例：

```
GOTO MyLabel
```

反復文

反復文では、条件が真の間、または条件が真になるまで、アクションが繰り返されます。

FOR..NEXT..STEP 文

このタイプの文の構文は、以下のとおりです。

```
FOR variable=expression TO expression  
    [STEP expression]  
    文ブロック  
NEXT variable
```

ここで、*expression* は整数に評価される必要があります。

注：FOR..NEXT..STEP 文中に EXIT FOR を使用して、FOR ループを終了することができます。

例：

この文では、2、4、6、および8の各数が各行に出力され、次に、4つすべての数が1行に出力されます。

```
DIM A as String  
DIM J as Integer  
A=""  
FOR J=2 TO 8 STEP 2  
A=A+Str(J)  
PRINT Str(J)  
NEXT J  
PRINT A  
end:
```

WHILE..WEND 文

WHILE..WEND 文には、文ブロックの反復実行を制御する式が含まれます。

式の結果は **Integer** タイプである必要があります。文ブロックは、式が真（ゼロ以外）と評価される間中、繰り返し実行されます。初回に式が偽（ゼロ）と評価された場合、文ブロックは実行されません。

文の構文は以下のとおりです。

```
WHILE <expression>  
<statement block>  
WEND
```

EXIT WHILE により、WHILE..WEND ループは終了します。

例：

```
DIM FileName as String
DIM NoOfLines as Integer
DIM TmpLine as string
DIM hFile as Integer
FileName="C:¥AUTOEXEC.BAT"
NoOfLines=0
hFile=OpenFile(FileName,0,0)
WHILE NOT(Eof(hFile))
    ReadFile(hFile,TmpLine)
    NoOfLines=NoOfLines+1
WEND
CloseFile(hFile)
NoOfLines=NoOfLines-1
PRINT FileName + " Contains " + str(NoOfLines) + " lines."
end:
```

REPEAT..UNTIL 文

REPEAT..UNTIL 文には、文ブロックが含まれます。1つの式により、1つの式の反復実行が制御されます。

式の結果は **Integer** タイプである必要があります。文ブロックは、式が真（ゼロ以外）と評価されるまで、繰り返し実行されます。

文の構文は以下のとおりです。

```
REPEAT
  文ブロック
UNTIL expression
```

例：REPEAT ... UNTIL 文

```
DIM FileName as String
DIM NoOfLines as Integer
DIM TmpLine as string
DIM hFile as Integer
FileName="C:¥AUTOEXEC.BAT"
NoOfLines=0
hFile=OpenFile(FileName,0,0)
REPEAT
    ReadFile(hFile,TmpLine)
    NoOfLines=NoOfLines+1
UNTIL Eof(hFile)
CloseFile(hFile)
NoOfLines=NoOfLines-1
PRINT FileName + " Contains " + str(NoOfLines) + " lines."
end:
```

関数

Desktop Management スクリプト言語関数は、関連タイプ別にグループ化されています。

注： スクリプト言語では大文字小文字は区別されません。文字列の内容は、大文字と小文字が区別されます（例：文字列「ABCD」、「Abcd」、および「abcd」はそれぞれ異なります）。

共通設定関数

共通設定関数は **Windows** および **Unix** プラットフォームで実行できます。

注： これらの関数は、**CA Client Automation** で新しく使用されるようになったため、古いバージョンのスクリプトインタプリタでは動作しません。

CcnfGetParameterInt - 共通設定パラメータ整数値の読み取り

Windows および UNIX で有効です。

CcnfGetParameterInt 関数では、共通設定パラメータ整数値が読み取られます。

関数の形式

```
CcnfGetParameterInt(Parametername as string, Value as integer) as Boolean
```

Parametername

絶対パス（例："itrm/rc/protocols/tcp/connectiontimeout"）を含む共通設定パラメータの名前を定義します。

Value

Parametername の整数値を定義します。

この関数の戻り値はブール値です。この関数が正常に実行されると TRUE が返され、**Value** には **Parametername** の値が組み込まれます。この関数の実行が失敗すると、ログファイルにエラーが記録されます。

例：

この例では comstore パラメータを使用します。

```
Dim pNum as integer
if CcnfGetParameterInt("itrm/usb/shared/localtime deviation ", pNum) then
Print("itrm/usb/shared/localtime deviation = " + val(pNum) + " ")
else
Print("CcnfGetParameterInt failed")
endif
```

CcnfGetParameterStr - 共通設定パラメータ文字列値の読み取り

Windows および Unix で有効です。

CcnfGetParameterStr 関数では、共通設定パラメータ文字列値が読み取られます。

関数の形式

```
CcnfGetParameterStr(Parametername as string, Value as string) as boolean
```

Parametername

絶対パスを含む共通設定パラメータの名前
("itrm/rc/protocols/encrypt/dll" など)

Value

Parametername の文字列値。

この関数の戻り値はブール値です。この関数が正常に実行されると TRUE が返され、Value には Parametername の値が組み込まれます。この関数の実行が失敗すると、ログ ファイルにエラーが記録されます。

例 :

この例では comstore パラメータを使用します。

```
Dim pText as string
if CcnfGetParameterStr("itrm/usd/shared/nos", pText) then
Print("itrm/usd/shared/nos = "" + pText + """)
else
Print("CcnfGetParameterStr failed")
endif
```

CcnfSetParameterInt - 既存共通設定パラメータ (文字列型) の値変更

Windows および UNIX で有効です。

CcnfSetParameterInt 関数では、既存の共通設定パラメータ (整数型) の値が変更されます。

この共通設定パラメータ関数の形式は、以下のとおりです。

CcnfSetParameterInt(Parametername as string, Value as integer) as boolean

Parametername

絶対パスを含む共通設定パラメータの名前 (例 :
"itrm/rc/protocols/tcp/connectiontimeout")

Value

Parametername の文字列値

この関数の戻り値はブール値です。この関数が正常に実行されると、TRUE が返されます。この関数の実行が失敗すると、ログ ファイルにエラーが記録されます。

値が正しいかどうかは検証されません。

例：

この例では **comstore** パラメータを使用します。

```
if CcnfSetParameterInt("itrm/usd/shared/localtimedeviation ", 9) then
Print("value of itrm/usd/shared/localtimedeviation was modified" )
else
Print("CcnfSetParameterInt failed")
Endif
```

CcnfSetParameterStr - 既存共通設定パラメータ (文字列型) の値変更

Windows および Unix で有効です。

CcnfSetParameterStr 関数では、既存の共通設定パラメータ (文字列型) の値が変更されます。

この共通設定パラメータ関数の形式は、以下のとおりです。

```
CcnfSetParameterStr(Parametername as string, Value as string) as boolean
```

Parametername

絶対パスを含む共通設定パラメータの名前
("itrm/rc/protocols/encrypt/dll" など)

Value

Parametername の文字列値

この関数の戻り値はブール値です。この関数が正常に実行されると、TRUE が返されます。この関数の実行が失敗すると、ログ ファイルにエラーが記録されます。

値が正しいかどうかは検証されません。

例 : CcnfSetParameterStr 関数

この例では `comstore` パラメータを使用します。詳細については、「実装ガイド」の `asm.cnf` と `comstore.xml` 間のマッピングに関する章を参照してください。

```
if CcnfSetParameterStr("itrm/usd/shared/nos", "newValue") then
Print("value of itrm/usd/shared/nos was modified" )
else
Print("CcnfSetParameterStr failed")
endif
```

Intellisigs 関数

カスタム Intellisig を使用するときには、エージェント コンピュータ上で検出されるソフトウェア レコードをレポートするために Intellisig スクリプト内で `DMScript` 関数を呼び出します。`DMScript` は、検出されたソフトウェア レコードを出力ファイルに書き込む組み込み関数を提供します。

Intellisig スクリプトは、以下の順序で関数を呼び出す必要があります。

1. `OpenDetectedSoftwareOutputFiles`
2. `CreateDetectedSoftwareProduct`
3. `CreateDetectedSoftwareRelease`
4. `CreateDetectedSoftwarePatch`
5. `CreateDetectedSoftwareReleaseInstance`
6. `CloseDetectedSoftwareOutputFiles`

OpenDetectedSoftwareOutputFiles

OpenDetectedSoftwareOutputFiles 関数は、Intellisig 用に空のソフトウェア検出出力ファイルを作成します。ソフトウェア検出出力ファイルには、Intellisig に関して検出されたレコードが含まれます。Intellisig スクリプトがエージェント コンピュータで実行される場合、

OpenDetectedSoftwareOutputFiles 関数は、出力ファイルを作成して、Intellisig によって検出されたソフトウェアを格納します。たとえば、Intellisig にすべての Adobe 製品、リリース、およびパッチの定義が含まれる場合、すべての Adobe 検出レコードを同じ出力ファイルに書き込むことができます。

注： Intellisig スクリプトは、Intellisig がソフトウェアを検出するかどうかにかかわらず、**OpenDetectedSoftwareOutputFiles** 関数を呼び出す必要があります。出力ファイルを作成しないと、Intellisig の実行中にエラーが発生します。

関数の形式

OpenDetectedSoftwareOutputFiles (IntellisigUUID as String, Version as String, Name as String)as integer

例：

```
OpenDetectedSoftwareOutputFiles  
("A7C1E14A-7C93-4E17-B4E5-45B796717F49", "V1", "OS Detection for  
Windows")
```

入力パラメータ

この関数には、以下の入力パラメータがあります。

IntellisigUUID

Intellisig のユニバーサルユニーク識別子（UUID）を指定します。この関数は、指定された UUID をファイル名として、ソフトウェア検出力ファイルを作成します。

IntellisigVersion

Intellisig のバージョン番号を指定します。

IntellisigName

Intellisig の名前を指定します。

戻り値

CA_SWDETECT_OK

関数が操作を正常に完了したことを示します。

CASWDETECT_BADARGS

関数に渡された 1 つ以上の必須パラメータが空白あることを示します。

CASWDETECT_FILEERROR

関数が、ファイルを開くことができなかったことを示します。詳細については、ログファイルを参照してください。

CreateDetectedSoftwareProduct

CreateDetectedSoftwareProduct 関数は、スクリプトが Intellisig に関連する製品を検出すると、ソフトウェア検出出力ファイルにレコードを追加します。

関数の形式

CreateDetectedSoftwareProduct (ProductName as String, ProductVersion as String, OptionalProperties as String) は整数

例 :

```
CreateDetectedSoftwareProduct(("Microsoft Windows 7 Ultimate", "6.1",  
"VersionNumber=6.1 |Manufacturer=Microsoft  
Corporation|Category=Operating Systems |Description=The Microsoft  
Windows 7 Product")
```

注: この関数は、製品のリリースを作成する CreateDetectedSoftwareRelease 関数を呼び出す前に呼び出します。

入力パラメータ

この関数には、以下の入力パラメータがあります。

ProductName

検出されるソフトウェアの製品名を指定します。

注 : 製品名は Intellisig の内で一意である必要があります。 Intellisig 内で同じ名前を持つ 2 つの製品は、オプション パラメータにかかわらず同じ製品として扱われます。

ProductVersionLabel

製品のバージョン ラベルを指定します。

注 : バージョン ラベルは、製品とその名前の識別に使用されます。空の値が許可されており、空のバージョンを持つ製品として扱われます。

OptionalProperties

検出されたソフトウェア製品に関連付けられるオプションのプロパティを指定します。製品に対して、以下のオプションのプロパティを使用できます。

VersionNumber

製品のバージョン番号を指定します。

Language

製品の言語を指定します。

Bitness

製品が **32** ビットまたは **64** ビット アーキテクチャを使用するかどうかを指定します。

Architecture

製品が使用するアーキテクチャ名を指定します。

メーカー

製品の製造元の名前を指定します。

Category

製品カテゴリの名前を指定します。

ManufacturerUUID

製品の製造元の **UUID** を指定します。

CategoryUUID

製品カテゴリの **UUID** を指定します。

説明

製品の説明を指定します。

戻り値

CA_SWDETECT_OK

関数が操作を正常に完了したことを示します。

CASWDETECT_BADARGS

関数に渡された 1 つ以上の必須パラメータが空白あることを示します。

CASWDETECT_FILE_ERROR

関数が、ファイルへの書き込みに失敗したことを示します。詳細については、ログファイルを参照してください。

CreateDetectedSoftwareRelease

スクリプトがエージェント コンピュータ上で製品のリリースを検出すると、**CreateDetectedSoftwareRelease** 関数が、ソフトウェア検出出力ファイルにレコードを追加します。

注：DMScript は、MDB 内で関連する製品レコードを見つけた場合にのみレコードを追加します。

この関数の形式は、以下のとおりです。

CreateDetectedSoftwareRelease (ProductName as String, ProductVersionLabel, as String, ReleaseName as String, ReleaseVersionLabel as String, OptionalProperties as String) as an integer

例：

```
CreateDetectedSoftwareRelease ("Microsoft Windows 7 Ultimate", "6.1",  
"Microsoft Windows 7 Ultimate x64 64 en-us", "6.1.7600",  
"VersionNumber=6.1.7600 |Language=en-us |Bitness=64  
|Architecture=x64 |Manufacturer=Microsoft  
Corporation|Category=Operating Systems |Description=The Microsoft  
Windows 7 Release")
```

入力パラメータ

この関数には、以下の入力パラメータがあります。

ProductName

リリースが所属する製品名を指定します。スクリプト内で、この製品に対して **CreateDetectedSoftwareProduct** 関数を呼び出している必要があります。

ProductVersionLabel

製品のバージョン ラベルを指定します。

注：バージョン ラベルは、製品とその名前の識別に使用されます。空の値が許可されており、空のバージョンを持つ製品として扱われます。

ReleaseName

検出されるソフトウェア リリースを指定します。

注：リリースは、製品と同じ名前を持つことができます。ただし、Intellisig は、異なるリリース名をサポートしており、同じ製品に複数のリリースを関連付けることができます。リリース名とバージョン ラベルは、各 Intellisig チェーン内で一意である必要があります。Intellisig 内で同じ名前を持つ 2 つのリリースは、オプション パラメータにかかわらず、同じ親定義を共有する場合にのみ同じリリースとして扱われます。2 つの別々の Intellisig を使用して同じリリースが検出される場合、2 つの個別の検出レコードが Intellisig ごとに作成されます。

ReleaseVersionLabel

リリースのバージョン ラベルを指定します。

注：バージョン ラベルは、リリースとその名前の識別に使用されます。空の値が許可されており、空のバージョンを持つリリースとして扱われます。

OptionalProperties

検出されたソフトウェア リリースに関連付けられるオプションのプロパティを指定します。 リリースに対して、以下のオプションのプロパティを使用できます。

VersionNumber

リリースのバージョン番号を指定します。

Language

リリースがインストールされている言語を指定します。

Bitness

リリースが **32** ビットまたは **64** ビット アーキテクチャを使用するかどうかを指定します。

Architecture

アーキテクチャ名を指定します。

メーカー

リリースの製造元の名前を指定します。

Category

リリース カテゴリを指定します。

ManufacturerUUID

リリースの製造元の **UUID** を指定します。

CategoryUUID

リリース カテゴリの **UUID** を指定します。

説明

リリースの説明を指定します。

戻り値

CA_SWDETECT_OK

関数が操作を正常に完了したことを示します。

CASWDETECT_BADARGS

関数に渡された 1 つ以上の必須パラメータが空白あることを示します。

CASWDETECT_FILE_ERROR

関数が、ファイルへの書き込みに失敗したことを示します。詳細については、ログファイルを参照してください。

CreateDetectedSoftwarePatch

スクリプトが特定リリースのパッチを検出すると、**CreateDetectedSoftwarePatch** 関数が、ソフトウェア検出出力ファイルにレコードを追加します。Intellisig スクリプトに含めるすべてのパッチにこの機能を追加します。

関数の形式

CreateDetectedSoftwarePatch (ProductName as String, ProductVersionLabel, as String, ReleaseName as String, ReleaseVersionLabel as String, PatchName as String, PatchVersionLabel as String, OptionalProperties as String) as an integer

例：

```
CreateDetectedSoftwarePatch ("Microsoft Windows 7 Ultimate", "6.1",  
"Microsoft Windows 7 Ultimate x64 64 en-us", "6.1.7600", "KB971033 x64  
64 en-us", "Language=en-us |Bitness=64 |Architecture=x64  
|Manufacturer=Microsoft Corporation|Category=Operating Systems  
|Description=The Microsoft Windows 7 Activation Checker Update")
```

注：この関数は、各スクリプト内で **CreateDetectedSoftwareRelease** 関数を呼び出した後に呼び出します。

入力パラメータ

この関数には、以下の入力パラメータがあります。

ProductName

パッチが所属する製品の名前を指定します。スクリプト内で、この製品に対して **CreateDetectedSoftwareProduct** 関数を呼び出している必要があります。

ProductVersionLabel

製品のバージョン ラベルを指定します。

注：バージョン ラベルは、製品とその名前の識別に使用されます。空の値が許可されており、空のバージョンを持つ製品として扱われます。

ReleaseName

パッチが所属するリリースの名前を指定します。スクリプト内で、このリリースに対して **CreateDetectedSoftwareRelease** 関数を呼び出している必要があります。

ReleaseVersionLabel

リリースのバージョン ラベルを指定します。

注：バージョン ラベルは、リリースとその名前の識別に使用されます。空の値が許可されており、空のバージョンを持つリリースとして扱われます。

PatchName

検出されるソフトウェア パッチの名前を指定します。

注：パッチは、それが所属するリリースまたは製品と同じ名前にすることはできません。パッチ名とバージョン ラベルは、各 **Intellisig** ソフトウェア定義チェーン内で一意である必要があります。**Intellisig** 内で同じ名前を持つ 2 つのパッチは、オプション パラメータにかかわらず、同じ親定義を持つ場合にのみ同じパッチとして扱われます。2 つの別々の **Intellisig** を使用して同じパッチが検出される場合、2 つの個別の検出レコードが **Intellisig** ごとに作成されます。

PatchVersionLabel

パッチのバージョン ラベルを指定します。

注：バージョン ラベルは、パッチとその名前の識別に使用されます。空の値が許可されており、空のバージョンを持つパッチとして扱われます。

OptionalProperties

検出されたソフトウェア パッチに関連付けられるオプションのプロパティを指定します。パッチに対して、以下のオプションのプロパティを使用できます。

VersionNumber

パッチのバージョン番号を指定します。

Language

パッチがインストールされている言語を指定します。

Bitness

パッチが 32 ビットまたは 64 ビット アーキテクチャを使用するかどうかを指定します。

Architecture

パッチが使用するアーキテクチャ名を指定します。

メーカー

パッチ製造元名を指定します。

Category

パッチ カテゴリを指定します。

ManufacturerUUID

パッチ製造元の UUID を指定します。

CategoryUUID

パッチ カテゴリの UUID を指定します。

説明

パッチの説明を指定します。

戻り値**CA_SWDETECT_OK**

関数が操作を正常に完了したことを示します。

CASWDETECT_BADARGS

関数に渡された 1 つ以上の必須パラメータが空白あることを示します。

CASWDETECT_FILE_ERROR

関数が、ファイルへの書き込みに失敗したことを示します。詳細については、ログ ファイルを参照してください。

CreateDetectedSoftwareReleaseInstance

スクリプトが特定リリースのインスタンスを検出すると、**CreateDetectedSoftwareReleaseInstance** 関数が、ソフトウェア検出出力ファイルにレコードを追加します。

関数の形式

CreateDetectedSoftwareReleaseInstance (ProductName as String, ProductVersionLabel as String, ReleaseName as String, ReleaseVersionLabel as String, OptionalProperties as String) as an integer.

例：

```
CreateDetectedSoftwareReleaseInstance ("Microsoft Windows 7  
Ultimate", "6.1", "Microsoft Windows 7 Ultimate x64 64 en-us",  
"6.1.7600", "Origin=Forward Inc | TrustLevel=5 |  
InstallPath=C:\Windows | SerialNumber=1234-567-890414-86668 |  
LastAccessed=2011-11-29:-12:30 |")
```

注：この関数は、各スクリプト内で **CreateDetectedSoftwareRelease** 関数を呼び出した後に呼び出します。**CreateDetectedSoftwareReleaseInstance** 関数は、インスタンスが存在する回数だけ呼び出すことができ、各インスタンスは、検出された個別のソフトウェア レコードを取得します。

注：2つの別々の **Intellisig** で同じソフトウェア インスタンスが検出される場合、2つの検出ソフトウェア レコードが作成されます。

入力パラメータ

この関数には、以下の入力パラメータがあります。

ProductName

リリースが所属する製品名を指定します。スクリプト内で、この製品に対して **CreateDetectedSoftwareProduct** 関数を呼び出している必要があります。

ProductVersionLabel

製品のバージョン ラベルを指定します。

ReleaseName

検出されるソフトウェア リリースを指定します。

ReleaseVersionLabel

リリースのバージョン ラベルを指定します。

OptionalProperties

検出されたソフトウェア インスタンスに関連付けられるオプションのプロパティを指定します。インスタンスに対して、以下のオプションのプロパティを使用できます。

ProductGUID

スクリプトによって検出される製品 **UUID** を指定します。たとえば、**MSI** パッケージの製品 **GUID** などです。

Label

このインスタンス用の一意のラベルを指定します。たとえば、**Microsoft SQL Server** インスタンス名などです。

Origin

Intellisig 作成者の名前を指定します。

TrustLevel

Intellisig、スクリプト、またはその両方の作成者の信頼レベルを指定します。

InstallPath

エージェント コンピュータ上の製品インストール ディレクトリまたは実行可能ファイルへのパスを指定します。

SerialNumber

インスタンスのシリアル番号を指定します。

LastAccessed

インスタンスが最後にアクセスされた日時を指定します。

yyyy-mm-dd-hr : **mm** 形式で値を指定します。この値は、ローカル時間であると仮定されます。

注：時間を処理する **dmscript** 内のその他のすべての関数は、ローカル時間で動作します。**dmscript** は、出力の前に時間値を **UNIX** 形式 (**1-1-1970 UTC** からの秒数) に変換します。**DSM** エクスプローラは、実行されているマシン上でローカル タイム ゾーンを使用して、これを表示します。

CustomData

インスタンスに関して格納する別の任意のカスタム データを指定します。

戻り値

CA_SWDETECT_OK

関数が操作を正常に完了したことを示します。

CASWDETECT_BADARGS

関数に渡された 1 つ以上の必須パラメータが空白あることを示します。

CASWDETECT_FILE_ERROR

関数が、ファイルへの書き込みに失敗したことを示します。詳細については、ログファイルを参照してください。

CreateDetectedSoftwarePatchInstance

スクリプトが特定パッチのインスタンスを検出すると、**CreateDetectedSoftwarePatchInstance** 関数が、ソフトウェア検出出力ファイルにレコードを追加します。

関数の形式

CreateDetectedSoftwarePatchInstance (ProductName as String, ProductVersionLabel, as String, ReleaseName as String, ReleaseVersionLabel as String, PatchName as String, PatchVersionLabel as String, OptionalProperties as String) as an integer

例 : CreateDetectedSoftwarePatchInstance

```
CreateDetectedSoftwarePatchInstance ("Microsoft Windows 7 Ultimate", "6.1",  
"Microsoft Windows 7 Ultimate x64 64 en-us", "6.1.7600", "KB971033 x64 64  
en-us", "", "Origin=Forward Inc |TrustLevel=5 |")
```

注：この関数は、各スクリプト内で **CreateDetectedSoftwarePatch** 関数を呼び出した後で呼び出す必要があります。

CreateDetectedSoftwarePatchInstance 関数は、インスタンス数と同じ回数だけ呼び出すことができます。各インスタンスは、検出された個別のソフトウェア レコードを取得します。

注：2つの別々の **Intellisig** で同じソフトウェア インスタンスが検出される場合、2つの個別の検出レコードが **Intellisig** ごとに作成されます。

入力パラメータ

この関数には、以下の入力パラメータがあります。

ProductName

パッチが所属する製品の名前を指定します。スクリプト内で、この製品に対して **CreateDetectedSoftwareProduct** 関数を呼び出している必要があります。

ProductVersionLabel

製品のバージョン ラベルを指定します。

ReleaseName

パッチが所属するリリースの名前を指定します。スクリプト内で、このリリースに対して **CreateDetectedSoftwareRelease** 関数を呼び出している必要があります。

ReleaseVersionLabel

リリースのバージョン ラベルを指定します。

PatchName

検出されるソフトウェア パッチの名前を指定します。スクリプト内で、このリリースに対して **CreateDetectedSoftwarePatch** 関数を呼び出している必要があります。

PatchVersionLabel

パッチのバージョン ラベルを指定します。

OptionalProperties

検出されたソフトウェア インスタンスに関連付けられるオプションのプロパティを指定します。インスタンスに対して、以下のオプションのプロパティを使用できます。

ProductGUID

スクリプトによって検出される製品 **UUID** を指定します。

Label

このインスタンス用の一意のラベルを指定します。

Origin

Intellisig 作成者の名前を指定します。

TrustLevel

Intellisig、スクリプト、またはその両方の作成者の信頼レベルを指定します。

InstallPath

エージェント コンピュータ上の製品インストール ディレクトリまたは実行可能ファイルへのパスを指定します。

SerialNumber

インスタンスのシリアル番号を指定します。

LastAccessed

インスタンスが最後にアクセスされた日時を指定します。
yyyy-mm-dd-hr : mm 形式で値を指定します。この値は、ローカル時間であると仮定されます。

注：時間を処理する **dmscript** 内のその他のすべての関数は、ローカル時間で動作します。**dmscript** は、出力の前に時間値を **UNIX** 形式 (**1-1-1970 UTC** からの秒数) に変換します。**DSM** エクスプローラは、実行されているマシン上でローカル タイム ゾーンを使用して、これを表示します。

CustomData

インスタンスに関して格納する別の任意のカスタム データを指定します。

戻り値

CA_SWDETECT_OK

関数が操作を正常に完了したことを示します。

CASWDETECT_BADARGS

関数に渡された 1 つ以上の必須パラメータが空白あることを示します。

CASWDETECT_FILE_ERROR

関数が、ファイルへの書き込みに失敗したことを示します。詳細については、ログ ファイルを参照してください。

LogDetectedSoftwareError

LogDetectedSoftwareError 関数は、エラー ログ ファイルにエラー メッセージを書き込みます。この関数は、ローカライズ可能なエラー メッセージを生成します。作成関数によって生成されたエラーを処理するために、この関数を呼び出します。エンジンは、これらのエラー メッセージを読み取り、**DSM** エクスプローラに表示します。

関数の形式

LogDetectedSoftwareError (MessageID as String, Properties as String)

例 :

```
LogDetectedSoftwareError("ISE:00400","PARAM5=Microsoft Windows 7  
Ultimate x64 64 en-us|PARAM6=VersionNumber=6.1.7600  
|VersionLabel=6.1.7600 |Language=en-us |Bitness=64  
|Architecture=x64 |Manufacturer=Microsoft  
Corporation|Category=Operating Systems |Description=The Microsoft  
Windows 7 Product|PARAM7=SWDETECT_BADARGS");
```

このコード例では、ログ ファイルに以下のメッセージが追加されます。

```
Intellisig Microsoft Windows 7 version 6.1.7600, UUID  
A7C1E14A-7C93-4E17-B4E5-45B796717F49, script win7.xml, Failed to  
create software product. The parameters were ProductName:Microsoft  
Windows 7 Ultimate x64 64 en-us OptionalProperties :  
VersionNumber=6.1.7600 |VersionLabel=6.1.7600 |Language=en-us  
|Bitness=64 |Architecture=x64 |Manufacturer=Microsoft  
Corporation|Category=Operating Systems |Description=The Microsoft  
Windows 7 Product. Return Code : SWDETECT_BADARGS
```

入力パラメータ

この関数には、以下の入力パラメータがあります。

MessageID

エラー コードを指定します。利用可能なエラー コードの詳細については、「エラー コードおよびオプションプロパティ」を参照してください。

プロパティ

書式設定されたメッセージ文字列の作成に必要なプロパティのリストを指定します。処理中の各作成関数に渡されるパラメータを指定します。パラメータは、エラーテキストに一致する必要があります。正確なエラーテキストと関数構文については、「エラーコードおよびオプションプロパティ」を参照してください。エラーテキストは、PARAM8 から PARAM1 を使用して、メッセージ文字列を作成します。文字列に param（値ペア）が含まれない場合、文字列はそのまま表示されます。パラメータ PARAM1 から PARAM4 には、以下の値が自動的に割り当てられます。

- PARAM1 には、Intellisig UUID が含まれます
- PARAM2 には Intellisig 名が含まれます
- PARAM3 には Intellisig バージョンが含まれます
- PARAM4 には、関数からのリターンコードが含まれます

パラメータ PARAM5 から PARAM8 の値は、処理している作成機能によって異なります。たとえば、CreateDetectedSoftwareProduct 関数の障害を処理している場合、エラーコードとテキストは以下のようになります。

エラーコード：ISE:00400

エラーテキスト：Intellisig %1\$t version %2\$t, UUID %3\$t, script %4\$t, Failed to create software product. The parameters were ProductName:%5\$t OptionalProperties: %6\$t. リターンコード：%7\$t

この例では、PARAM5 は製品名、PARAM6 はオプションのプロパティ、また PARAM7 はリターンコードを指定します。

CloseDetectedSoftwareOutputFiles

CloseDetectedSoftwareOutputFiles 関数は、OpenDetectedSoftwareOutputFiles 関数によって開かれたファイルを閉じます。

関数の形式

```
CloseDetectedSoftwareOutputFiles() as integer
```

戻り値

CA_SWDETECT_OK

関数が操作を正常に完了したことを示します。

CASWDETECT_FILEERROR

関数が、ファイルを閉じることができなかったことを示します。詳細については、ログ ファイルを参照してください。

Intellisig に対する追加情報

Intellisig の整合性を確認に使用される証明書

CA Client Automation は、データベース内の各スクリプトと暗号化されたハッシュを関連付けることにより、Intellisig の整合性を維持します。エージェントは、ハッシュを確認した後、各 Intellisig を実行します。ハッシュの検証が失敗すると、エージェントは、Intellisig の実行を拒否し、GUI にメッセージを送信して、この検証エラー、および整合性の確認に失敗した特定の Intellisig の詳細を示します。

注：ハッシュは、ManagerSigner のタグ付き証明書 x509cert://dsmr11/cn=manager signer,o=computer associates,c=us を使用して暗号化されます。カスタム x509 証明書を使用すると、Intellisig は、暗号化に対して自動的にカスタム証明書を使用します。

32 ビットおよび 64 ビット レジストリの両方を使用した 64 ビット OS 上のソフトウェアの検出

レジストリおよびファイルエントリの両方に関して、DSM エクスプローラで従来のカスタム シグネチャを作成する場合、レジストリのアーキテクチャ値を指定できます。指定された値は、指定された Windows アーキテクチャに対してバイナリが構築されていることを確認します。

次の手順に従ってください：

1. [DSM エクスプローラ] - [ソフトウェア] - [定義] - [ソフトウェア カテゴリ] - [ソフトウェア カテゴリのリリース] - [プロパティ] - [識別] - [詳細設定] - [レジストリ] に移動します。
2. 64 ビット OS の [アーキテクチャ] ドロップダウン ボックスで、以下のいずれかの値を選択します。

32

32 ビット レジストリ ハイブ内の検索をトリガします。

64

64 ビット レジストリ ハイブ内の検索をトリガします。

注：OS グループ タイプを UNIX に変更すると、[アーキテクチャ] ドロップダウン ボックスが無効になります。

Intellisig に対するデフォルト タイムアウトの設定

Intellisig のデフォルト タイムアウトにより、不正な Intellisig がコンピュータ上で無期限に実行されないことが保証されます。コンピュータ上で実行されるすべての Intellisig スクリプト、または特定の Intellisig に対して、デフォルト タイムアウト値を定義できます。たとえばデータベース サーバなど負荷が大きい場合、および Intellisig の完了に余分な時間が必要な場合には、エージェントにより長いデフォルト時間を適用できます。デスクトップコンピュータでは、負荷が小さい場合、より短いタイムアウト値を設定できます。

次の手順に従ってください：

1. [DSM] - [エージェント] - [Asset Management] に移動します。
2. 以下のパラメータに対する値を定義します。

IntellisigDefaultExecutionTimeout

デフォルト タイムアウト値を分単位で指定します。

デフォルト：5

注：Intellisig を作成するときには、そのタイムアウト値をデフォルトに設定します。このアクションにより、エージェントが、設定されたデフォルト タイムアウトを適用することが保証されます。Intellisig に 0 より大きいタイムアウト値がある場合、Intellisig の実行時に、指定された値が適用されます。

intellisigcmd - コマンドライン ツール

intellisigcmd は、Intellisig のコマンドライン ツールです。このツールの形式は、以下のとおりです。

```
intellisigcmd <cmd> param1=value1 param2=value2 ... [<DB_Credentials>]
```

cmd

import、export、または genuuid コマンドを指定します。

DB_Credentials

MDB のデータベース クレデンシャルを指定します。デフォルトでは、クレデンシャルは comstore から取得されます。

DB クレデンシャルの指定では、以下のサンプル形式を使用します。

例：SQLServer DB クレデンシャル

```
dbvendor=mssql dbhost=myhost dbname=mdb dbuser=ca_itrm dbpassword=mypwd  
dbinstance=inst
```

例：Oracle DB クレデンシャル

```
dbvendor=oracle dbhost=myhost dbname=orcl dbuser=ca_itrm dbpassword=mypwd  
dbinstance=1521
```

intellisigcmd export -- Intellisig のエクスポート

`intellisigcmd export` コマンドでは、Intellisig をエクスポートできます。DSM エクスプローラまたはコマンドを使用して Intellisig をエクスポートできます。

コマンドの形式

```
intellisigcmd export file=<export name> [type=xml|zip] [platform=all|windows|unix]
```

export name

エクスポートする Intellisig XML または ZIP ファイルの名前を指定します。ファイル拡張子を指定せず、タイプが `xml` の場合、このコマンドは指定された名前でフォルダを作成します。

type

XML または ZIP ファイルをエクスポートするかどうかを指定します。`type` パラメータが含まれない場合、このコマンドは、エクスポートファイル拡張子に応じてエクスポートタイプを想定します。

有効な値： `xml`、`zip`

platform

どの Intellisig がエクスポートされるかを決定するために、プラットフォームを指定します。

有効な値： `all`、`windows`、`unix`

デフォルト： `all`

intellisigcmd import -- Intellisig のインポート

intellisigcmd import コマンドでは、Intellisig をインポートできます。DSM エクスプローラまたはコマンドを使用できます。

コマンドの形式

```
intellisigcmd import [file=<import source>] [type=xml|zip]  
[mode=replace|mergenew|mergeall] [updateactive=yes|no] [delete=yes|no]
```

import source

Intellisig をインポートする XML または ZIP ファイルの名前を指定します。ファイル拡張子を指定しない場合、このコマンドは、タイプに応じてファイル拡張子を想定します。

注：XML ファイルにインポートする場合、サポートするディレクトリが XML ファイルと同じフォルダに存在することを確認します。

type

XML または ZIP ファイルをインポートするかどうかを指定します。このパラメータが含まれない場合、このコマンドは、インポートファイル拡張子に応じてインポートタイプを想定します。

有効な値：xml、zip

mode

インポート モードを指定します。以下のインポート モードがサポートされています。

デフォルト : mergenew

replace

インポートされた定義で既存の定義を置換します。既存の定義が失われます。

mergenew

マネージャ上の定義に、新しい **Intellisig** バージョンを追加します。既存の定義は変更されません。

mergeall

新しい **Intellisig** バージョンを追加して、インポート ファイルに含まれている既存の定義を更新します。インポート ファイルで定義されていない **Intellig** バージョンは変更されません。

updateactive

インポート中に、アクティブな **Intellisig** バージョンが更新可能かどうかを指定します。

有効な値 : Yes、Y、true、1 または No、N、false、0

デフォルト : Yes

delete

インポートの前に **Intellisig** を削除するかどうかを指定します。 **delete** スイッチを指定しない場合、いずれの **Intellisig** もインポートの前に削除されません。

デフォルト : No

intellisigcmd genuuid -- UUID の生成

intellisigcmd genuuid コマンドでは、カスタム Intellisig の作成時に使用可能な一意の UUID を生成できます。

コマンドの形式

```
intellisigcmd genuuid [num=<count>]
```

num

生成する UUID の数を指定します。このパラメータを指定しない場合、単一の UUID が生成されます。それ以外の場合、<count> 数の UUID が生成されます。

有効な値：1 ～ 1000

表示関数

表示関数は、Linux プラットフォームではサポートされません。

ClearScreen または ClrScr - 画面のクリア

Windows プラットフォームで有効

ClearScreen または ClrScr 関数では、画面がクリアされます。

関数の形式

ClearScreen() as Boolean

表示関数では、常に TRUE が返されます。

例：

この例では、5 行が出力され、5 秒間待機した後、画面がクリアされます。

```
Dim counter as integer

'Print 5 lines
For counter = 1 to 5
    Print("Line " + str(counter))
Next

'Wait 5 seconds
Sleep(5000)

'Clear the print screen
ClearScreen()
Print("Screen Cleared")
```

InputBox - メッセージ ボックス ウィンドウの作成、表示、および操作

Windows および **Windows CE** で有効です。

InputBox 関数はメッセージ ボックスを作成、表示、および操作します。メッセージ ボックスにはアプリケーションで定義されたプロンプト、タイトル、およびアプリケーションで定義されたデフォルト値が入ったテキスト ボックス、さらに [OK] ボタンや [キャンセル] ボタンが含まれます。

このその他の関数の形式は、以下のとおりです。

```
InputBox(  
    prompt as String,  
    title as String,  
    default as String) as Boolean
```

prompt

表示するプロンプトを示します。

title

メッセージボックスのタイトルを指定します。

default

デフォルト プロンプト入力を指定します。デフォルトが変数でない場合は、構文エラーが表示されます。

注： 入力ボックスは前面に配置されません。

[OK] ボタンを選択した場合は、**True** が返され、テキスト ボックスの内容がデフォルト変数にコピーされます。 [キャンセル] ボタンを選択した場合は、**False** が返され、デフォルト変数は変更されません。

例：

以下に、InputBox の使用例を示します。

```
Dim TargetPath, Question as string
```

```
TargetPath=WindowsDir
```

```
Question="Please specify the directory to use for installation"
```

```
if InputBox(Question,"Install",TargetPath) then
```

```
    ' Place code for actual installation here...
```

```
    MessageBox("Installation completed")
```

```
else
```

```
    MessageBox("Installation aborted")
```

```
End if
```

```
End:
```

MessageBox または MsgBox - メッセージ ボックス ウィンドウの表示

Windows および Windows CE で有効です。

MessageBox または **MsgBox** 関数は、メッセージ ボックス ウィンドウを表示します。メッセージ ボックスにはメッセージやタイトル、および **style** パラメータで示された定義済みボタンの組み合わせが含まれます。

MessageBox の代わりに **MsgBox** を使用して関数を呼び出すことができます。

関数の形式

```
MessageBox(message as String, title as String, style as Integer) as Integer
```

```
MessageBox(message as String, title as String) as Integer
```

```
MessageBox(message as String, style as Integer) as Integer
```

```
MessageBox(message as String) as Integer
```

メッセージ

表示するメッセージを含む文字列を指定します。

title

メッセージ ボックス タイトルを含むオプション文字列を指定します

デフォルト : DMS

style

メッセージ ボックスの内容および動作を指定するオプションの整数を指定します。**style** パラメータには、以下のいずれかの事前定義された定数を指定できます。

MB_OK

[OK] ボタンを含むボックスを追加 (デフォルト)

MB_OKCANCEL

[OK] および [キャンセル] ボタンを含むボックスを追加

MB_YESNO

[はい] および [いいえ] ボタンを含むボックスを追加

MB_RETRYCANCEL

[再試行] および [キャンセル] ボタンを含むボックスを追加

MB_YESNOCANCEL

「はい」、「いいえ」、および「キャンセル」ボタンを含むボックスを追加

MB_ABORTRETRYIGNORE

「中止」、「再試行」、および「無視」ボタンを含むボックスを追加

MessageBox のモーダル性に関する定数

デフォルトでは、ユーザはメッセージボックスに応答してから、現在のウィンドウの作業を継続する必要があります。ただし、別のアプリケーションのウィンドウに移動して、作業することはできます。

MB_SYSTEMMODAL

すべてのアプリケーションは、ユーザがメッセージボックスに応答するまで中断されます。システム モーダル メッセージボックスは、すぐに対処する必要がある重大で、損失を引き起こす可能性のあるエラーがあることをユーザに通知する場合に使用します。

MessageBox のアイコンに関する定数

デフォルトでは、アイコンは表示されません。

MB_ICONEXCLAMATION

メッセージボックスに感嘆符アイコンを追加します。

MB_ICONINFORMATION

メッセージボックスに「i」が円で囲まれたアイコンを追加します。

MB_ICONQUESTION

メッセージボックスに疑問符アイコンを追加します。

MB_ICONSTOP

メッセージボックスに、停止記号アイコン（赤い円で囲まれた白い「X」）を追加します。

MessageBox のデフォルト ボタンに関する定数

デフォルトは以下のとおりです。1 番目のボタンがデフォルトです。

MB_DEFBUTTON2

2 番目のボタンをデフォルトにします。

MB_DEFBUTTON3

3 番目のボタンをデフォルトにします。

その他の定数

MB_SETFOREGROUND

メッセージボックスを前面ウィンドウにします。コード化されていない場合、現在の前面ウィンドウがそのまま前面に配置されます。

特定のグループの各定数を、別のグループの任意の定数と組み合わせることができます。たとえば、[中断]、[再試行]、[無視]（デフォルト）ボタン、および停止記号アイコンが配置されたメッセージボックスを表示して、ユーザが応答するまですべてのアプリケーションを中断する場合は、以下のスタイルでコード化します。

`MB_ABORTRETRYIGNORE + MB_DEFBUTTON3 + MB_ICONSTOP + MB_SYSTEMMODAL`

`style` パラメータを省略した場合、`MB_OK` が使用され、メッセージボックスには [OK] ボタンのみが配置されます。

メッセージボックスを作成するために必要なメモリがない場合、関数の戻り値はゼロです。それ以外の場合は、以下のボタン値のいずれかが返されます。

IDOK

値 1： [OK] ボタンがクリックされました。

IDCANCEL

値 2： [キャンセル] ボタンがクリックされました（またはキーボードの Esc キーが押されました）。

IDABORT

値 3 : [中止] ボタンがクリックされました。

IDRETRY

値 4 : [再試行] ボタンがクリックされました。

IDIGNORE

値 5 : [無視] ボタンがクリックされました。

IDYES

値 6 : [はい] ボタンがクリックされました。

IDNO

値 7 : [いいえ] ボタンがクリックされました。

例 :

この例は **config.sys** ファイルのバックアップ コピーの作成をユーザに確認し、必要な場合、バックアップを作成します。

```
Dim Src,Dst as string
Dim Question as string

Question="Do you want to make a backup of your config.sys?"

if MessageBox(Question,MB_YESNO)=IDYES then
    Src="C:¥CONFIG.SYS"
    Dst="C:¥CONFIG.BAK"
    if CopyFile(Src,Dst,TRUE) then MessageBox("CONFIG.BAK created.")
end if
```

ProgressBar - プログレス バーの設定

Windows プラットフォームで有効

ProgressBar 関数は、メイン ウィンドウのプログレス バーを指定された割合に設定します。

表示関数の形式:

```
ProgressBar(percentage as Integer) as Boolean
```

percentage

進捗状況の値（パーセント）。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

以下の例では、**ProgressBar** および **ProgressText** 関数を示します。

```
Dim counter as Integer
For counter = 1 TO 10
    ProgressBar(counter * 10)
    ProgressText("Now processing : " + Str(counter) )
    Sleep(500)      'Wait half second
Next
```

ProgressText - 進捗テキスト フィールドへの文字列のコピー

Windows でのみ有効です。

ProgressText 関数はメイン ウィンドウの進捗テキスト フィールドに文字列をコピーします。

表示関数の形式:

```
ProgressText(text as String) as Boolean
```

テキスト

進捗テキストフィールドのテキストを定義します。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

以下の例では、**ProgressBar** および **ProgressText** 関数を示します。

```
Dim counter as Integer
For counter = 1 TO 10
  ProgressBar(counter * 10)
  ProgressText("Now processing : " + Str(counter) )
  Sleep(500)      'Wait half second
Next
```

環境関数

環境関数は、Windows NT/9x および UNIX プラットフォームで使用可能です。

EnvExport - テキスト ファイルへの環境のエクスポート

UNIX および Windows で有効です。

EnvExport 関数では、現在の環境がテキスト ファイルにエクスポートされます。

関数の形式

```
EnvExport(filename as String) as Boolean
```

filename

テキスト ファイルの名前を定義します。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例 :

```
Rem
Rem Exports defined environment variables
Rem
EnvExport("c:¥temp¥env.dat")
```

EnvGetFirst - 最初の環境変数の名前の取得

UNIX および Windows で有効です。

EnvGetFirst 関数は、最初の環境変数の名前を取得します。

関数の形式

```
EnvGetFirst() as string
```

最初の環境変数の名前を含む文字列が返されるように定義します。環境変数がこれ以上存在しない場合は、空の文字列が返されます。

例 :

以下の例では、すべての環境変数の名前が表示されます。

```
Dim txt as String

txt = EnvGetFirst()
While txt <> ""
    Print(txt)
    txt = EnvGetNext()
Wend
```

EnvGetNext - 以降の環境変数の名前の取得

UNIX および Windows で有効です。

EnvGetNext 関数は EnvGetFirst 関数を継続して、以降の環境変数の名前を取得します。 EnvGetNext を呼び出すごとに、環境変数が見つからなくなるまで、環境変数名が 1 つ戻されます。

関数の形式

```
EnvGetNext() as string
```

この戻り値は、環境変数の名前を含む文字列です。 環境変数がこれ以上存在しない場合は、空の文字列が返されます。

例：

以下の例では、すべての環境変数の名前が表示されます。

```
Dim txt as String

txt = EnvGetFirst()
While txt <> ""
    Print(txt)
    txt = EnvGetNext()
Wend
```

EnvGetString - 環境変数の値の取得

UNIX および Windows で有効です。

EnvGetString 関数は、環境変数の値を取得します。

関数の形式

```
EnvGetString(name as string) as string
```

name

環境変数の名前を定義します。

この関数は、名前に関連付けられた値を含む文字列を返します。名前で指定された環境変数が存在しない場合は、空の文字列が返されます。

例：

以下の例では、現在の検索パスが表示されます。

```
Rem variable `MyPath`  
Rem  
Dim txt as String  
  
txt = EnvGetString("PATH")  
  
Print("The path: "+txt)  
EnvSetString("MyPath",txt)  
Print("My path: "+txt)
```

EnvImport - 環境変数のインポート

UNIX および Windows で有効です。

EnvImport 関数はテキスト ファイルから環境変数をインポートします。

関数の形式

```
EnvImport(filename as String) as Boolean
```

filename

ファイルの名前を定義します。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

以下の例では、ファイルから追加環境変数をインポートします。

```
EnvImport("c:%temp%env.dat")
```

C:%temp%env.dat contains for example:

```
Xantippe=greek  
Yokohama=japanese  
Fuel=explosive
```

ファイルの内容に関する関数

ファイルの内容に関する関数は任意のプラットフォームで使用できます。

CloseFile - ファイルのクローズ

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

CloseFile 関数は、OpenFile または CreateFile への呼び出しから取得されたファイルハンドルに関連するファイルを閉じます。

関数の形式

```
CloseFile(handle as Integer) as Boolean
```

handle

OpenFile または CreateFile の呼び出しから取得されたファイルハンドル。

この関数が正常に終了すると、**TRUE** が返されます（ゼロ以外）。この関数に失敗すると（たとえば、ハンドルが有効なオープン ファイルのハンドルでない場合）、この関数は **FALSE** を返します。

例：

この例では、**CONFIG.SYS** ファイルのバックアップが作成されます。

```
Dim fIn, fOut as integer    ' Declare file handles

Dim OneLine as string      ' String to hold one line

' First open the Input file...

fIn=OpenFile("C:¥CONFIG.SYS",0_READ)
if fIn<0 then
    MsgBox("Unable to open input file","Error")
    Goto End
End if

' ...Then create the output file...

fOut=CreateFile("C:¥CONFIG.BAK")
if fOut<0 then
    MsgBox("Unable to create output file","Error")
    Goto End
End if

' ...Copy lines until none left...

while Not(Eof(fIn))
    if ReadFile(fIn,OneLine) then WriteFile(fOut,OneLine)
wend

' ...Close Files, and signal success.

CloseFile(fIn)
CloseFile(fOut)
MsgBox("A backup of the CONFIG.SYS file was created","MyScript")

end:
```

CreatePipe - パイプの作成

UNIX および Windows で有効です。

注： この関数は、CA Client Automation で新しく使用されるようになったため、古いバージョンのスクリプト インタープリタでは動作しません。

CreatePipe 関数は、サーバ役割内に読み書き用の名前付きパイプを作成します。

関数の形式

CreatePipe(pipename as String, access as Integer) as Integer

pipename

作成するパイプの名前を指定します。 UNIX では、pipename は有効なパス名です。

Windows では、以下の形式になります。

¥¥.¥pipe¥xxx

ここで、xxx には「¥」を除く任意の文字を含めることができます。

access

パイプを使用する方向を指定します。

O_READ

(値 0) 読み取りのために開きます。

O_WRITE

(値 1) 書き込みのために開きます。

この関数は、関数 `ReadFile`、`WriteFile`、および `CloseFile` で使用できるファイルハンドルを返します。

`CreatePipe` はパイプ サーバとして機能するプロセス内で使用します。パイプ クライアントは `OpenPipe` 関数を使用します。

パイプは、ブロック モードで機能します。`CreatePipe` の後に続く読み書き関数は、クライアントが反対の関数を使用するまで待機します。

注：この関数は Windows 95/98/ME では使用できません。

この関数が正常に完了すると、負でない整数（ファイルハンドル）が返されます。この関数が失敗すると、-1 が返されます。

例：

以下の例では、`OpenPipe` がファイル `p0cl.dms` として格納されると想定しています。

```
rem example named pipe server
rem create two pipes
rem start client process
rem write command to one pipe
rem read response from second pipe

dim rc, h0, h1 as integer
dim pnam0, pnam1 as string
dim stopstring as string
dim s, cmd as string
dim iswin as boolean

if left(osstring, 3) = "Win" then
    pnam0 = "¥¥.¥pipe¥p0rsp"
    pnam1 = "¥¥.¥pipe¥p0cmd"
    iswin = 1
else
    pnam0 = "/tmp/p0rsp"
    pnam1 = "/tmp/p0cmd"
    iswin = 0
endif
stopstring = "---END---"
```

```
h0 = CreatePipe(pnam0, 0_READ)      'read response form client
h1 = CreatePipe(pnam1, 0_WRITE)     'send command to client
if h0 < 0 or h1 < 0 then
    print "create pipe failure. h0: " + str(h0) + " h1: " + str(h1)
    exit
end if

if iswin then
    cmd = "dmscript p0cl.dms -o_dms: p0clout.txt -script " + pnam1 + " " +
pnam0
    rc = Exec(cmd, 0)
else
    cmd = "dmscript p0cl.dms -o_dms: p0clout.txt -script " + pnam1 + " " + pnam0
+ " &"
    rc = Exec(cmd)
end if
print "exec rc: " + str(rc) + " cmd: " + cmd

rc = WriteFile(h1, "command")
if not(rc) then
    print "write command failed"
else
    while rc
        rc = ReadFile(h0, s)
        print "read response rc: " + str(rc) + " " + s
        if s = stopstring then
            rc = 0
            print "test successful"
        end if
    wend
end if

CloseFile(h0)
CloseFile(h1)
```

CreateFile - 新規ファイルの作成

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

注：Windows 9x では、この関数を使用して .ini ファイルを作成しないでください。Windows OS は、WriteIniSection() または WriteIniEntry() で指定された .ini ファイルを作成します。

CreateFile 関数は、書き込み用ファイルを作成します。

関数の形式

```
CreateFile(filename as String, mode as Integer) as Integer
```

```
CreateFile(filename as String) as Integer
```

filename

ファイル名を指定します。

mode

ファイルを作成するディレクトリを指定します。この値は、以下の定義済み定数のいずれかです。

O_TEXT

(値 0、デフォルト) テキスト モード

O_BINARY

(値 1) バイナリ モード

mode パラメータを省略すると、テキスト モードになります。

この関数に成功すると、戻り値は負でない整数（つまりファイルハンドル）になります。この関数が失敗すると、-1 が返されます。

例：

この例では、**CONFIG.SYS** ファイルのバックアップが作成されます。

```
Dim fIn, fOut as integer ' Declare file handles
```

```
Dim OneLine as string ' String to hold one line
```

```
' First open the Input file...
```

```
fIn=OpenFile("C:\CONFIG.SYS",0_READ)
```

```
if fIn<0 then
```

```
    MessageBox("Unable to open input file","Error")
```

```
    Goto End
```

```
End if
```

```
' ...Then create the output file...
```

```
fOut=CreateFile("C:\CONFIG.BAK")
if fOut<0 then
    MsgBox("Unable to create output file","Error")
    Goto End

End if

' ...Copy lines until none left...

while Not(Eof(fIn))
    if ReadFile(fIn,OneLine) then WriteFile(fOut,OneLine)
wend

' ...Close Files, and signal success.

CloseFile(fIn)
CloseFile(fOut)
MsgBox("A backup of the CONFIG.SYS file was created","MyScript")

end:
```

EOF - ファイルの終わりのチェック

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

EOF 関数は、ファイルの終わりに到達したかどうかを判別します。

関数の形式

Eof(handle as Integer) as Boolean

handle

ファイルハンドルを指定します。

この関数は、ファイルの終わりに到達した場合、**TRUE** を返します。それ以外の場合は、**False** を返します。

注：直前の読み取り操作がファイルの終わりに到達した場合、EOF は **TRUE** を返します。この関数は、以降の読み取りコマンドがファイルの終わりに到達したかどうかを前もって確認しません。

例：

この例では、CONFIG.SYS ファイルのバックアップが作成されます。

```
Dim fIn, fOut as integer    ' Declare file handles
Dim OneLine as string      ' String to hold one line

Rem First open the Input file...

fIn=OpenFile("C:¥CONFIG.SYS",0_READ)
if fIn<0 then
    MsgBox("Unable to open input file","Error")
    Goto End
End if

Rem ...Then create the output file...

fOut=CreateFile("C:¥CONFIG.BAK")
if fOut<0 then
    MsgBox("Unable to create output file","Error")

CloseFile(fIn)
    Goto End
End if

Rem ...Copy lines until none left...

while Not(Eof(fIn))
    if ReadFile(fIn,OneLine) then WriteFile(fOut,OneLine)
wend

Rem ...Close Files, and signal success.

CloseFile(fIn)
CloseFile(fOut)
MsgBox("A backup of the CONFIG.SYS file was created","MyScript",MB_OK)

end:
```

OpenFile - ファイルのオープン

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

OpenFile 関数は読み書きのためにファイルを開きます。

関数の形式

```
OpenFile(filename as String, access as Integer, mode as Integer) as Integer
```

```
OpenFile(filename as String, access as Integer) as Integer
```

filename

オープンするファイル名を指定します。

access

ファイルのオープン方法を指定します。access パラメータには、以下のいずれかの事前定義された定数を指定できます。

O_READ

(値 0) 読み取りのために開きます。

O_WRITE

(値 1) 書き込みのために作成します。指定したファイルが存在する場合は、上書きされます。

O_APPEND

(値 2) ファイルの終わりに、書き込みのために開きます。指定したファイルが存在しない場合は、ファイルが作成されます。

O_UPDATE

(値 3) 読み書きのために開きます。

mode

バイナリ モードを指定するオプション パラメータを指定します。

mode パラメータには、以下のいずれかの事前定義された定数を指定できます。

O_TEXT

値 0 (テキスト モード)

O_BINARY

値 1 (バイナリ モード)

mode パラメータを省略すると、テキスト モードになります。

この関数に成功すると、戻り値は負でない整数 (つまりファイルハンドル) になります。この関数が失敗すると、-1 が返されます。

例 :

この例では、**CONFIG.SYS** ファイルのバックアップが作成されます。

```
Dim fIn, fOut as integer    ' Declare file handles
Dim OneLine as string      ' String to hold one line

' First open the Input file...

fIn=OpenFile("C:¥CONFIG.SYS",O_READ)
if fIn<0 then
    MsgBox("Unable to open input file","Error")
    Goto End
End if

' ...Then create the output file...

fOut=CreateFile("C:¥CONFIG.BAK")
if fOut<0 then
    MsgBox("Unable to create output file","Error")
    Goto End
End if

' ...Copy lines until none left...

while Not(Eof(fIn))
    if ReadFile(fIn,OneLine) then WriteFile(fOut,OneLine)
wend

' ...Close Files, and signal success.
```

```
CloseFile(fIn)
CloseFile(fOut)
MessageBox("A backup of the CONFIG.SYS file was created", "MyScript")

end:
```

OpenPipe - パイプのオープン

UNIX および Windows で有効です。

注：この関数は、CA Client Automation で新しく使用されるようになったため、古いバージョンのスクリプトインタプリタでは動作しません。

OpenPipe 関数は、クライアント役割の読み書き用の名前付きパイプを開きます。

関数の形式

```
OpenPipe(pipename as String, access as Integer) as Integer
```

pipename

オープンするパイプの名前を指定します。

UNIX では、**pipename** は有効なパス名です。

Windows では、**¥¥.¥pipe¥xxx** の形式になります。ここで、**xxx** には「¥」を除く任意の文字列を含むことができます。

access

パイプを使用する方向を指定します。

O_READ

(値 0) 読み取りのために開きます。

O_WRITE

(値 1) 書き込みのために開きます。

返されたファイルハンドルは、関数 `ReadFile`、`WriteFile`、および `CloseFile` で使用できます。 `OpenPipe` はパイプクライアントとして機能するプロセス内で使用します。 `OpenPipe` を呼び出す前に、`CreatePipe` を呼び出します。

パイプは、ブロック モードで機能します。 `OpenPipe` の後に続く読み書き関数は、サーバが反対の関数を使用するまで待機します。

この関数が正常に完了すると、負でない整数（ファイルハンドル）が返されます。 この関数が失敗すると、`-1` が返されます。

例：

以下の例では、名前付きパイプクライアントを表示します。

```
dim rc, h0, h1, i as integer
dim pnam0, pnam1 as string
dim stopstring as string
dim s, s1 as string
dim iswin as boolean

pnam0 = argv(1)
pnam1 = argv(2)
stopstring = "---END---"

h0 = OpenPipe(pnam0, 0_READ)
h1 = OpenPipe(pnam1, 0_WRITE)
if h0 < 0 or h1 < 0 then
    print "open pipe failure. h0: " + str(h0) + " h1: " + str(h1)
    print "pnam0: " + pnam0 + " pnam1: " + pnam1
    exit
end if

rc = ReadFile(h0, s)
print "read command rc: " + str(rc) + " " + s
if rc then
    for i = 1 to 5
        s1 = "RSP " + s + " line " + str(i)
        rc = WriteFile(h1, s1)
        if not(rc) then
            print "write failed: " + s1
            exit for
        end if
    end if
end if
```

```
        next i
    if rc then
        rc = WriteFile(h1, stopstring)
        if rc then
            print "test successful"
        end if
    end if
end if

CloseFile(h0)
CloseFile(h1)
```

ReadFile - ファイルからのデータの読み取り

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

ReadFile 関数はファイルからデータを読み取ります。

ファイルの内容に関する関数は、以下のいずれかの形式をとります。

ReadFile (handle as Integer, buffer as String) as Boolean

ReadFile (handle as Integer, buffer as String, buflen as Integer) as Boolean

ReadFile (handle as Integer, buffer as Void) as Boolean

ReadFile (handle as Integer, buffer as Void, buflen as Integer) as Boolean

handle

OpenFile または CreateFile の以前の呼び出しで返されたファイルハンドルを指定します。

buffer

任意のタイプの変数を指定します。

配列タイプが char のバッファは、文字列として扱われます。

引数 **buffer** が文字列タイプ変数の場合、この関数は現在のファイル位置から読み取りを行い、改行文字を読み取るか、または指定最大文字数に達した場合に、読み取りを停止します。引数 **buffer** がその他のタイプの場合は、現在のファイル位置から読み取り、バッファが一杯になるか、またはファイルポインタがファイルの終わりに達したときに停止します。

void は、オブジェクトのタイプに、char タイプの文字列および配列を除く任意の有効なタイプを設定できることを意味します。

buflen

ファイルからバッファに読み取るバイト数を指定します。ファイルのバイト数が不十分な場合、残りのすべてのバイトがバッファに送信されます。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

注： **ReadFile()** 関数のフォーマットを混在させないでください。予期せぬ結果になることがあります。

その他のフォーマットと異なり、最初のフォーマットは、文字列を改行記号またはファイルの終わりで終了するテキスト行として読み取り、改行記号を挿入しないで情報をバッファに格納します。インタープリタはこの関数を、その他の 3 つのフォーマットと異なるシステム読み取り関数に対応付けます。その他の **ReadFile** フォーマットは、一般的なシステム読み取り関数に対応付けられます。

各システム読み取り関数はそれぞれ異なるキャッシュを使用し、ポインタを読み取ってデータを取得することもあります。フォーマットが混在している場合、データはディスクに格納された順番で読み込まれないことがあります。

最初のフォーマットの場合、読み取られた文字列は、改行記号またはファイルの終わりで終了するテキスト行です。この情報はこの改行を除いて、バッファに格納されます。その他の **ReadFile** 関数では、この方法が適用されません。

インタープリタはこの関数を、その他の 3 つのフォーマット（一般的なシステム読み取り関数に対応するフォーマット）と異なるシステム読み取り関数に対応付けます。これらの各システム読み取り関数もそれぞれ異なるキャッシュを使用し、ポインタを読み取ってデータを取得します。フォーマットが混在している場合、データはディスクに格納された順番で読み込まれないことがあります。

例：

```
Dim fIn, fOut as integer    ' Declare file handles
Dim OneLine as string      ' String to hold one line

Rem First open the Input file...

fIn=OpenFile("C:¥CONFIG.SYS",0_READ)
if fIn<0 then
    MsgBox("Unable to open input file","Error")
    Goto End
End if

Rem ...Then create the output file...

fOut=CreateFile("C:¥CONFIG.BAK")
if fOut<0 then
    MsgBox("Unable to create output file","Error")

CloseFile(fIn)
    Goto End
End if

Rem ...Copy lines until none left...

while Not(Eof(fIn))
    if ReadFile(fIn,OneLine) then WriteFile(fOut,OneLine)
wend

Rem ...Close Files, and signal success.

CloseFile(fIn)
CloseFile(fOut)
MsgBox("A backup of the CONFIG.SYS file was created","MyScript",MB_OK)

end:
```

SeekFile--オープン ファイルの現在位置の変更

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

SeekFile 関数は、オープン ファイル内の現在位置を変更します。

関数の形式

```
Seekfile(handle as integer, position as integer) as Boolean
```

handle

以前の呼び出しによって返されたハンドルを指定します。

position

ファイル内の新しい位置を指定します。 **position** が **0** の場合、位置はファイルの先頭に設定されます。次の読み書きは、ファイル内の新しい位置で実行されます。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例 :

```
'      4 番目のダブル ワードのファイルを更新し、 0 に設定します。
Dim fHandle as integer

fHandle = OpenFile("out5.a", 0_UPDATE)
if (fHandle = -1) then
    Print("OpenFile("out5.a", 0_UPDATE) failed.")
    exit
endif
if Not(SeekFile(fHandle, 16)) then
    Print("SeekFile(fHandle, 16) failed.")
    exit
endif
if Not(WriteFile(fHandle, 0)) then
    Print("WriteFile(fHandle, 0) failed.")
    exit
endif
CloseFile(fHandle)
```

WriteFile - ファイルへのデータの書き込み

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

WriteFile 関数はファイルにデータを書き込みます。

関数形式 :

WriteFile(handle as Integer, buffer as String) as Boolean

WriteFile(handle as Integer, buffer as String, buflen as Integer) as Boolean

WriteFile(handle as Integer, buffer as Void) as Boolean

WriteFile(handle as Integer, buffer as Void, buflen as Integer) as Boolean

handle

`OpenFile` または `CreateFile` から返されたファイルハンドルを指定します。

buffer

任意のタイプの変数を指定します。

buflen

ファイルに格納する文字数を指定します。文字数が `buflen` を超えた場合は、最初の `buflen` 個の文字列のみがファイルに格納されます。それ以外の場合は、バッファ全体が格納されます。

この関数が正常に完了すると、`TRUE` が返されます。それ以外の場合は、`FALSE` が返されます。

注： 複数の `WriteFile()` 関数フォーマットを混在させないでください。フォーマットを混在させると、予期せぬ結果になることがあります。

インタープリタは最初のフォーマットを別の方法で処理します。文字列をディスクに書き込む場合、インタープリタは常に改行記号を追加します。

インタープリタは最初のフォーマットを、その他の 3 つのフォーマット（一般的なシステム書き込み関数に対応するフォーマット）と異なるシステム書き込み関数に対応付けます。各システム書き込み関数はそれぞれ異なるキャッシュを使用して、書き込まれたデータを格納することがあります。クローズ時間に、キャッシュが順番にディスクに書き込まれます。フォーマットが混在している場合、データは正しい順序で表示されないことがあります。

インタープリタは、別の方法で以下の形式を処理します。

```
WriteFile(handle as Integer, buffer as String) as Boolean
```

文字列がディスクに書き込まれる場合は、改行が常に追加されます。この方法は、その他のタイプのフォーマットには適用されません。インタープリタはこの関数を、その他の 3 つのフォーマット（一般的なシステム書き込み関数に対応するフォーマット）と異なるシステム書き込み関数に対応付けます。各システム書き込み関数もそれぞれ異なるキャッシュを使用して、書き込まれたデータを格納することもあります。クローズ時間に、キャッシュが順番にディスクに書き込まれます。フォーマットが混在している場合、データは書き込まれた順番で表示されないことがあります。したがって、このフォーマットとほかのフォーマットを混在させないでください。

例：

```
Dim fIn, fOut as integer    ' Declare file handles
Dim OneLine as string      ' String to hold one line

Rem First open the Input file...

fIn=OpenFile("C:¥CONFIG.SYS",0_READ)
if fIn<0 then
    MsgBox("Unable to open input file","Error")
    Goto End
End if

Rem ...Then create the output file...

fOut=CreateFile("C:¥CONFIG.BAK")
if fOut<0 then
    MsgBox("Unable to create output file","Error")

CloseFile(fIn)
    Goto End
End if

Rem ...Copy lines until none left...

while Not(Eof(fIn))
    if ReadFile(fIn,OneLine) then WriteFile(fOut,OneLine)
wend

Rem ...Close Files, and signal success.
```

```
CloseFile(fIn)
CloseFile(fOut)
MsgBox("A backup of the CONFIG.SYS file was created", "MyScript", MB_OK)

end:
```

ファイル システム関数

CopyFile or Copy Link - 既存ファイルのコピー

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

CopyFile または **CopyLink** 関数は既存ファイルを新しいファイルにコピーします。

関数の形式

```
CopyFile(
    source as String,
    target as String,
    overwrite as Boolean) as Boolean

CopyFile(source as String, target as String) as Boolean

source
```

既存ソース ファイルのファイル名を指定します。

target

新規ファイルの名前を指定します。

overwrite

関数で既存のターゲットを上書きするかどうかを指定します。デフォルトの **TRUE** では、既存のターゲットが上書きされます。ターゲット ファイルがすでに存在している場合は、**FALSE** になると関数が失敗します。

この関数が完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例 :

この例ではバックアップディレクトリが作成され、このディレクトリに `autoexec.bat` がコピーされ、`autoexec.bat` に名前が変更されます。

```
HELP_GETFILEATTRIBUTES
```

```
If Not(ExistDirectory("c:¥backup")) then CreateDirectory("c:¥backup")
CopyFile("c:¥autoexec.bat", "c:¥backup¥autoexec.bat")
RenameFile("c:¥backup¥autoexec.bat", "c:¥backup¥autoexec.bak")
```

CopyTree - ディレクトリ ツリーのコピー

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

CopyTree 関数はディレクトリ ツリーをコピーします。ターゲットパスがない場合は、作成されます。

関数の形式

```
CopyTree(
    source as String,
    target as String,
    overwrite as Boolean) as Boolean
```

`CopyTree(source as String, target as String) as Boolean`

source

既存ディレクトリのの名前を示します。

target

新規ディレクトリの名前を指定します。

overwrite

関数で既存のターゲットを上書きするかどうかを指定します。

デフォルトの **TRUE** では、既存のターゲットが上書きされます。ターゲットファイルがすでに存在している場合は、**FALSE** になると関数が失敗します。

例：

以下の例では、**clientws** ディレクトリ ツリーを **cliws** にコピーします。

```
CopyTree("c:¥clientws", "c:¥cliws")
```

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

CreateDirectory または Mkdir - 新規ディレクトリの作成

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

CreateDirectory または **Mkdir** 関数は新規ディレクトリを作成します。

関数の形式

```
Mkdir(pathstring as String) as Boolean
```

```
CreateDirectory(pathstring as String) as Boolean
```

pathstring

作成するディレクトリの名前を指定します。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

この例ではバックアップディレクトリが作成され、このディレクトリに **autoexec.bat** がコピーされ、**autoexec.bat** に名前が変更されます。

```
HELP_GETFILEATTRIBUTES
```

```
If Not(ExistDirectory("c:¥backup")) then CreateDirectory("c:¥backup")
CopyFile("c:¥autoexec.bat", "c:¥backup¥autoexec.bat")
RenameFile("c:¥backup¥autoexec.bat", "c:¥backup¥autoexec.bak")
```

CreateFile - 新規ファイルの作成

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

注： Windows 9x では、この関数を使用して .ini ファイルを作成しないでください。 Windows OS は、WriteIniSection() または WriteIniEntry() で指定された .ini ファイルを作成します。

CreateFile 関数は、書き込み用ファイルを作成します。

関数の形式

```
CreateFile(filename as String, mode as Integer) as Integer
```

```
CreateFile(filename as String) as Integer
```

filename

ファイル名を指定します。

mode

ファイルを作成するディレクトリを指定します。 この値は、以下の定義済み定数のいずれかです。

O_TEXT

(値 0、デフォルト) テキスト モード

O_BINARY

(値 1) バイナリ モード

mode パラメータを省略すると、テキスト モードになります。

この関数に成功すると、戻り値は負でない整数（つまりファイルハンドル）になります。 この関数が失敗すると、-1 が返されます。

例：

この例では、**CONFIG.SYS** ファイルのバックアップが作成されます。

```
Dim fIn, fOut as integer    ' Declare file handles

Dim OneLine as string      ' String to hold one line

' First open the Input file...
```

```
fIn=OpenFile("C:¥CONFIG.SYS",0_READ)
if fIn<0 then
    MsgBox("Unable to open input file","Error")
    Goto End
End if

' ...Then create the output file...

fOut=CreateFile("C:¥CONFIG.BAK")
if fOut<0 then
    MsgBox("Unable to create output file","Error")
    Goto End

End if

' ...Copy lines until none left...

while Not(Eof(fIn))
    if ReadFile(fIn,OneLine) then WriteFile(fOut,OneLine)
wend

' ...Close Files, and signal success.

CloseFile(fIn)
CloseFile(fOut)
MsgBox("A backup of the CONFIG.SYS file was created","MyScript")

end:
```

DeleteFile または DelFile - ファイルの削除

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

DeleteFile または DelFile 関数は、指定されたファイルを削除します。

関数の形式

DeleteFile(filename as String) as Boolean

DelFile(filename as String) as Boolean

DeleteFile(filename as String, mode as Integer) as Boolean

DelFile(filename as String, mode as Integer) as Boolean

filename

ファイルの名前およびパスを指定します。

mode

Windows でのみ使用します。mode 値は以下のとおりです。

- 0 読み取り専用でないファイルを削除します。
- 1 読み取り専用のファイルを含めて削除します。
- 2 次に再起動するまで、削除を延期します。

この関数が正常に完了すると、TRUE が返されます。それ以外の場合は、FALSE が返されます。

例：

```
Rem
Rem By deleting the file Ncclient.ini a new workstation name detect
Rem will be forced next time the Unicenter Asset Management agent is executed.
Rem
DeleteFile(ComputerPath+"Ncclient.ini")
Rem ComputerPath is an Unicenter Asset Management constant
```

DeleteTree または DelTree - ディレクトリ ツリーの削除

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

DeleteTree または DelTree 関数は、すべてのサブディレクトリを含めて、指定されたディレクトリを（空でない場合も）削除します。

注：指定したディレクトリは、書き込み保護されていない必要があります。

関数の形式

```
DeleteTree(pathname as String) as Boolean
```

```
DelTree(pathname as String) as Boolean
```

pathname

削除するツリーのパスを指定します。

この関数が正常に完了すると、TRUE が返されます。それ以外の場合は、FALSE が返されます。

例：

```
Rem
Rem By deleting the directory bak this forces a full collect next time
Rem the Unicenter Asset Management agent is executed.
Rem
DeleteTree(ComputerPath+"bak")
Rem ComputerPath is a Unicenter Asset Management constant
```

ExistDir または ExistDirectory - ディレクトリの有無の調査

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

ExistDir または ExistDirectory 関数は、ディレクトリの有無を調べます。

関数の形式

```
ExistDir(pathname as String) as Boolean
```

```
ExistDirectory(pathname as String) as Boolean
```

pathname

ディレクトリ パスを指定します。

ディレクトリが存在する場合、関数の戻り値は **True**、それ以外の場合は **False** です。

例：

```
HELP_GETFILEATTRIBUTES
```

```
If Not(ExistDirectory("c:¥backup")) then CreateDirectory("c:¥backup")
CopyFile("c:¥autoexec.bat", "c:¥backup¥autoexec.bat")
RenameFile("c:¥backup¥autoexec.bat", "c:¥backup¥autoexec.bak")
```

ExistFile - ファイルの有無の調査

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

ExistFile 関数はファイルの有無を調べます。

関数の形式

```
ExistFile(filename as String) as Boolean
```

filename

ファイルパスを識別します。

ファイルが存在する場合、関数の戻り値は **True**、それ以外の場合は **False** です。

例：

```
Dim file, section, entry, value as string

Dim LF, CR as char
Dim rtr as integer

ClrScr()
LF = 0x0a
CR = 0x0d
file = "c:\dmscript.ini"
if Not(ExistFile(file)) then
    rtr = CreateFile(file, 0_TEXT)
    if rtr = -1 then
        SetStatus(1)
        exit
    end if
    closeFile(rtr)
end if

section = "Section 1"
value = "Param_1 = Wert_1" + LF + "Param_2 = Wert_2"

if WriteIniSection( section, value, file) then
    Print("WriteIniSection successfully completed.")
else
    Print("WriteIniSection failed.")
endif

section = "Section 2"
entry = "Param_3"
value = "Wert_3"

if WriteIniEntry( section, entry, value, file) then
    Print("WriteIniFile successfully completed.")
else
    Print("WriteIniFile failed.")
endif
```

```
section = "Section 1"
rtr = ReadIniSection(section, value, file)
if (rtr > 0) then
    Print( Str(rtr) + CR + LF + value)
else
    Print("ReadIniSection failed.")
endif

section = "Section 2"
entry = "Param_3"
rtr = ReadIniEntry(section, entry, value, file)

if (rtr > 0) then
    Print( Str(rtr) + CR + LF + entry + " = " + value)
else
    Print("ReadIniEntry failed.")
endif

endif
```

FindClose - 検索処理の終了

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

FindClose 関数は検索処理を終了します。

関数の形式

FindClose(Handle as Integer) as Boolean

handle

終了する検索処理のハンドルを指定します。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

```
Function contents(root as string) As Boolean
    Dim fileName As String
    Dim iHandle As Integer
    Dim rootRoot As String
```

```

    rootRoot = Mid(root, 0, len(root)-1)
    iHandle = FindFirstFile(root, fileName)) Then
    If iHandle = 0 Then
        Print("No files found at "" + root + "")
    Else
        Repeat
            If Is_Dir(rootRoot + fileName) Then
                Print(""" + rootRoot + fileName + "" is a
directory")
                if Not(fileName = ".") AND Not(fileName = "..") Then
                    contents(rootRoot + fileName + "¥*")
                End If
            Else
                Print(""" + rootRoot + fileName + "" is a file")
            End If
            Until Not(FindNextFile(iHandle, fileName))

            FindClose(iHandle)
        End If
        contents = TRUE
    End Function

Dim i As Integer
ClrScr()
if Argc() = 1 Then
    終了
End If
For i=1 To Argc()-1
    Print(NEWLINE$ + "Recursive List of "" + Argv(i) + """)
    Contents(Argv(i))
Next i

```

Invokation: dmscript sample.dms c:¥* -w_dms

関連項目 :

[FindFirstFile - ディレクトリ内のファイル検索 \(P. 93\)](#)

FindFirst - 検索で最初に見つけたファイルの名前を返す

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

FindFirst 関数では、指定のフィルタに一致する最初のファイルの名前が返されます。

関数の形式

```
FindFirst(Filter as string, Attrib as integer) as string
```

```
FindFirst(Filter as string) as string
```

Filter

スキャンするファイルを指定します。このパラメータには、検出するファイルに対する、ドライブ指定、パス、およびファイル名を含めることができます。ファイル名には、ワイルドカード文字 (? または *) を含めることができます。

Attrib

スキャンの対象となるファイルを選択するためのファイル属性を指定します。Attrib パラメータが省略されている場合、FA_ALLFILES が想定されます。Attrib は、以下の値の OR 演算により指定できます。

FA_NONE

通常のファイル、属性なし

FA_RDONLY

読み取り専用ファイル

FA_HIDDEN

隠しファイル

FA_SYSTEM

システム ファイル

FA_LABEL

ボリューム ラベル

FA_DIR

ディレクトリ

FA_ARCHIVE

アーカイブ ファイル

FA_ALLFILES

すべてのファイル

FA_ANYFILE

任意のファイル

正常に終了した場合、FindFirst 関数では、最初に一致したファイルの名前を含む文字列が返されます。それ以外の場合は、空の文字列が返されます。

例：

```
Dim Filename as string

Filename=FindFirst("C:¥*.BAT",FA_ANYFILE)
while Filename<>""
    print Filename
    Filename=FindNext()
wend
```

FindFirstFile - ディレクトリ内のファイル検索

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

FindFirstFile 関数では、指定のディレクトリ内のファイルが検索されます。

関数の形式

```
FindFirstFile(namespec as String, filename as String) as Integer
```

namespec

ディレクトリパス、またはワイルドカードファイル指定を含むファイルパスを識別します。リテラル文字列を使用しないでください。例：

以下は使用不可です。

```
Dim filename as string
Dim findHandle as integer
findHandle=FindFirstFile("c:¥windows¥*.dll",filename).
```

以下は使用可能です。

```
Dim filename as string
Dim filehandle as integer
Dim path as string
Path="c:¥windows¥*.dll"
Filehandle=FindFirstFile(path,filename)
```

filename

見つかったファイルの名前を含む出力パラメータ。

正常に終了した場合、この関数では、**FindNextFile** および **FindClose** の入力として使用する検索ハンドルが返されます。それ以外の場合、**0** が返されます。関数が失敗した場合、**0** が返されます。

例：

この例では、引数として渡されるディレクトリの内容が繰り返しリストされます。

```
Function contents(root as string) As Boolean
    Dim fileName As String
    Dim iHandle As Integer
    Dim rootRoot As String
```

```

        rootRoot = Mid(root, 0, len(root)-1)
        iHandle = FindFirstFile(root, fileName)) Then
        If iHandle = 0 Then
            Print("No files found at "" + root + "")
        Else
            Repeat
                If Is_Dir(rootRoot + fileName) Then
                    Print(""" + rootRoot + fileName + "" is a
directory")

                    if Not(fileName = ".") AND Not(fileName = "..") Then
                        contents(rootRoot + fileName + "¥*")
                    End If
                Else
                    Print(""" + rootRoot + fileName + "" is a file")
                End If
            Until Not(FindNextFile(iHandle, fileName))

            FindClose(iHandle)
        End If
        contents = TRUE
    End Function

Dim i As Integer
ClrScr()
if Argc() = 1 Then
    終了
End If
For i=1 To Argc()-1
    Print(NEWLINE$ + "Recursive List of "" + Argv(i) + "")
    Contents(Argv(i))
Next i

```

Invokation: dmscript sample.dms c:¥* -w_dms

FindNext

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

FindNext 関数では、ディレクトリ スキャンが続行されるため、**FindFirst** で指定した条件に一致する後続のファイルを抽出するためにこの関数を使用できます。**FindNext** では、ディレクトリ内の一致するファイルが検出されなくなるまで、**FindNext** の呼び出しごとに 1 つのファイル名が返されます。

関数の形式

FindNext() as string

FindNext の呼び出しごとに、1 つのファイル名が返されます。ディレクトリ内の一致するファイルが検出されなくなると、空の文字列 ("") が返されます。

例：

```
Dim Filename as string
```

```
Filename=FindFirst("C:¥*.BAT",FA_ANYFILE)
while Filename<>""
    print Filename
    Filename=FindNext()
wend
```

FindNextFile - FindFirstFile ディレクトリ スキャンの続行

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

FindNextFile 関数では、ディレクトリ スキャンが続行されます。

関数の形式

FindNextFile(findHandle as Integer, filename as String) as Integer

findHandle

FindFirstFile で返されたハンドルを識別します。

filename

見つかったファイルの名前を含む出力パラメータを指定します。

この関数では、ファイルが検出された場合、値 1 が返され、検出されない場合、値 0 が返されます。

例：

```
Function contents(root as string) As Boolean
    Dim fileName As String
    Dim iHandle As Integer
    Dim rootRoot As String
```

```

    rootRoot = Mid(root, 0, len(root)-1)
    iHandle = FindFirstFile(root, fileName)) Then
    If iHandle = 0 Then
        Print("No files found at "" + root + "")
    Else
        Repeat
            If Is_Dir(rootRoot + fileName) Then
                Print("" + rootRoot + fileName + "" is a
directory")
                if Not(fileName = ".") AND Not(fileName = "..") Then
                    contents(rootRoot + fileName + "¥")
                End If
            Else
                Print("" + rootRoot + fileName + "" is a file")
            End If
        Until Not(FindNextFile(iHandle, fileName))

        FindClose(iHandle)
    End If
    contents = TRUE
End Function

Dim i As Integer
ClrScr()
if Argc() = 1 Then
    終了
End If
For i=1 To Argc()-1
    Print(NEWLINE$ + "Recursive List of "" + Argv(i) + """)
    Contents(Argv(i))
Next i

```

Invokation: dmscript sample.dms c:¥* -w_dms

GetDir または GetDirectory - 現行ディレクトリ名の取得

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

GetDir または GetDirectory 関数では、現行ディレクトリ名が取得されます。

関数の形式

```
GetDir(n1 as Integer) as String
```

```
GetDirectory(n1 as Integer) as String
```

n1

ドライブ番号を示します。

正常に終了した場合、この関数では、指定ドライブの現行ディレクトリ名を含む文字列が返されます。それ以外の場合は、空の文字列が返されます。

この関数では、パラメータ **n1** (ドライブ番号を示します。デフォルトは **0**、**A** は **1**、**B** は **2** など) により示されるドライブの現在の作業ディレクトリの名前が返されます。

注： UNIX プラットフォームでは、「**n1**」には意味がありません。

例：

この例ではディレクトリが作成され、そのディレクトリに **Active Directory** が設定された後、そのディレクトリの名前が取得されます。この例では、デフォルトディレクトリが **C:¥** に設定され、作成したディレクトリの名前が変更されます。このディレクトリが存在している（つまり、名前が変更されていない）場合は、削除されます。

```
'Create dir
CreateDirectory("C:¥AMTest")

'Set active dir
SetDirectory("C:¥AMTest")
MessageBox(" Current Directory is : " + GetDirectory(0) )
```

```
'Set Default to C:¥  
SetDirectory("C:¥")  
  
'Rename the directory  
RenameDirectory("C:¥AMTest","C:¥AMTest_1")  
  
If ExistDirectory("C:¥AMTest_1") THEN  
    RemoveDirectory("C:¥AMTest_1")  
  
Else  
    MessageBox("Could not find directory : C:¥AMTest_1")  
Endif
```

GetDiskFree、GetDiskFreeB、GetDiskFreeKB、および GetDiskFreeMB - 空きディスク領域を返す

Symbian OS、Windows、および Windows CE で有効です。

関数の形式

```
GetDiskFree(drivenumbl as Integer) as Integer  
GetDiskFreeB(drivenumbl as Integer) as Integer  
GetDiskFreeKB(drivenumbl as Integer) as Integer  
GetDiskFreeMB(drivenumbl as Integer) as Integer  
drivenumbl
```

ドライブ番号を示します。デフォルトには 0、A には 1、B には 2 などのように使用します。

この関数では、32 ビットの整数形式でサイズが返されます。大きいディスクはこの表示能力を超えるため、切り捨てが行われて不正なサイズや負の値が表示されることがあります。

各関数では、以下のように、空きディスク領域を示す整数が返されます。

GetDiskFree

空きディスク領域 (KB)。

GetDiskFreeB

空きディスク領域 (バイト)。

GetDiskFreeKB

空きディスク領域 (KB)。

GetDiskFreeMB

空きディスク領域 (MB)。

注：GetFileStoreFree 関数が、移植可能であり、GetDiskFree 関数よりも優先して使用してください。

例：

```
Rem
```

```
Rem This displays all information about the current drive in KiloBytes
```

```
Rem
```

```
Dim drive AS Integer
```

```
drive = GetDrive()
```

```
Print("Current drive is : " + Str( drive ) )
```

```
Print("Current drive Size : " + Str( GetDiskSizeKB( drive )) + " KB"))
```

```
Print("Current drive Free : " + Str( GetDiskFreeKB( drive ))
```

GetDiskSize、GetDiskSizeB、GetDiskSizeKB、および GetDiskSizeMB - ディスク サイズを返す

Symbian OS、Windows、および Windows CE で有効です。

GetDiskSize 関数では、ディスク サイズが KB で返されます。GetDiskSizeB 関数では、ディスク サイズがバイトで返されます。GetDiskSizeKB 関数では、ディスク サイズが KB で返されます。GetDiskSizeMB 関数では、ディスク サイズが MB で返されます。

関数の形式

```
GetDiskSize(drivenumb1 as Integer) as Integer
```

```
GetDiskSizeB(drivenumb1 as Integer) as Integer
```

```
GetDiskSizeKB(drivenumb1 as Integer) as Integer
```

```
GetDiskSizeMB(drivenumb1 as Integer) as Integer
```

drivenumb1

ドライブ番号を示します。デフォルトには 0、A には 1、B には 2 などのように使用します。

この関数では、32 ビットの整数形式でサイズが返されます。大きいディスクはこの表示能力を超えるため、切り捨てが行われて不正なサイズや負の値が表示されることがあります。

各関数では、以下のように、ディスク サイズを示す整数が返されます。

GetDiskSize

ディスク サイズの値 (KB)。

GetDiskSizeB

ディスク サイズの値 (バイト)。

GetDiskSizeKB

ディスク サイズの値 (KB)。

GetDiskSizeMB

ディスク サイズの値 (MB)。

注：GetDiskSize 関数の代わりに、移植可能な GetFileStoreSize 関数を使用してください。

例：

```
Dim Size,Used,Free as integer
Dim out as string

Size=GetDiskSize(3)
if (Size<>-1) then
    out="Total size: "+chr(9)+str(Size)+" bytes"+ chr(10)
    Used=GetDiskUsage(3)
    out=out+"Bytes used: "+chr(9)+str(Used)+" bytes"+ chr(10)
    Free=GetDiskFre
```

GetDiskUsage、GetDiskUsageB、GetDiskUsageKB、および GetDiskUsageMB - ディスク使用量を返す

Symbian OS、Windows、および Windows CE で有効です。

GetDiskUsage および **GetDiskUsageKB** 関数では、ディスク使用量が **KB** で返されます。**GetDiskUsageB** 関数では、ディスク使用量がバイトで返されます。**GetDiskUsageMB** 関数では、ディスク使用量が **MB** で返されます。

関数の形式

```
GetDiskUsage(drivenumb1 as Integer) as Integer
```

```
GetDiskUsageB(drivenumb1 as Integer) as Integer
```

```
GetDiskUsageKB(drivenumb1 as Integer) as Integer
```

```
GetDiskUsageMB(drivenumb1 as Integer) as Integer
```

drivenumb1

ドライブ番号を示します。

この関数では、**32** ビットの整数形式でサイズが返されます。大きいディスクは表示できる上限を超えるため、切り捨てが行われて不正なサイズや負の値が表示されることがあります。

ドライブは、**drivenumb1** パラメータ（ドライブ番号を示します。デフォルトは **0**、**A** は **1**、**B** は **2** など）により示されます。

各関数では、以下のように、ディスク使用量を示す整数が返されます。

GetDiskUsage

ディスク使用率の値（KB）を指定します。

GetDiskUsageB

ディスク使用率の値（バイト）を指定します。

GetDiskUsageKB

ディスク使用率の値（KB）を指定します。

GetDiskUsageMB

ディスク使用率の値（バイト）を指定します。

注： 移植可能な **GetFileStore** 関数を使用してください。

例：

```
Dim drive AS Integer
drive = GetDrive()
Print("Current drive is : " + Str( drive ))
Print("Current drive Size : " + Str( GetDiskSizeMB( drive )) + " MB")
Print("Current drive Free : " + Str( GetDiskFreeMB( drive )) +
```

GetDrive - 現在のドライブ番号を返す

Symbian OS および Windows で有効です。

GetDrive 関数では、現在のドライブ番号が、A は 1、B は 2、C は 3 などのように返されます。

関数の形式

GetDrive() as Integer

この関数では、現在のドライブ番号が、A は 1、B は 2、C は 3 などのように返されます。

例：

```
Dim drive AS Integer

drive = GetDrive()

Print("Current drive is : " + Str( drive ) )
Print("Current drive Size : " + Str( GetDiskSize( drive ) ) )
Print("Current drive Free : " + Str( GetDiskFree( drive ) ) )
Print("Current drive Usage: " + Str( GetDiskUsage( drive ) ) )
```

GetFileAttributes - ファイル属性を返す

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

GetFileAttributes 関数では、filename で示されたファイルまたはディレクトリの属性が取得されます。

関数の形式

```
GetFileAttributes(filename as String) as Integer
```

filename

ファイルパスを識別します。

すべての Windows プラットフォームでサポートされるファイル属性値を以下のリストに示します。パラメータ値は 10 進数です。

READONLY

値 : 1

HIDDEN

値 : 2

SYSTEM

値 : 4

DIRECTORY

値 : 16

ARCHIVE

値 : 32

NORMAL

値 : 128

TEMPORARY

値： 256

COMPRESSED

値： 2048

OFFLINE

値： 4096

正常に終了した場合、この関数では、指定したファイルまたはディレクトリの属性を含む値が返されます。ファイルが存在しない場合、この関数では、値 -1 が返されます。

例：

この例では **AUTOEXEC.BAT** の属性がレポートされ、[非表示] フラグの切り替えが行われます。

```
dim attr as integer
dim out as string

attr=GetFileAttributes("C:\BAT")
if (attr<>-1) then
    out=out+"Attributes: "+chr(9)+"["
    if attr and FA_ARCHIVE then out=out+"A"
    if attr and FA_SYSTEM then out=out+"S"
    if attr and FA_HIDDEN then out=out+"H"
    if attr and FA_RDONLY then out=out+"R"

    out=out+"]"
    MsgBox("Information on C:\AUTOEXEC.BAT"+chr(10)+chr(10)+out)
else
    MsgBox("Cannot find C:\AUTOEXEC.BAT")
end if
if attr and FA_HIDDEN then
    attr = attr ? FA_HIDDEN
    SetFileAttributes("C:\AUTOEXEC.BAT", attr)
else
    attr = attr + FA_HIDDEN
    SetFileAttributes("C:\AUTOEXEC.BAT", attr)
End If
end:
```

GetFileSize - ファイル サイズを返す

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

GetFileSize 関数では、ファイル サイズがバイトで示されます。

関数の形式

```
GetFileSize(filename as String) as Integer
```

filename

ファイルパスを識別します。

戻り値はバイトで示されます。

例：

この例では、ファイル AUTOEXEC.BAT のサイズと作成日がレポートされます。

```
dim size as integer
dim time as DateTime
dim out as string

size=GetFileSize("C:¥AUTOEXEC.BAT")
if (size<>-1) then
    out="File size: "+chr(9)+chr(9)+str(size)+" bytes"+ chr(10)
    time=GetFileTime("C:¥AUTOEXEC.BAT")
    out=out+"Creation date: "+chr(9)+str(time.month)+"/"+str(time.day)+"/"+str(time.year)
    out=out+" "+str(time.hour)+":"+str(time.minute)+" "
    MsgBox("Information on C:¥AUTOEXEC.BAT"+chr(10)+chr(10)+out)
else
    MsgBox("Cannot find C:¥AUTOEXEC.BAT")
end if
end:
```

GetFileStoreFree、GetFileStoreSize、および GetFileStoreUsage - 空き容量、総容量、および使用容量を返す

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

GetFileStoreFree、GetFileStoreSize、および GetFileStoreUsage 関数では、ファイルシステムの空き容量、総容量、および使用容量がそれぞれ KB で示されます。

関数形式：

```
GetFileStoreFree(filename as String) as Integer
```

```
GetFileStoreSize(filename as String) as Integer
```

```
GetFileStoreUsage(filename as String) as Integer
```

filename

ファイルまたはディレクトリのパスを識別し、ファイルシステムを識別します。ファイルシステムは、UNIX の通常の意味で使用されます。Windows では、ドライブです。

各関数では、KB で量が返されます。

例：

この例のリストでは、ドライブ **C:¥** のサイズ、使用容量および空き容量が示されます。

```
Dim size, free, used As Integer
Dim TAB As Char
TAB = 0x0009
ClrScr()

size = GetFileStoreSize("c:¥")
free = GetFileStoreFree("c:¥")
used = GetFileStoreUsage("c:¥")
Print("Capacity of drive c:¥")
Print("Total size" + TAB + TAB + Str(size))
Print("Free storage" + TAB + Str(free))
Print("Used storage" + TAB + Str(used))
```

GetFileTime - ファイル作成日、最終アクセス日、または最終変更日を返す

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

GetFileTime 関数では、ファイルの作成日が返される（形式 1）か、または時間のタイプが指定されている場合（形式 2）は、作成日、最終アクセス日、または最終変更日が返されます。

関数の形式

```
GetFileTime(filename as String) as DateTime
```

```
GetFileTime(filename as String, type as Integer) as DateTime
```

filename

調査するファイルの名前を指定します。

type

取得するファイル時間のタイプ。

有効な値は以下のとおりです。

- 0 作成日時。
- 1 最終アクセス日時。
- 2 最終変更日時（デフォルト）。

DateTime 構造体は以下のように定義されています。

```
Type DateTime
  Year as Integer
  Month as Integer
  Day as Integer
  DayOfWeek as Integer
  Hour as Integer
  Minute as Integer
  Second as Integer
End Type
```

正常に終了した場合、この関数では、ファイルの作成日時（形式 1）か、または指定されているタイプの時間（形式 2）が返されます。ファイルが存在しない場合、すべてのメンバに -1 が設定された **DateTime** 構造体が返されます。

例 :

```
dim size as integer
dim time as DateTime
dim out as string

size=GetFileSize("C:¥AUTOEXEC.BAT")
if (size<>-1) then
    out="File size: "+chr(9)+chr(9)+str(size)+" bytes"+ chr(10)
    time=GetFileTime("C:¥AUTOEXEC.BAT")
    out=out+"Creation date:
"+chr(9)+str(time.month)+"/"+str(time.day)+"/"+str(time.year)
    out=out+" "+str(time.hour)+":"+str(time.minute)+" "
    MsgBox("Information on C:¥AUTOEXEC.BAT"+chr(10)+chr(10)+out)

else
    MsgBox("Cannot find C:¥AUTOEXEC.BAT")
end if
end:
```

GetLongFileName - 短縮ファイル名をロングファイル名に変換する**Windows で有効**

GetLongFileName 関数では、短縮ファイル名がロングファイル名に変換されます。

関数の形式

GetLongFileName(shortFileName as string, longFileName as string) as Boolean

shortFileName

入力パラメータ。ロングファイル名形式に拡張される MS DOS 8.3 ファイル名です。

longFileName

出力パラメータ。拡張されたロングファイル名を受け取ります。

この関数では、正常に終了した場合、TRUE が返され、それ以外の場合、FALSE が返されます。ファイル名を変換するには、関連するファイルがシステムに存在している必要があります。それ以外の場合は、この関数が失敗します。

例：

この例では、ロング ファイル名が短縮名に変換された後、ロング ファイル名に戻されます。

```
Dim lName, oName, sName As String
ClrScr()

oName = "c:\temp\%a bb ccc dddd\abcdefghijklmnopqrstuvwxy.txt"
If Not(GetShortFileName(oName, sName)) Then
    MessageBox("Can not convert to short", "DMS", MB_OK + MB_ICONEXCLAMATION)
    SetStatus(1)
    終了
End If
Print( oName + " - " + sName)
If Not(GetLongFileName(sName, lName)) Then
    MessageBox("Can not convert to long", "DMS", MB_OK + MB_ICONEXCLAMATION)
    SetStatus(2)
    終了
End If
Print( sName + " - " + lName)
```

GetShortFileName - ロング ファイル名を短縮ファイル名に変換する

Symbian OS および **Windows** で有効です。

GetShortFileName 関数では、ロング ファイル名が短縮ファイル名に変換されます。

関数の形式

```
GetShortFileName(longFileName as string, shortFileName as string) as Boolean
```

longFileName

これは入力パラメータです。短縮ファイル名（8.3 表記法）に変換されるファイル名です。

shortFileName

これは出力パラメータです。変換されたファイル名を受け取ります。

この関数では、正常に終了した場合、**TRUE** が返され、それ以外の場合、**FALSE** が返されます。ファイル名を変換するには、関連するファイルがシステムに存在している必要があります。それ以外の場合は、この関数が失敗します。

例：

この例では、ロング ファイル名が短縮名に変換された後、ロング ファイル名に戻されます。

```
Dim lName, oName, sName As String
ClrScr()

oName = "c:\temp\%a bb ccc dddd\abcdefghijklmnopqrstuvwxy.txt"
If Not(GetShortFileName(oName, sName)) Then
    MsgBox("Can not convert to short", "DMS", MB_OK + MB_ICONEXCLAMATION)
    SetStatus(1)
    終了
End If
Print( oName + " - " + sName)
If Not(GetLongFileName(sName, lName)) Then
    MsgBox("Can not convert to long", "DMS", MB_OK + MB_ICONEXCLAMATION)
    SetStatus(2)
    終了
End If

Print( sName + " - " + lName)
```

GetSpecialDir - 特定ディレクトリ名およびパスの取得

Windows および Windows CE で有効です。

GetSpecialDir 関数では、特定ディレクトリの名前およびパスが取得されます。

関数の形式

```
GetSpecialDir(Id as integer, Path as string) as integer
```

Id

特定ディレクトリの識別子。

path

指定ディレクトリのドライブおよびパス名（出力パラメータ）。

この関数では、正常に終了した場合、ゼロ（0）が返され、それ以外の場合、ゼロ以外の値が返されます。

注：CSIDL パラメータにより指定されるロケーションが、ファイル システムの一部ではない場合、この関数は失敗します。

例：

```
Rem sample for special directories

Dim dirName As String
ClrScr()

If GetSpecialDir(CSIDL_APPDATA, dirName) <> 0 Then
    MsgBox("Retrieve CSIDL_APPDATA failed!", "DMS", MB_OK + MB_ICONEXCLAMATION)
    SetStatus(1)
    Exit
End If

Print("CSIDL_APPDATA: "" + dirName + """)
```

Is_Archive - ファイルがアーカイブされているかどうかの確認

NetWare、Windows、および Windows CE で有効

Is_Archive 関数では、指定の属性についてファイルが確認されます。

関数の形式

Is_Archive(filename as String) as Boolean

filename

ファイルパスを識別します。

ファイルがアーカイブされている場合、この関数では **TRUE** が返され、それ以外の場合、**FALSE** が返されます。

例：

```
If Is_Archive(Argv(1)) Then
    print(Argv(1) + " indicates 'Archive'.")
Else
    print(Argv(1) + " does not indicate 'Archive'.")
End If
```

Is_Blk - ファイルがブロック スペシャル ファイルであるかどうかの確認

UNIX でのみ有効です。

Is_Blk 関数では、指定の属性についてファイルが確認されます。

関数の形式

Is_Blk(filename as String) as Boolean

filename

ファイルパスを識別します。

ファイルがブロック スペシャル ファイルである場合、TRUE が返され、その他の場合はすべて FALSE が返されます。

例 :

```
If Is_Blk(Argv(1)) Then
    print(Argv(1) + " indicates 'Blk'.")
Else
    print(Argv(1) + " does not indicate 'Blk'.")
End If
```

Is_Chr - ファイルがキャラクタ スペシャル ファイルであるかどうかの確認

UNIX でのみ有効です。

Is_Chr 関数では、指定の属性についてファイルが確認されます。

関数の形式

Is_Chr(filename as String) as Boolean

filename

ファイルパスを識別します。

ファイルがキャラクタ スペシャル ファイルである場合、TRUE が返され、それ以外の場合、FALSE が返されます。

例：

```
If Is_Chrg(Argv(1)) Then
    print(Argv(1) + " indicates 'Chr'.")
Else
    print(Argv(1) + " does not indicate 'Chr'.")
End If
```

Is_Dir - ファイルがディレクトリであるかどうかの確認

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

Is_Dir 関数では、指定の属性についてファイルが確認されます。

関数の形式

Is_Dir(filename as String) as Boolean

filename

ファイルパスを識別します。

ファイル名およびディレクトリ名に対して正しい構文を使用してください。たとえば、**C:¥WINNT** はファイル名であり、ディレクトリ名ではありません。ディレクトリに対する構文は、**C:¥WINNT¥*** です。

ファイルがディレクトリである場合、この関数では **TRUE** が返され、それ以外の場合、**FALSE** が返されます。

例：

```
If Is_Dir(Argv(1)) Then
    print(Argv(1) + " indicates 'Dir'.")
Else
    print(Argv(1) + " does not indicate 'Dir'.")
End If
```

Is_Fifo - ファイルに「パイプ スペシャル」属性があるかどうかの確認

UNIX でのみ有効です。

Is_Fifo 関数では、指定の属性についてファイルが確認されます。

関数の形式

Is_Fifo(filename as String) as Boolean

filename

ファイルパスを識別します。

ファイルがパイプ スペシャル ファイルである場合、この関数では **TRUE** が返され、それ以外の場合、**FALSE** が返されます。

例 :

```
If Is_Fifo(Argv(1)) Then
    print(Argv(1) + " indicates 'Fifo'.")
Else
    print(Argv(1) + " does not indicate 'Fifo'.")
End If
```

Is_File - ファイルが通常のファイルであるかどうかの確認

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。.

Is_File 関数では、指定の属性についてファイルが確認されます。

関数の形式

Is_File(filename as String) as Boolean

filename

ファイルパスを識別します。

ファイルが通常のファイルである場合、この関数では **TRUE** が返され、それ以外の場合、**FALSE** が返されます。

例：

```
If Is_File(Argv(1)) Then
    print(Argv(1) + " indicates 'File'.")
Else
    print(Argv(1) + " does not indicate 'File'.")
End If
```

Is_G_R - ファイルに「グループ読み取り」属性があるかどうかの確認

UNIX でのみ有効です。

Is_G_R 関数では、指定の属性（グループ読み取り）についてファイルが確認されます。

関数の形式

Is_G_R(filename as String) as Boolean

filename

ファイルパスを識別します。

属性が設定されている場合、この関数では **TRUE** が返されます。設定されていない場合は **FALSE** が返されます。

例：

```
If Is_G_R(Argv(1)) Then
    print(Argv(1) + " indicates 'G_R'.")
Else
    print(Argv(1) + " does not indicate 'G_R'.")
End If
```

Is_G_W - ファイルに「グループ書き込み」属性があるかどうかの確認

UNIX でのみ有効です。

Is_G_W 関数では、指定の属性（グループ書き込み）についてファイルが確認されます。

関数の形式

Is_G_W(filename as String) as Boolean

filename

ファイルパスを識別します。

属性が設定されている場合、この関数では **TRUE** が返されます。設定されていない場合は **FALSE** が返されます。

例：

```
If Is_G_W(Argv(1)) Then
    print(Argv(1) + " indicates 'G_W'.")
Else
    print(Argv(1) + " does not indicate 'G_W'.")
End If
```

Is_G_X - ファイルに「グループ実行」属性があるかどうかの確認

UNIX でのみ有効です。

Is_G_X 関数では、指定の属性（グループ実行）についてファイルが確認されます。

関数の形式

Is_G_X(filename as String) as Boolean

name

ファイルパスを識別します。

属性が設定されている場合、この関数では **TRUE** が返されます。設定されていない場合は **FALSE** が返されます。

例：

```
If Is_G_X(Argv(1)) Then
    print(Argv(1) + " indicates 'G_X'.")
Else
    print(Argv(1) + " does not indicate 'G_X'.")
End If
```

Is_Hidden - ファイルに「隠しファイル」属性があるかどうかの確認

Windows で有効

Is_Hidden 関数では、指定の属性についてファイルが確認されます。

関数の形式

Is_Hidden(filename as String) as Boolean

filename

ファイルパスを識別します。

ファイルが隠しファイルである場合、この関数では **TRUE** が返され、それ以外の場合、**FALSE** が返されます。

例：

```
If Is_Hidden(Argv(1)) Then
    print(Argv(1) + " indicates 'Hidden'.")
Else
    print(Argv(1) + " does not indicate 'Hidden'.")
End If
```

Is_O_R - ファイルに「他者読み取り」属性があるかどうかの確認

UNIX でのみ有効です。

Is_O_R 関数では指定の属性（他者読み取り）についてファイルが確認されます。

関数の形式

Is_O_R(filename as String) as Boolean

filename

ファイルパスを識別します。

属性が設定されている場合、この関数では **TRUE** が返されます。設定されていない場合は **FALSE** が返されます。

例：

```
If Is_O_R(Argv(1)) Then
    print(Argv(1) + " indicates 'O_R'.")
Else
    print(Argv(1) + " does not indicate 'O_R'.")
End If
```

Is_O_W - ファイルに「他者書き込み」属性があるかどうかの確認

UNIXでのみ有効です。

Is_O_W 関数では指定の属性（他者書き込み）についてファイルが確認されます。

関数の形式

Is_O_W(filename as String) as Boolean

filename

ファイルパスを識別します。

属性が設定されている場合、この関数では **TRUE** が返されます。設定されていない場合は **FALSE** が返されます。

例：

```
If Is_O_W(Argv(1)) Then
    print(Argv(1) + " indicates 'O_W'.")
Else
    print(Argv(1) + " does not indicate 'O_W'.")
End If
```

Is_O_X - ファイルに「他者実行」属性があるかどうかの確認

UNIXでのみ有効です。

Is_O_X 関数では指定の属性（他者実行）についてファイルが確認されます。

関数の形式

Is_O_X(filename as String) as Boolean

filename

ファイルパスを識別します。

属性が設定されている場合、この関数では **TRUE** が返されます。設定されていない場合は **FALSE** が返されます。

例：

```
If Is_O_X(Argv(1)) Then
    print(Argv(1) + " indicates 'O_X'.")
Else
    print(Argv(1) + " does not indicate 'O_X'.")
End If
```

Is_ReadOnly - ファイルに「読み取り専用」属性があるかどうかの確認

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

Is_ReadOnly 関数では、指定の属性についてファイルが確認されます。

関数の形式

Is_ReadOnly(filename as String) as Boolean

filename

ファイルパスを識別します。

属性が設定されている場合、この関数では **TRUE** が返されます。設定されていない場合は **FALSE** が返されます。

例：

```
If Is_ReadOnly(Argv(1)) Then
    print(Argv(1) + " indicates 'ReadOnly'.")
Else
    print(Argv(1) + " does not indicate 'ReadOnly'.")
End If
```

Is_SGID - ファイルに「グループ ID 設定」属性があるかどうかの確認

UNIX でのみ有効です。

Is_SGID 関数では、指定の属性（グループ ID 設定）についてファイルが確認されます。

関数の形式

Is_SGID(filename as String) as Boolean

filename

ファイルパスを識別します。

属性が設定されている場合、この関数では **TRUE** が返されます。設定されていない場合は **FALSE** が返されます。

例：

```
If Is_SGID(Argv(1)) Then
    print(Argv(1) + " indicates 'SGID'.")
Else
    print(Argv(1) + " does not indicate 'SGID'.")
End If
```

Is_SUID - ファイルに「ユーザ ID 設定」属性があるかどうかの確認

UNIX でのみ有効です。

Is_SUID 関数では、指定の属性（ユーザ ID 設定）についてファイルが確認されます。

関数の形式

Is_SUID(filename as String) as Boolean

filename

ファイルパスを識別します。

属性が設定されている場合、この関数では **TRUE** が返されます。設定されていない場合は **FALSE** が返されます。

例：

```
If Is_SUID(Argv(1)) Then
    print(Argv(1) + " indicates 'SUID'.")
Else
    print(Argv(1) + " does not indicate 'SUID'.")
End If
```

Is_System - ファイルに「システム ファイル」属性があるかどうかの確認

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

Is_System 関数では、指定の属性（システム ファイル）についてファイルが確認されます。

関数の形式

Is_System(filename as String) as Boolean

filename

ファイルパスを識別します。

属性が設定されている場合、この関数では **TRUE** が返されます。設定されていない場合は **FALSE** が返されます。

例：

```
If Is_System(Argv(1)) Then
    print(Argv(1) + " indicates 'System'.")
Else
    print(Argv(1) + " does not indicate 'System'.")
End If
```

Is_U_R - ファイルに「ユーザ読み取り」属性があるかどうかの確認

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

Is_U_R 関数では、指定の属性（ユーザ読み取り）についてファイルが確認されます。

関数の形式

Is_U_R(filename as String) as Boolean

filename

ファイルパスを識別します。

属性が設定されている場合、この関数では **TRUE** が返されます。設定されていない場合は **FALSE** が返されます。

例：

```
If Is_U_R(Argv(1)) Then
    print(Argv(1) + " indicates 'U_R'.")
Else
    print(Argv(1) + " does not indicate 'U_R'.")
End If
```

Is_U_W - ファイルに「ユーザ書き込み」属性があるかどうかの確認

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

Is_U_W 関数では、指定の属性（ユーザ書き込み）についてファイルが確認されます。

関数の形式

Is_U_W(filename as String) as Boolean

filename

ファイルパスを識別します。

属性が設定されている場合、この関数では **TRUE** が返されます。設定されていない場合は **FALSE** が返されます。

例：

```
If Is_U_W(Argv(1)) Then
    print(Argv(1) + " indicates 'U_W'.")
Else
    print(Argv(1) + " does not indicate 'U_W'.")
End If
```

Is_U_X - ファイルに「ユーザ実行」属性があるかどうかの確認

UNIX でのみ有効です。

Is_U_X 関数では、指定の属性（ユーザ実行）についてファイルが確認されます。

関数の形式

Is_U_X(filename as String) as Boolean

filename

ファイルパスを識別します。

属性が設定されている場合、この関数では **TRUE** が返されます。設定されていない場合は **FALSE** が返されます。

例：

```
If Is_U_X(Argv(1)) Then
    print(Argv(1) + " indicates 'U_X'.")
Else
    print(Argv(1) + " does not indicate 'U_X'.")
End If
```

MkDir - ファイル システム関数

MkDir は、CreateDirectory 関数のエイリアスです。

関連項目：

[CreateDirectory または MkDir - 新規ディレクトリの作成](#) (P. 84)

RemoveDirectory または Rmdir - ディレクトリの削除

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

RemoveDirectory または Rmdir 関数では、**pathstring** で指定されたディレクトリが削除されます。

注： 指定したディレクトリは、書き込み保護されていない必要があります。

関数の形式

```
RemoveDirectory(pathstring as String) as Boolean
```

```
Rmdir(pathstring as String) as Boolean
```

pathstring

削除対象のディレクトリを識別します。 指定するディレクトリには、以下の条件が必要です。

- 空である必要があります。
- 現在の作業ディレクトリではない必要があります。
- ルート ディレクトリではない必要があります。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

```
'Create dir
CreateDirectory("C:¥AMTest")

'Set active dir
SetDirectory("C:¥AMTest")
MessageBox(" Current Directory is : " + GetDirectory(0) )

'Set Default to C:¥
SetDirectory("C:¥")

'Rename the directory
RenameDirectory("C:¥AMTest","C:¥AMTest_1")
```

```
If ExistDirectory("C:¥AMTest_1") THEN
    RemoveDirectory("C:¥AMTest_1")

Else
    MessageBox("Could not find directory : C:¥AMTest_1")

Endif
```

RenameDirectory または RenDir - ディレクトリの名前の変更または移動

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

RenameDirectory または **RenDir** 関数を使用して、指定のディレクトリの名前を変更したり、指定のディレクトリを移動します。

関数の形式

```
RenameDirectory(oldname as String, newname as String) as Boolean
```

```
RenDir(oldname as String, newname as String) as Boolean
```

oldname

既存のディレクトリの名前を指定します。

newname

ディレクトリの新しい名前を示します。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

```
'Create dir
CreateDirectory("C:¥AMTest")

'Set active dir
SetDirectory("C:¥AMTest")
MessageBox(" Current Directory is : " + GetDirectory(0) )

'Set Default to C:¥
SetDirectory("C:¥")

'Rename the directory
RenameDirectory("C:¥AMTest","C:¥AMTest_1")
```

```
If ExistDirectory("C:¥AMTest_1") THEN
    RemoveDirectory("C:¥AMTest_1")

Else
    MessageBox("Could not find directory : C:¥AMTest_1")
Endif
```

RenameFile - ファイルの名前の変更または移動

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

RenameFile または RenFile 関数を使用して、指定のファイルの名前を変更したり、指定のファイルを移動します。

関数の形式

RenFile(oldname as String, newname as String) as Boolean

RenameFile(oldname as String, newname as String) as Boolean

oldname

既存のファイルの名前を指定します。

newname

ファイルの新しい名前を示します。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

HELP_GETFILEATTRIBUTES

```
If Not(ExistDirectory("c:¥backup")) then CreateDirectory("c:¥backup")
CopyFile("c:¥autoexec.bat", "c:¥backup¥autoexec.bat")
RenameFile("c:¥backup¥autoexec.bat", "c:¥backup¥autoexec.bak")
```

SetFileAttributes - ファイル属性の設定

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

SetFileAttributes 関数では、filename で指定されたファイルまたはディレクトリの属性が設定されます。

関数の形式

```
SetFileAttributes(filename as String,attr as Integer) as Boolean
```

filename

ファイルパスを指定します。

attr

設定する値を示します。

Windows

Windows でのファイル属性の値：

READONLY

値： 1

HIDDEN

値： 2

SYSTEM

値： 4

DIRECTORY

値： 16

ARCHIVE

値： 32

NORMAL

値： 128

TEMPORARY

値： 256

COMPRESSED

値： 2048

OFFLINE

値： 4096

例：

この例では AUTOEXEC.BAT の属性がレポートされ、[非表示] フラグの切り替えが行われます。

```
dim attr as integer
dim out as string

attr=GetFileAttributes("C:BAT")
if (attr<>-1) then
    out=out+"Attributes: "+chr(9)+"["
    if attr and FA_ARCHIVE then out=out+"A"
    if attr and FA_SYSTEM then out=out+"S"
    if attr and FA_HIDDEN then out=out+"H"
    if attr and FA_RDONLY then out=out+"R"

    out=out+"]"
    MsgBox("Information on C:¥AUTOEXEC.BAT"+chr(10)+chr(10)+out)
else
    MsgBox("Cannot find C:¥AUTOEXEC.BAT")
end if
if attr and FA_HIDDEN then
    attr = attr ? FA_HIDDEN
    SetFileAttributes("C:¥AUTOEXEC.BAT", attr)
else
    attr = attr + FA_HIDDEN
    SetFileAttributes("C:¥AUTOEXEC.BAT", attr)
End If
end:
```

SetFileTime - ファイルまたはディレクトリのタイムスタンプ設定

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

SetFileTime 関数では、ファイルまたはディレクトリのタイムスタンプが設定されます。

関数の形式

```
SetFileTime(fileName as string, dateAndTime as DateTime, Type as integer) as Boolean
```

```
SetFileTime(fileName as string, dateAndTime as DateTime) as Boolean
```

fileName

日時を変更するファイルの名前を指定します。

dateAndTime

新しく設定する日時を指定します。

タイプ

値 0：作成日時が変更されます（デフォルト）。

値 1：ファイルへの最終アクセス日時が変更されます。

値 2：ファイルが最後に変更された日時が変更されます。

値 3：作成日時、最終アクセス日時、および最終書き込み日時が変更されます。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

```
Rem modify write time
Dim newTime as DateTime
newTime.year = 1999
newTime.month = 12
newTime.day = 31
newTime.hour = 23
newTime.minute = 59
newTime.second = 59
If Not(SetFileTime("c:\%autoexec.bat",newTime,2)) Then
    Print ("SetFileTime failed")
    Exit
End If
```

ファイル バージョンの読み取りと比較

DMScript 関数を使用して、任意のファイルのバージョン情報を読み取って比較できます。たとえば、**dmscript.exe** のバージョンを読み取って比較し、スクリプトが適切な DMScript のバージョン上で実行されているかどうかを確認できます。

GetFileInfo

GetFileInfo 関数は、指定されたファイルのファイル情報を読み取ります。CompareVersions など、取得されたファイル情報を別の関数で使用できます。

この関数の形式は、以下のとおりです。

GetFileInfo (option as string, filename as string, result as string) as boolean

例 : GetFileInfo

```
getfileinfo("version","dmscript.exe",version)
```

入力パラメータ

この関数には、以下の入力パラメータがあります。

オプション

取得するファイル情報を指定します。以下のいずれかのオプションを指定できます。

"version"

完全なファイルバージョン文字列を、major.minor.patch.build 形式で取得します。例 : "12.5.0.1234"

"major"

ファイルのメジャーバージョン番号を取得します。

"minor"

ファイルのマイナーバージョン番号を取得します。

"patch"

ファイルのパッチ番号を取得します。

"build"

ファイルのビルド番号を取得します。

"desc"

ファイル記述 (Windows のみ) を取得します。

"ext"

ファイル拡張子を取得します。

"filename"

拡張子のないパス名からファイル名を取得します。

結果

関数によって返される結果を保持する文字列変数を指定します。

戻り値

関数は、ファイル情報を取得できる場合には **true** を返します。

CompareVersions

CompareVersions 関数は 2 つのファイルのバージョンを比較し、それらが同一かどうかを返します。この関数の形式は、以下のとおりです。

```
CompareVersions (version1 as string, version2 as string) as integer
```

例 : CompareVersions

```
CompareVersions ("12.5.1.4", "12.5.0.3")
```

入力パラメータ

この関数には、以下の入力パラメータがあります。

version1 と version2

比較するファイルのバージョンを指定します。バージョン文字列に、任意の数のバージョン識別項目を指定できます。たとえば、**major.minor.patch.build** 形式または単に **major.minor** 形式を使用して、バージョン文字列を指定できます。バージョン番号は、ドット、スペース、またはカンマによって分離できます。1 つのバージョン文字列に別のバージョン文字列より多くの識別項目が存在する場合、存在しない項目は比較時にゼロで置換されます。たとえば、**CompareVersions ("12.5.180.123", "12.5")** は **CompareVersions ("12.5.180.123", "12.5.0.0")** として扱われます。

戻り値

この関数は、以下のいずれかの事前定義済み整数定数を返します。

ECVEQUAL (0)

バージョンが等しいことを示します。

ECVGREATERTHAN (1)

バージョン 1 がバージョン 2 より大きいことを示します。

ECVLESSTHAN (-1)

バージョン 1 がバージョン 2 より小さいことを示します。

ECVINVALID (2)

1 つまたは両方のバージョン文字列が空白か、または非数値の値が含まれていることを示します。

アイコン関数

AddItem - アイコンの追加

Windows でのみ有効です。

AddItem 関数は、既存グループにアイコンを追加します。

関数の形式

```
AddItem(  
    title as String,  
    cmdline as String,  
    iconfile as String,  
    workdir as String,  
    group as String,  
    min as Boolean) as Boolean
```

title

[グループ] ウィンドウ内のアイコンの下に表示されるタイトルを指定します。

cmdline

アプリケーションを実行するために必要な完全なコマンドラインを指定します。このパラメータにはアプリケーションの実行可能ファイルの名前を設定する必要がありますが、アプリケーションの完全パスや、アプリケーションに必要なパラメータを含めることもできます。

iconfile

[グループ] ウィンドウに表示するアイコンのファイル名を指定します。このファイルは、実行可能ファイルまたはアイコンファイルになります。このパラメータが空の文字列の場合は、コマンドライン実行可能ファイル内の最初のアイコンが使用されます。

workdir

デフォルト（または作業中の）ディレクトリの名前を示します。

group

アクティブにするプログラム マネージャ グループを指定します。このパラメータが空の文字列の場合、**Windows** は現在アクティブなグループを使用します。

min

True の場合は、実行可能ファイルが最小化された状態で開始します。この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

AddItem 関数はブール値を返します。

CreateGroup- 新規グループの作成または既存グループの起動

Windows でのみ有効です。

CreateGroup 関数は、新規グループを作成したり、既存グループのウィンドウを起動します。

関数の形式

```
CreateGroup (name as String) as Boolean
```

```
CreateGroup (name as String,  
             offset as Integer) as Boolean
```

name

グループ名を指定します。

offset

以下のいずれかです。

- LNK_PROGRAMS
- LNK_DESKTOP
- LNK_STARTMENU
- LNK_STARTUP
- LNK_ABSOLUTE

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

以下の例では、スクリプトエディタのアイコンを含む **DMS Test** というアイコングループを作成します。また、デスクトップにショートカットを追加します。このアイコングループおよびショートカットは削除されます。

```
Rem
' determine location of dmsedit.exe
Dim hkey, dummy As Integer
Dim dmseditPath As String

hkey = RegOpenKey(HKEY_LOCAL_MACHINE,
"SOFTWARE\ComputerAssociates\DMScript\DMSEdit")
If hKey = 0 Then
    MsgBox("Can not open DMSEdit-key", "Desktop Management Scripting", MB_OK +
    MB_ICONEXCLAMATION)
    SetStatus(1)
    Exit
End If

If Not(RegQueryVariable(hkey, "dmseditInstalledAt", dmseditPath, dummy) =
REG_STRING) Then
    MsgBox("dmseditInstalledAt not found or invalid", "Desktop Management
    Scripting", MB_OK + MB_ICONEXCLAMATION)
    SetStatus(2)
    Exit
End If

RegCloseKey(hkey)

' Now Create group and icons
If Not(CreateGroup("DMS Test", LNK_PROGRAMS)) Then
    MsgBox("Can not create icon group", "Desktop Management Scripting", MB_OK +
    MB_ICONEXCLAMATION)
    SetStatus(3)
    Exit
End If

If Not(AddItem("DMS Editor", dmseditPath+"%dmsedit.exe", "", "", "DMS Test", FALSE,))
Then
    MsgBox("Failed to install dmsedit at icon group", "Desktop Management
    Scripting", MB_OK + MB_ICONEXCLAMATION)
    SetStatus(4)
    Exit
End If

if Not(CreateLink("DMS Editor", dmseditPath+"%dmsedit.exe", "", "", "", LNK_NORMAL,
LNK_DESKTOP)) Then
```

```
        MsgBox("Failed to install dmsedit at desktop", "Desktop Management Scripting",
        MB_OK + MB_ICONEXCLAMATION)
        SetStatus(5)
        Exit
    End If

    MsgBox("Icons and shortcut has been created, press OK to delete them
    again", "Desktop Management Scripting: Icon Confirm", MB_OK + MB_ICONINFORMATION)

    'Delete shortcut
    if Not(DeleteItem("DMS Editor", "", LNK_DESKTOP)) Then
        MsgBox("Failed to deinstall dmsedit from desktop", "Desktop Management
        Scripting", MB_OK + MB_ICONEXCLAMATION)
        SetStatus(6)
        Exit
    End If

    if Not(DeleteGroup("DMS Test", LNK_PROGRAMS)) Then
        MsgBox("Failed to delete icon group", "Desktop Management Scripting", MB_OK
        + MB_ICONEXCLAMATION)
        SetStatus(7)
        Exit
    End If
```

CreateLink - ショートカットの作成

Windows でのみ有効です。

CreateLink 関数はファイルへのショートカットを作成します。

デフォルト フォーマット

CreateLink(Title as string, FileName as string, Arguments as string, WorkDir as string, Group as string, IconFile as string, Style as Integer, Offset as Integer) as Boolean.

その他のフォーマット

CreateLink(Title as string, Filename as string, Arguments as string, Workdir as string, Group as string) as Boolean

iconfile = ""、style = LNK_NORMAL、および offset = LNK_PROGRAMS を指定する以外は、メインフォーマットと同様

CreateLink(Title as string, Filename as string, Arguments as string, Workdir as string, Group as string, Iconfile as string) as Boolean

style = LNK_NORMAL、および offset = LNK_PROGRAMS を指定する以外は、メインフォーマットと同様

CreateLink(Title as string, Filename as string, Arguments as string, Workdir as string, Group as string, Style as integer) as Boolean

iconfile = "" および offset = LNK_PROGRAMS を指定する以外は、メインフォーマットと同様

CreateLink(Title as string, Filename as string, Arguments as string, Workdir as string, Group as string, Style as integer, Offset as integer) as Boolean

iconfile = "" を指定する以外は、メインフォーマットと同じ

CreateLink(Title as string, Filename as string, Arguments as string, Workdir as string, Group as string, Iconfile as string, Style as integer) as Boolean

offset = LNK_PROGRAMS を指定する以外は、メインフォーマットと同様。 Software Delivery スタイルとの互換性のために、True は LNK_MINIMIZE に、FALSE は LNK_NORMAL にマップされています。

CreateLink(Title as string, Filename as string, Arguments as string, Workdir as string, Group as string, Iconfile as string, Iconindex as integer, Style as integer, Offset as integer, Hotkey as word, Description as string) as Boolean

Hotkey は、低次バイトの仮想キー コードおよび高次バイトの修飾子フラグで構成されます。修飾子フラグでは、以下の定数が定義されます。

HOTKEYF_ALT

Alt キー

HOTKEYF_CONTROL

Ctrl キー

HOTKEYF_EXT

拡張キー

HOTKEYF_SHIFT

Shift キー

ホットキー Ctrl + Alt + X を定義するコード

Dim HOTKEYF as Word

hotKey = HHOTKEYF_ALT + HOTKEYF_CONTROL + asc("X")

タイトル

ショートカットのタイトルを指定します。

FileName

ショートカットの参照先となるファイルの名前とパスを指定します。最低でも、パラメータにはファイル名を指定する必要があります。

引数

ファイルに対するオプションの引数またはスイッチを指定します。

WorkDir

デフォルト（または作業中の）ディレクトリの名前を指定します。

Group

ショートカットを追加するグループ(またはフォルダ)を指定します。
このパラメータが空の文字列の場合、**offset** パラメータ値によって
ショートカットの配置場所が決まります。

IconFile

グループ ウィンドウに表示するアイコンのファイル名を指定します。
このファイルは、実行可能ファイルまたはアイコン ファイルになります。
このパラメータが空の文字列であるか、または書略されている場合は、**FileName** で指定されたファイル内の最初のアイコンが使用されます。

Style

ショートカットが起動された場合のファイルの表示法を指定します。
以下の定数のいずれかを使用します。

LNK_NORMAL

通常どおり実行します。

LNK_MAXIMIZE

最大化して実行します。

LNK_MINIMIZE

最小化して実行します。**Style** の指定がない場合は、値 **LNK_NORMAL** が使用されます。

Offset

Group 引数の相対位置を指定します。以下の定数のいずれかを使用します。

LNK_PROGRAMS

項目を **Programs** グループに配置します。

LNK_DESKTOP

項目をデスクトップに配置します。

LNK_STARTMENU

項目をスタートメニューに配置します。

LNK_STARTUP

項目を Startup グループに配置します。

LNK_ABSOLUTE

項目を指定されたディレクトリに配置します。

Iconindex

表示するアイコンのインデックスを指定します。**iconindex** をコード化しない場合、アイコンリストの最初のアイコンが選択されます。

Hotkey

アイコンのホットキーコードを指定します。**0** をコード化した場合、ホットキーは設定されません。

説明

ショートカットに保存される透過的情報を指定します。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。**DOS** 環境で実行した場合、このコマンドは Windows 起動時に実行されるキューに格納されます。このため、戻り値は常に **True** になります。

DeleteGroup - グループの削除

Windows でのみ有効です。

DeleteGroup 関数は、名前で指定されたグループを削除します。

関数の形式

DeleteGroup(name as String) as Boolean

name

削除するグループの名前を指定します。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

DeleteItem - グループからの項目の削除

Windows でのみ有効です。

DeleteItem 関数は、指定グループから指定項目を削除します。

関数の形式

```
DeleteItem(name as String, group as String) as Boolean
```

name

削除する項目の名前を指定します。

group

削除対象の項目を含むグループの名前を指定します。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

RemoveLink - ショートカットの削除

Windows でのみ有効です。

RemoveLink 関数は、ショートカットを削除します。

関数の形式

```
RemoveLink(Name as string, Group as string, Offset as integer) as Boolean
```

名前

削除するショートカットの名前を指定します。

Group

削除するショートカットが配置されたグループ（またはフォルダ）を指定します。パラメータが空の文字列の場合、*offset* パラメータ値によってショートカットの削除場所が決まります。

LNK_PROGRAMS

Programs グループから項目が削除されます。

LNK_DESKTOP

デスクトップから項目が削除されます。

LNK_STARTMENU

スタートメニューから項目が削除されます。

LNK_STARTUP

Startup グループから項目が削除されます。

LNK_ABSOLUTE

指定されたディレクトリから項目が削除されます。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

```
if Not(RemoveLink("DMS Editor", "", LNK_DESKTOP)) Then
    MsgBox("Failed to deinstall dmsedit from desktop", "Desktop Management
Scripting", MB_OK + MB_ICONEXCLAMATION)
    SetStatus(1)
    Exit
End If
```

初期化ファイル (.INI) 関数

Windows NT、Windows 9x、および UNIX プラットフォームでは、初期化 (.ini) ファイル関数を使用できます。

重要： .ini ファイルでは Linux コンピュータ上で日本語を含むことはできません。

DeleteIniEntry - 初期化ファイル内のエントリの削除

Windows でのみ有効です。

DeleteIniEntry 関数は、指定された初期化 (.ini) の指定セクション内の指定エントリを削除します。

関数の形式

DeleteIniEntry(section as String, entry as String, filename as String) as Boolean

section

エントリを含むセクションを指定します。

entry

削除するエントリを指定します。

filename

初期化（.ini）ファイルの名前を指定します。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

```
Rem By deleting the file Ncclient.ini a new workstation name detect
Rem will be forced next time the AM agent is executed.
Rem
DeleteFile(ComputerPath+"Ncclient.ini")
Rem ComputerPath is an Asset Management constant
```

DeleteIniSection - 初期化ファイルのセクションの削除

UNIX プラットフォームで有効

DeleteIniSection 関数は、指定された初期化（.ini）ファイルのエントリ セクションを削除します。

関数の形式

```
DeleteIniSection(section as String, filename as String) as Boolean
```

section

削除するセクションを指定します。

filename

初期化（.ini）ファイルの名前を指定します。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

以下の例では、ファイル c:\temp\test.ini、セクション LIST、およびセクション SINGLE からエントリ Entry を読み取って、出力します。また、セクション LIST、およびセクション SINGLE のエントリ Entry を削除します。

```
Function strTok(in As string, BYREF pos As Integer, token As String, BYREF out As String) As Boolean
    Dim buf As String
    Dim len As Integer

    If (pos > Len(in)) Then
        strTok = FALSE
    Else
        buf = Mid(in, pos)
        len = Instr(buf, token)
        If len = 0 Then
            out = buf
            pos = pos + Len(buf) + 1
        Else
            out = Mid(buf, 0, len-1)

            pos = pos + len + 1
        End If
        strTok = TRUE
    End If
End Function

Dim attrList, str, lf As string
Dim pos As Integer

' clear screen
ClsScr()

If Not(ReadIniSection("LIST", attrList, "c:\temp\test.ini")) Then
    MsgBox("Can not read LIST section", "DMS", MB_OK + MB_ICONEXCLAMATION)
    SetStatus(1)
    終了
End If
Print("[LIST]")
pos = 0
lf = chr(10)
While (strTok(attrList, pos, lf, str))
    Print(str)
Wend

If Not(ReadIniEntry("SINGLE", "Entry", attrList, "c:\temp\test.ini")) Then
    MsgBox("Can not read SINGLE section", "DMS", MB_OK + MB_ICONEXCLAMATION)
```

```
        SetStatus(2)
        終了
    End If

    Print("[SINGLE]" + NEWLINE$ + "Entry=" + attrList)

    If Not(DeleteIniEntry("SINGLE", "Entry", "c:¥temp¥test.ini")) Then
        MsgBox("Can not delete [SINGLE].Entry ", "DMS", MB_OK +
        MB_ICONEXCLAMATION)
        SetStatus(3)
        終了
    End If
    If Not(DeleteIniSection("LIST", "c:¥temp¥test.ini")) Then
        MsgBox("Can not delete LIST section", "DMS", MB_OK + MB_ICONEXCLAMATION)

    SetStatus(4)
    終了
End If
```

ReadIniSection - セクションの取得

UNIX および Windows プラットフォームで有効

ReadIniSection 関数は初期化 (.ini) ファイルからセクション全体を取得します。

関数の形式

```
ReadIniSection( section as String, result as String, filename as String) as Integer
section
```

取得するセクションを指定します。

result

セクションを取得する文字列変数を指定します。

filename

初期化 (.ini) ファイルの名前を指定します。Windows 環境でファイル名のみを指定した場合は、Windows ディレクトリが検索されますが、Windows 以外の環境では現在のディレクトリが検索されます。

ReadIniEntry() および **ReadIniSection()** は、呼び出しに成功した場合、正の値を返します。失敗した場合は、ゼロ（0）を返します。この関数は **section** で指定した名前と一致するセクションを初期化ファイル内で検索し、すべてのエントリを以下の形式で結果ファイルにコピーします。

```
Entry_1 = Value_1 <LF>  
Entry_n = Value_n <LF>
```

例：

```
Function strTok(in As string, BYREF pos As Integer, token As String, BYREF out As
String) As Boolean
    Dim buf As String
    Dim len As Integer

    If (pos > Len(in)) Then
        strTok = FALSE
    Else
        buf = Mid(in, pos)
        len = Instr(buf, token)
        If len = 0 Then
            out = buf
            pos = pos + Len(buf) + 1
        Else
            out = Mid(buf, 0, len-1)

pos = pos + len + 1
        End If
        strTok = TRUE
    End If
End Function

Dim attrList, str, lf As string
Dim pos As Integer

' clear screen
ClsScr()

If Not(ReadIniSection("LIST", attrList, "c:\temp\test.ini")) Then
    MsgBox("Can not read LIST section", "DMS", MB_OK + MB_ICONEXCLAMATION)
    SetStatus(1)
    終了
End If
Print("[LIST]")
pos = 0
lf = chr(10)
While (strTok(attrList, pos, lf, str))
    Print(str)
Wend

If Not(ReadIniEntry("SINGLE", "Entry", attrList, "c:\temp\test.ini")) Then

MsgBox("Can not read SINGLE section", "DMS", MB_OK + MB_ICONEXCLAMATION)
    SetStatus(2)
    終了
End If
```

```
Print("[SINGLE]" + NEWLINE$ + "Entry=" + attrList)

If Not(DeleteIniEntry("SINGLE", "Entry", "c:¥temp¥test.ini")) Then
    MsgBox("Can not delete [SINGLE].Entry ", "DMS", MB_OK +
    MB_ICONEXCLAMATION)
    SetStatus(3)
    終了
End If
If Not(DeleteIniSection("LIST", "c:¥temp¥test.ini")) Then
    MsgBox("Can not delete LIST section", "DMS", MB_OK + MB_ICONEXCLAMATION)

SetStatus(4)
    終了
End If
```

WriteIniEntry - .ini ファイルへの値の格納

UNIX および Windows で有効です。

WriteIniEntry 関数は、初期化 (.ini) ファイルに値を格納します。ファイルがない場合は、作成されます。

関数の形式

```
WriteIniEntry(section as String, entry as String, value as String, grpfilename as
String) as Boolean
```

```
WriteIniEntry(filename as String) as Boolean (Windows only)
```

section

エントリおよび値を格納するセクションを指定します。

entry

値を格納するエントリを指定します。

value

格納する値を指定します。

grpfilename

初期化ファイルの名前を指定します。

filename

アドレス指定された初期化ファイルの名前。

注: WriteIniEntry{Filename as String} as Boolean の場合は、システムのキャッシュがフラッシュされるため、新しい値はリブートしなくてもアクティブになります。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

```
Dim file, section, entry, value as string

Dim LF, CR as char
Dim rtr as integer

ClrScr()
LF = 0x0a
CR = 0x0d
file = "c:\%dmscript.ini"
if Not(ExistFile(file)) then
    rtr = CreateFile(file, 0_TEXT)
    if rtr = -1 then
        SetStatus(1)
        exit
    end if
    closeFile(rtr)
end if

section = "Section 1"
value = "Param_1 = Wert_1" + LF + "Param_2 = Wert_2"

if WriteIniSection( section, value, file) then
    Print("WriteIniSection successfully completed.")
else
    Print("WriteIniSection failed.")
endif

section = "Section 2"
entry = "Param_3"
value = "Wert_3"

if WriteIniEntry( section, entry, value, file) then
    Print("WriteIniFile successfully completed.")
else
    Print("WriteIniFile failed.")
endif
```

```
section = "Section 1"
rtr = ReadIniSection(section, value, file)
if (rtr > 0) then
    Print( Str(rtr) + CR + LF + value)
else
    Print("ReadIniSection failed.")
endif

section = "Section 2"
entry = "Param_3"
rtr = ReadIniEntry(section, entry, value, file)

if (rtr > 0) then
    Print( Str(rtr) + CR + LF + entry + " = " + value)
else
    Print("ReadIniEntry failed.")
endif

endif
```

WriteIniSection - セクション全体の作成および上書き

UNIX および Windows で有効です。

WriteIniSection 関数は、特定の初期化 (.ini) ファイルのセクション全体を作成または上書きします。 .ini ファイルがない場合は、作成されます。

関数の形式

```
WriteIniSection(section as String, value as String, filename as String) as Boolean
section
```

書き込むセクションの名前を指定します。

value

書き込むエントリを示します。値のエントリは以下の形式で指定します。

```
Entry_1 = Value_1 <LF>
```

```
Entry_n = Value_n <LF>
```

filename

初期化ファイル名を指定します。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

```
Dim entries as String
Dim LF as String
LF = chr(10)
entries = "Entry_1 = Value_1" + LF + "Entry_n = Value_n"
Rem      この例では、c:¥temp¥test.ini にセクション LIST を書き込みます。

Dim attrList as String

lf = chr(10)
attrList = "Entry_1=Value_1"+lf+"Entry_2=Value_2"

If Not(WriteIniSection("LIST", attrList, "c:¥temp¥test.ini")) Then
    MsgBox("Can not create LIST section", "DMS", MB_OK + MB_ICONEXCLAMATION)
    SetStatus(1)
    Exit
End If
```

ReadIniEntry - 指定されたセクションからの値の取得

UNIX および Windows で有効です。

ReadIniEntry 関数は、初期化（.ini）ファイル内の指定セクションから値を取得します。

関数の形式

```
ReadIniEntry(section as String, entry as String, result as String, filename as String)
as Integer
```

section

エントリを含むセクションを指定します。

entry

関連する文字列を取得するエントリを指定します。

filename

初期化（.ini）ファイルの名前を指定します。**Windows** 環境でファイル名のみを指定した場合は、**Windows** ディレクトリが検索されますが、**Windows** 以外の環境では現在のディレクトリが検索されます。

result

値を検索する文字列変数を指定します。

`ReadIniEntry()` および `ReadIniSection()` は、呼び出しに成功した場合、正の値を返します。失敗した場合は、ゼロ (0) を返します。この関数は `entry` で指定した名前と一致するエントリを、初期化ファイル内の `section` で指定したセクション見出しで検索し、関連する文字列を結果変数にコピーします。

例：

```
Dim file, section, entry, value as string

Dim LF, CR as char
Dim rtr as integer

ClrScr()
LF = 0x0a
CR = 0x0d
file = "c:\dmscript.ini"
if Not(ExistFile(file)) then
    rtr = CreateFile(file, 0_TEXT)
    if rtr = -1 then
        SetStatus(1)
        exit
    end if
    closeFile(rtr)
end if

section = "Section 1"
value = "Param_1 = Wert_1" + LF + "Param_2 = Wert_2"

if WriteIniSection( section, value, file) then
    Print("WriteIniSection successfully completed.")
else
    Print("WriteIniSection failed.")
endif

section = "Section 2"
entry = "Param_3"
value = "Wert_3"

if WriteIniEntry( section, entry, value, file) then
    Print("WriteIniFile successfully completed.")
else
    Print("WriteIniFile failed.")
endif
```

```
section = "Section 1"
rtr = ReadIniSection(section, value, file)
if (rtr > 0) then
    Print( Str(rtr) + CR + LF + value)
else
    Print("ReadIniSection failed.")
endif

section = "Section 2"
entry = "Param_3"
rtr = ReadIniEntry(section, entry, value, file)

if (rtr > 0) then
    Print( Str(rtr) + CR + LF + entry + " = " + value)
else
    Print("ReadIniEntry failed.")
endif

endif
```

MIF および Inv ファイル関数

CreateMIFFile - MIF ファイルの作成

UNIX および Windows で有効です。

CreateMIFFile 関数は MIF ファイルを作成します。

関数の形式

CreateMIFFile(Filename as string, Name as string, Description as string) as integer

Filename

MIF ファイルの名前を指定します。

名前

MIF ファイルを格納するためのコンポーネント名を指定します。

説明

MIF ファイルを格納するための説明を指定します。

この関数の戻り値は整数です。

例：

以下の例では、C: の現在のディレクトリを、空きディスク領域と共に MIF ファイルに格納します。データは **Workstation** ディレクトリに格納されます。この例が有効なのは、**Asset Management** がインストールされている場合のみです。

```
Dim Dir as String
Dim DriveFree as Integer

'Get directory
Dir = GetDirectory(3)
DriveFree = GetDiskFreeMB(3)

'Create .MIF file
CreateMIFFile(ComputerPath+"Disk.MIF", "My Disk Inventory ", "", "")
CreateMIFGroup(ComputerPath+"Disk.MIF", "Disk Info", "Disk information", "AMClass")
CreateMIFString(ComputerPath+"Disk.MIF", "Disk Info", "Current Dir", Dir, "Dir name")
CreateMIFInteger(ComputerPath+"Disk.MIF", "Disk Info", "Free in MB", DriveFree, "Free MB")
```

CreateMIFGroup - グループの作成

UNIX および Windows で有効です。

CreateMIFGroup は既存の MIF ファイル内にグループを作成します。

関数の形式

```
CreateMIFGroup(Filename as string, Name as string, Description as string, Class as string) as integer
```

Filename

グループを作成する MIF ファイルの名前を指定します。

名前

作成するグループの名前を指定します。

説明

グループの説明を指定します。

Class

グループのクラスを指定します。

この関数が正常に完了すると、**MIF** ファイル内のグループに割り当てられた **ID** が返されます。それ以外の場合は、**0** が返されます。

例：

```
Dim Dir as String
Dim DriveFree as Integer

'Get directory
Dir = GetDirectory(3)
DriveFree = GetDiskFreeMB(3)

'Create .MIF file
CreateMIFFile(ComputerPath+"Disk.MIF","My Disk Inventory ","","")
CreateMIFGroup(ComputerPath+"Disk.MIF","Disk Info","Disk information","AMClass")
CreateMIFString(ComputerPath+"Disk.MIF","Disk Info","Current Dir",Dir,"Dir name")
CreateMIFInteger(ComputerPath+"Disk.MIF","Disk Info","Free in MB",DriveFree,"Free MB")
```

CreateMIFInteger - 整数属性の作成

UNIX および **Windows** で有効です。

この関数は、既存の **MIF** ファイル内の既存グループ内に整数属性を作成します。

関数の形式

```
CreateMIFInteger(Filename as string, Group as string, Name as string, Value as integer, Description as string) as integer
```

```
CreateMIFInteger(Filename as string, Group as integer, Name as string, Value as integer, Description as string) as integer
```

Filename

整数属性を作成する MIF ファイルの名前を指定します。

Group

整数属性を作成するグループを指定します。グループ名を指定する文字列、またはグループの ID を指定する整数のいずれかです。この ID は、**CreateMIFGroup** 関数を呼び出すことによって返される値です。

名前

属性の名前を指定します。

Value

属性の値を指定する整数。

説明

属性に関する説明を指定します。

この関数が正常に完了すると、MIF ファイル内のグループの属性の ID が返されます。それ以外の場合は、0 が返されます。

例：

```
Dim Dir as String
Dim DriveFree as Integer

'Get directory
Dir = GetDirectory(3)
DriveFree = GetDiskFreeMB(3)

'Create .MIF file
CreateMIFFile(ComputerPath+"Disk.MIF","My Disk Inventory ","","")
CreateMIFGroup(ComputerPath+"Disk.MIF","Disk Info","Disk information","AMClass")
CreateMIFString(ComputerPath+"Disk.MIF","Disk Info","Current Dir",Dir,"Dir name")
CreateMIFInteger(ComputerPath+"Disk.MIF","Disk Info","Free in MB",DriveFree,"Free MB")
```

CreateMIFString - 文字列属性の作成

UNIX および Windows で有効です。

CreateMIFString 関数は、既存の **MIF** ファイル内の既存グループに文字列属性を作成します。

関数の形式

CreateMIFString (Filename as String, Group as String, Name as String, Value as String, Description as String) as Integer

Filename

文字列属性を作成する **MIF** ファイルの名前を指定します。

Group

文字列属性を作成するグループを指定します。グループ名を指定する文字列、またはグループの **ID** を指定する整数のいずれかです。この **ID** は、**CreateMIFGroup** 関数を呼び出すことによって返される値です。

名前

属性の名前を指定します。

Value

属性の値を指定する文字列。

説明

属性に関する説明を指定します。

この関数が正常に完了すると、**MIF** ファイル内のグループの属性の **ID** が返されます。それ以外の場合は、**0** が返されます。

例：

```
Dim Dir as String
Dim DriveFree as Integer

'Get directory
Dir = GetDirectory(3)
DriveFree = GetDiskFreeMB(3)

'Create .MIF file
CreateMIFFile(ComputerPath+"Disk.MIF","My Disk Inventory ","","")
CreateMIFGroup(ComputerPath+"Disk.MIF","Disk Info","Disk information","AMClass")
CreateMIFString(ComputerPath+"Disk.MIF","Disk Info","Current Dir",Dir,"Dir name")
CreateMIFInteger(ComputerPath+"Disk.MIF","Disk Info","Free in MB",DriveFree,"Free
MB")
```

GetMIFComponent または GetComponent - コンポーネント名の取得

UNIX および Windows で有効です。

GetMIFComponent または **GetComponent** は、既存の .MIF または Inv ファイルのコンポーネント名を取得します。

関数の形式

```
GetMIFComponent(Filename as string) as string
```

```
GetComponent(Filename as string) as string
```

Filename

コンポーネントを取得する **MIF** ファイルの名前を指定します。

この関数が正常に完了すると、コンポーネント名を含む文字列が返されます。それ以外の場合は、空の文字列が返されます。

例：

以下に、**GetMIFComponent**、**GetMIFInteger**、**GetMIFString**、**GetMIFValue**、および **SetMIFValue** 関数の例を示します。

```
Dim file, gname, aname as string
Dim gid as integer
```

```
ClrScr()
file = "h:¥test¥miffus¥file.mif"

Print("Component of "" + file + "": "" + GetMIFComponent(file) + "")

gname = "Strings"
gId = 1
aname = "string_1"
Print(gname + "." + aname + " = " + GetMIFString(file, gname, gId, aname, 1))
aname = "string_2"
Print(gname + "." + aname + " = " + GetMIFValue(file, gname, gId, aname, 2))

if SetMIFValue(file, gname, gId, aname, 2, "Text_new") then
    Print(gname + "." + aname + " = " + GetMIFValue(file, gname, gId, aname, 2))
else
    Print("1. SetMIFValue failed.")
    exit
endif

gname = "Numbers"
gId = 2
aname = "number_1"
Print(gname + "." + aname + " = " + Str(GetMIFInteger(file, gname, gId, aname, 1)))
aname = "number_2"
Print(gname + "." + aname + " = " + GetMIFValue(file, gname, gId, aname, 2))
if SetMIFValue(file, gname, gId, aname, 2, "999999") then

Print(gname + "." + aname + " = " + GetMIFValue(file, gname, gId, aname, 2))
else
    Print("2. SetMIFValue failed.")
    exit
endif
```

GetMIFInteger - 整数属性値の取得

UNIX および Windows で有効です。

GetMIFInteger は、MIF ファイルの特定の整数属性の値を取得します。

関数の形式

GetMIFInteger(Filename as string, GroupName as string, AttrName as string) as integer

GetMIFInteger(Filename as string, GroupID as integer, AttrID as integer) as integer

GetMIFInteger(Filename as string, GroupName as string, GroupID as integer, AttrName as string, AttrID as integer) as integer

Filename

MIF ファイルのファイル名です。

GroupName

属性が属しているグループの名前を指定します。

GroupID

属性が属しているグループの ID を指定します。

AttrName

値を取得する属性の名前を指定します。

AttrID

値を取得する属性の ID を指定します。

指定された属性が MIF ファイルにある場合、**GetMIFInteger** はその整数属性の値を返します。それ以外の場合は、0 を返します。

例：

以下に、**GetMIFComponent**、**GetMIFInteger**、**GetMIFString**、**GetMIFValue**、および **SetMIFValue** 関数の例を示します。

```
Dim file, gname, aname as string
Dim gid as integer

ClrScr()
file = "h:\test\miffus\file.mif"

Print("Component of "" + file + "": "" + GetMIFComponent(file) + "")

gname = "Strings"
gid = 1
aname = "string_1"
Print(gname + "." + aname + " = " + GetMIFString(file, gname, gid, aname, 1))
aname = "string_2"
Print(gname + "." + aname + " = " + GetMIFValue(file, gname, gid, aname, 2))

if SetMIFValue(file, gname, gid, aname, 2, "Text_new") then
    Print(gname + "." + aname + " = " + GetMIFValue(file, gname, gid, aname, 2))
else
    Print("1. SetMIFValue failed.")
    exit
endif
```

```
gname = "Numbers"
gId = 2
aname = "number_1"
Print(gname + "." + aname + " = " + Str(GetMIFInteger(file, gname, gId, aname, 1)))
aname = "number_2"
Print(gname + "." + aname + " = " + GetMIFValue(file, gname, gId, aname, 2))
if SetMIFValue(file, gname, gId, aname, 2, "999999") then

Print(gname + "." + aname + " = " + GetMIFValue(file, gname, gId, aname, 2))
else
    Print("2. SetMIFValue failed.")
    exit
endif
```

GetMIFString - 文字列属性値の取得

UNIX および Windows で有効です。

GetMIFString 関数は、MIF ファイルの特定の文字列属性の値を取得します。

関数の形式

GetMIFString(Filename as string, GroupName as string, AttrName as string) as string

GetMIFString(Filename as string, GroupID as integer, AttrID as integer) as string

GetMIFString(Filename as string, GroupName as string, GroupID as integer, AttrName as string, AttrID as integer) as string

Filename

MIF ファイル名を指定します。

GroupName

属性が属しているグループの名前を指定します。

GroupID

属性が属しているグループの ID を指定します。

AttrName

値を取得する属性の名前を指定します。

AttrID

値を取得する属性の ID を指定します。

指定された属性が MIF ファイルにある場合、GetMIFString はその属性値を表す文字列を返します。それ以外の場合は、空の文字列を返します。

例：

以下に、GetMIFComponent、GetMIFInteger、GetMIFString、GetMIFValue、および SetMIFValue 関数の例を示します。

```
Dim file, gname, aname as string
Dim gid as integer

ClrScr()
file = "h:¥test¥miffus¥file.mif"

Print("Component of "" + file + """: "" + GetMIFComponent(file) + """)

gname = "Strings"
gid = 1
aname = "string_1"
Print(gname + "." + aname + " = " + GetMIFString(file, gname, gid, aname, 1))
aname = "string_2"
Print(gname + "." + aname + " = " + GetMIFValue(file, gname, gid, aname, 2))

if SetMIFValue(file, gname, gid, aname, 2, "Text_new") then
    Print(gname + "." + aname + " = " + GetMIFValue(file, gname, gid, aname, 2))
else
    Print("1. SetMIFValue failed.")
    exit
endif

gname = "Numbers"
gid = 2
aname = "number_1"
Print(gname + "." + aname + " = " + Str(GetMIFInteger(file, gname, gid, aname, 1)))
aname = "number_2"
Print(gname + "." + aname + " = " + GetMIFValue(file, gname, gid, aname, 2))
if SetMIFValue(file, gname, gid, aname, 2, "999999") then

Print(gname + "." + aname + " = " + GetMIFValue(file, gname, gid, aname, 2))
else
    Print("2. SetMIFValue failed.")
    exit
endif
```

GetMIFValue - 属性値を取得します。

UNIX および Windows で有効です。

GetMIFValue は、MIF ファイルの特定の属性の値を取得します。

関数の形式

```
GetMIFValue(Filename as string, GroupName as string, AttrName as string) as string
```

```
GetMIFValue(Filename as string, GroupID as integer, AttrID as integer) as string
```

```
GetMIFValue(Filename as string, GroupName as string, GroupID as integer, AttrName as string, AttrID as integer) as string
```

Filename

MIF ファイルの名前を指定します。

GroupName

属性が属しているグループの名前を指定します。

GroupID

属性が属しているグループの ID を指定します。

AttrName

値を取得する属性の名前を指定します。

AttrID

値を取得する属性の ID を指定します。値がテキスト文字列の場合、戻り値は引用符で囲まれます。

指定された属性が MIF ファイルにある場合、GetMIFValue はその属性値を表す文字列を返します。それ以外の場合は、空の文字列を返します。

例：

以下に、GetMIFComponent、GetMIFInteger、GetMIFString、GetMIFValue、および SetMIFValue 関数の例を示します。

```
Dim file, gname, aname as string  
Dim gid as integer
```

```
ClrScr()  
file = "h:¥test¥miffus¥file.mif"
```

```

Print("Component of " + file + ": " + GetMIFComponent(file) + "")

gname = "Strings"
gId = 1
aname = "string_1"
Print(gname + "." + aname + " = " + GetMIFString(file, gname, gId, aname, 1))
aname = "string_2"
Print(gname + "." + aname + " = " + GetMIFValue(file, gname, gId, aname, 2))

if SetMIFValue(file, gname, gId, aname, 2, "Text_new") then
    Print(gname + "." + aname + " = " + GetMIFValue(file, gname, gId, aname, 2))
else
    Print("1. SetMIFValue failed.")
    exit
endif

gname = "Numbers"
gId = 2
aname = "number_1"
Print(gname + "." + aname + " = " + Str(GetMIFInteger(file, gname, gId, aname, 1)))
aname = "number_2"
Print(gname + "." + aname + " = " + GetMIFValue(file, gname, gId, aname, 2))
if SetMIFValue(file, gname, gId, aname, 2, "999999") then

Print(gname + "." + aname + " = " + GetMIFValue(file, gname, gId, aname, 2))
else
    Print("2. SetMIFValue failed.")
    exit

endif

```

SetComment - UAM の [ジョブ コメント] フィールドへのテキストのコピー

UNIX および Windows で有効です。

パラメータ *Text* で指定された文字列を Asset Management の [ジョブ コメント] フィールドにコピーします。

関数の形式

```
SetComment(Text as string) as integer
```

テキスト

コメント テキストを指定します。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

このスクリプトは、**SetStatus** 関数の使用法を示します。

```
if MsgBox("Should this job return an error?",MB_YESNO)=IDYES then
    SetStatus(NC_ERROR)
else
    SetStatus(NC_OK)
    Print ("The user selected OK when prompted...")
end if
```

SetInvValue - 属性値の変更

UNIX および **Windows** で有効です。

SetInvValue 関数は、Inv ファイル内の特定の属性値を変更します。

関数の形式

```
SetInvValue(Filename as string, GroupName as string, GroupID as string, AttrName as string, AttrID as string, Value as string) as string
```

Filename

Inv ファイルのファイル名を指定します。

GroupName

属性が属しているグループの名前を指定します。

GroupID

属性が属しているグループの ID を指定します。

AttrName

設定する属性の名前を指定します。

AttrID

設定する属性の ID を指定します。

Value

Inv ファイル内の属性に設定する値を指定します。

Inv 構造化ファイルの場合、**GroupName** は完全なグループ名を指定し、**GroupID** はタイプを含む関連インデックス番号を指定します。**AttrName** は属性の名前であり、**AttrID** はタイプおよび表示タイプを含む関連インデックスを指定します。

指定した属性が Inv ファイルにある場合、値は変更され、**SetInvValue** は **TRUE** を返します。それ以外の場合は、**FALSE** を返します。

例：

```
SetInvValue("sample.inc", "$GeneralInventory$|Drive$", "000A02", "name", "00010500", "C:")
```

SetLocalPath - UAM の [ジョブ ローカル パス] フィールドへの文字列のコピー

UNIX および Windows で有効です。

SetLocalPath 関数は、パラメータ **Text** で指定された文字列を **Asset Management (UAM)** の [ジョブ ローカル パス] フィールドにコピーします。

関数の形式

```
SetLocalPath(Text as string) as Boolean
```

Text

ローカルパス (C:¥TEMP¥NEW など) を定義します。

この関数が正常に完了すると、**TRUE** (ゼロ以外) が返されます。それ以外の場合は、**FALSE** (ゼロ) が返されます。

例：

```
If (SetLocalPath("C:¥TEMP¥NEW"))
Then Print ("SetLocalPath succeeded")
Else
    Print ("SetLocalPath failed")
Endif
```

SetMIFValue - 属性値の変更

UNIX および Windows で有効です。

SetMIFValue 関数は、MIF ファイルの特定の属性の値を変更します。

関数の形式

```
SetMIFValue(Filename as string, GroupName as string, AttrName as string, Value as
string) as Boolean
```

```
SetMIFValue(Filename as string, GroupID as integer, AttrID as integer, Value as
string) as Boolean
```

```
SetMIFValue(Filename as string, GroupName as string, GroupID as integer, AttrName as
string, AttrID as integer, Value as string) as Boolean
```

Filename

MIF ファイルのファイル名を指定します。

GroupName

属性が属しているグループの名前を指定します。

GroupID

属性が属しているグループの ID を指定します。

AttrName

設定する属性の名前を指定します。

AttrID

設定する属性の ID を指定します。

Value

MIF ファイル内の、属性を設定する値を指定します。

整数属性コードに新しい整数を割り当てるには、以下のようにします。

- value="123456"

新しい文字列を割り当てるには、文字列を追加引用符で囲む必要があります。そうしないと、以降の MIF 関数によって予期せぬ結果が生じることがあります。

- value=""123456""

指定した属性が MIF ファイルにある場合、値は変更され、SetMIFValue は True を返します。それ以外の場合は、False を返します。

例：

以下に、GetMIFComponent、GetMIFInteger、GetMIFString、GetMIFValue、および SetMIFValue 関数の例を示します。

```
Dim file, gname, aname as string
Dim gid as integer

ClrScr()
file = "h:\test\miffus\file.mif"

Print("Component of "" + file + "": "" + GetMIFComponent(file) + "")

gname = "Strings"
gid = 1
aname = "string_1"
Print(gname + "." + aname + " = " + GetMIFString(file, gname, gid, aname, 1))
aname = "string_2"
Print(gname + "." + aname + " = " + GetMIFValue(file, gname, gid, aname, 2))

if SetMIFValue(file, gname, gid, aname, 2, "Text_new") then
    Print(gname + "." + aname + " = " + GetMIFValue(file, gname, gid, aname, 2))
else
    Print("1. SetMIFValue failed.")
    exit
endif
```

```
gname = "Numbers"
gId = 2
aname = "number_1"
Print(gname + "." + aname + " = " + Str(GetMIFInteger(file, gname, gId, aname, 1)))
aname = "number_2"
Print(gname + "." + aname + " = " + GetMIFValue(file, gname, gId, aname, 2))
if SetMIFValue(file, gname, gId, aname, 2, "999999") then

Print(gname + "." + aname + " = " + GetMIFValue(file, gname, gId, aname, 2))
else
    Print("2. SetMIFValue failed.")
    exit

endif
```

その他の関数

Argc - スクリプトに渡された引数の数を返す

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

argc 関数は、スクリプトに渡された引数の数を返します。少なくとも 1 つの引数（スクリプト名）が常に表示されます。

関数の形式

Argc() as Integer

この関数は、スクリプトに渡された引数の数（整数）を返します。値が 1 の場合は、スクリプトのみが渡されています。

例：

```
Dim i as Integer
For i = 0 To argc() - 1
    Print(argv(i))
Next i
```

Argv - nth スクリプト引数を含む文字列を返す

UNIX および Windows で有効です。

Argv 関数は、*n* 番目のスクリプト引数を含む文字列を返します。**argv(0)** は、スクリプトの名前です。

関数の形式

Argv(*n* as Integer) as String

n

引数のインデックス

戻り値は、*n* 番目のスクリプト引数を含む文字列です。

例：

```
Dim i as Integer
For i = 0 To argc() - 1
    Print(argv(i))
Next i
```

ChDir または SetDirectory - ディレクトリの変更または設定

UNIX および Windows で有効です。

ChDir または **SetDirectory** 関数は現在のディレクトリを変更します。

関数の形式

ChDir(*pathstring* as String) as Boolean

SetDirectory(*pathstring* as String) as Boolean

pathstring

ディレクトリ名です。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例 :

```
'Create dir
CreateDirectory("C:¥AMTest")

'Set active dir
SetDirectory("C:¥AMTest")
MessageBox(" Current Directory is : " + GetDirectory(0) )

'Set Default to C:¥
SetDirectory("C:¥")

'Rename the directory
RenameDirectory("C:¥AMTest","C:¥AMTest_1")

If ExistDirectory("C:¥AMTest_1") THEN
  RemoveDirectory("C:¥AMTest_1")

Else
  MessageBox("Could not find directory : C:¥AMTest_1")

Endif
```

Delay または Sleep - スクリプトが実行されるまで待機

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

Delay または Sleep 関数は、スクリプト実行を継続するまで n ミリ秒間待機します。

関数の形式

Delay(n as Integer) as Boolean

Sleep(n as Integer) as Boolean

n

遅延時間をミリ秒単位で指定します。 1 秒 = delay(1000) です。

この関数では、常に TRUE が返されます。

例：

```
delay(50)
```

スクリプトが実行されるまで 50 ミリ秒間待機します。

```
delay(1000)
```

スクリプトが実行されるまで 1 秒間待機します。

Exec または Execute - アプリケーションの実行

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

Exec または Execute 関数によって、指定されたアプリケーションが実行されます。

関数の形式

```
Execute(cmdline as String, wait as Boolean, style as Integer) as Integer
```

```
Execute(cmdline as String, wait as Boolean) as Integer
```

```
Execute(cmdline as String) as Integer
```

cmdline

コマンドラインを指定します(ファイル名とオプションパラメータ)。

注： Windows プラットフォームでは、実行するファイルが拡張子 .exe を持たない場合、ファイル拡張子を明示的に指定する必要があります。たとえば、Win9 では「command」ではなく「command.com」を指定します。

wait

アプリケーションが終了するまでスクリプトを待機させるかどうかを指定します。(Windows のみ)

style

アプリケーションのウィンドウのスタイルを指定します。（Windowsのみ）

注：コマンドラインのコマンドにスペース文字が含まれている場合は、コマンドを引用符で囲みます。例：`Execute(""%temp1¥prgedit.exe"",true,0)`

wait パラメータによって、アプリケーションが終了するまで待機せずに関数がスクリプト実行に戻るかどうかを指定します。値が **TRUE** の場合、関数はアプリケーションが終了するまで強制的に待機します。デフォルトは **TRUE** です。

UNIX プラットフォームでは、**wait** および **style** パラメータは無視されます。この関数は常に、アプリケーションが終了するまで待機してから、スクリプト実行に戻ります。

style パラメータによって、**Windows** ベースのアプリケーションのメインウィンドウの表示方法を指定します。このパラメータの値は、以下のいずれかになります。

値：0

ウィンドウは非表示になり、起動は別のウィンドウに渡されます。

値：1

ウィンドウが起動され、デフォルトのサイズおよび位置に表示されます。これはデフォルトです。

値：2

ウィンドウが起動され、アイコンとして表示されます。

値：3

ウィンドウが起動され、最大化されたウィンドウとして表示されます。

値：7

ウィンドウがアイコンとして表示されます。現在アクティブなウィンドウはアクティブのままです。

Windows の場合、関数が成功すると、マイナスでないリターンコードが `execute()` 関数に呼び出された指定コマンドのリターンコードを示します。インタプリタによってコマンドを始動できなかった場合、以下のリターンコードが返され、問題が発生したことを示します。

リターンコード：-1

システムのメモリまたはリソースが不足しています。

リターンコード：-2

指定されたファイルが見つかりませんでした。

リターンコード：-3

指定したパスが見つかりませんでした。

リターンコード：-4

この .exe ファイルは無効です。

リターンコード：-5

不明なエラーです。

UNIX では、システム関数「`system()`」によって指定されたコマンドが起動されます。したがって、「`system()`」のリターンコードが戻りコードになります。つまり、`system()` が失敗すると、-1 が返されます。`system()` が成功すると、「`waitpid()`」のリターンコード形式に応じた値が返されることを意味します。スクリプトでは、リターンコードを分析し、起動されたコマンドが成功したか失敗したかを判別する必要があります。この処理は、以下の関数を使用して実行できます。

```
'
' Execute() リターンコード評価は UNIX をサポートします。
'
'
'
' *****
' *****
'
' Name
'   DMS_ExecExit(retVal As Integer) As Boolean
```

```
'
'
' Description
'   Execute() によって始動されたプロセスが正常終了したかどうかを判別します。
'
' Arguments
'   retVal      評価対象のリターン コード
'
' Returns
'   TRUE 正常に終了した場合
'   FALSE   その他
'
' Comment
'   この関数はマクロ WIFEXITED の waitpid(5) 記述に対応します。
'   #define WIFEXITED(stat)      ((int)((stat)&0xFF) == 0)
'
Function DMS_ExecExit(retVal As Integer) As Boolean
    If (retVal And 255) Then
        DMS_ExecExit = FALSE
    Else
        DMS_ExecExit = TRUE
    End If
End Function

'
*****
*****
'
' Name
'   DMS_ExecExitStatus(retVal As Integer) As Integer
'
' Description
'   Execute() 呼び出しのリターン値、つまり起動されたプロセスのリターン コードの
'   シェル終了コードを示します。
'
' Arguments
'   retVal      Execute() から返されるリターン コード
'
' Returns
'   Execute() が正常終了した場合、子の表示ステータスが返されます。
'   そうでない場合は、retVal が返されます。
'
' Comment
'   この関数はマクロ WEXITSTATUS の waitpid(5) 記述に対応します。
'   #define WEXITSTATUS(stat)    ((int)(((stat)>>8)&0xFF))
```

```

Function DMS_ExecExitStatus(retVal As Integer) As Integer
    If Not(DMS_ExecExit(retVal)) Then
        DMS_ExecExitStatus = retVal
    Else
        DMS_ExecExitStatus = retVal / 256
        DMS_ExecExitStatus = DMS_ExecExitStatus And 255
    End If
End Function

```

関数が成功した場合、戻り値はマイナスでない値になります (WAIT によってアプリケーション終了まで関数が待機するように指定された場合はアプリケーションの終了コード)。そうでない場合、関数から以下のエラー値のいずれが返されます。

値 : -1

システムのメモリまたはリソースが不足しています。

値 : -2

指定されたファイルが見つかりませんでした。

値 : -3

指定したパスが見つかりませんでした。

値 : -4

この .exe ファイルは無効です。

値 : -5

不明なエラーです。

例 :

```

Dim Tempname as string '一時ファイルのファイル名を保有します。
Dim hFile as integer   '一時ファイルを操作します

' 最初に一時ファイルのファイル名を取得してファイルを作成します...
Tempname=EnvGetString("TEMP")
if TempName="" then
    TempName="MyFile.txt"
else
    Tempname=Tempname+"¥¥MyFile.txt"
end if
hfile=CreateFile(Tempname)

```

```
if hFile<0 then
    ' ファイルの作成中にエラーが発生したため、ユーザに通知し終了します。
    MsgBox("Could not create temporary file...")
    goto end
end if

' ファイルにテキストを保存しテキストを閉じます...
WriteFile(hFile,"This text is shown using notepad")
WriteFile(hFile,"")
WriteFile(hFile,"Please exit notepad to resume script execution")
CloseFile(hFile)

'そのファイルでメモ帳を起動します...
if Execute("notepad "+Tempname,TRUE)<0 then MsgBox("Could not launch notepad...")

'作業を終了する前に一時ファイルを必ず消去します...
DeleteFile(Tempname)

end:
```

Exit - スクリプトの終了

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

Exit 関数によってスクリプトは終了します。

関数の形式

```
exit()
```

この関数は値を返しません。

Not - 式の論理否定を返す

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

Not 関数は、論理否定を返します。

関数の形式

```
Not(n as Integer) as Integer
```

n

式。

パラメータ n が 0 と評価された場合、関数から TRUE (0 以外) が返されます。それ以外の場合は、FALSE (0) が返されます。

例：

rem

Rem この例では、スクリプト エディタのアイコンを含む DMS Test というアイコン グループを作成します。

Rem また、この例ではデスクトップにショートカットが追加されます

Rem

Rem 作成後、アイコン グループおよびショートカットは削除されます

Rem

' determine location of dmsedit.exe

Dim hkey, dummy As Integer

Dim dmseditPath As String

hkey = RegOpenKey(HKEY_LOCAL_MACHINE,
"SOFTWARE\ComputerAssociates\DMScript\DMSEdit")

If hKey = 0 Then

 MessageBox("Can not open DMSEdit-key", "Desktop Management Scripting", MB_OK
+ MB_ICONEXCLAMATION)

SetStatus(1)

 終了

End If

If Not(RegQueryVariable(hkey, "dmseditInstalledAt", dmseditPath, dummy) =
REG_STRING) Then

 MessageBox("dmseditInstalledAt not found or invalid", "Desktop Management
Scripting", MB_OK + MB_ICONEXCLAMATION)

 SetStatus(2)

 終了

End If

```
RegCloseKey(hkey)

' Now Create group and icons
If Not(CreateGroup("DMS Test", LNK_PROGRAMS)) Then
    MsgBox("Can not create icon group", "Desktop Management Scripting", MB_OK
+ MB_ICONEXCLAMATION)

SetStatus(3)
    終了
End If

If Not(AddItem("DMS Editor", dmseditPath+"%dmsedit.exe","", "", "DMS Test",FALSE,))
Then
    MsgBox("Failed to install dmsedit at icon group", "Desktop Management
Scripting", MB_OK + MB_ICONEXCLAMATION)
    SetStatus(4)
        終了
End If

if Not(CreateLink("DMS Editor", dmseditPath+"%dmsedit.exe","", "", "",LNK_NORMAL,
LNK_DESKTOP)) Then    MsgBox("Failed to install dmsedit at desktop", "Desktop
Management Scripting", MB_OK + MB_ICONEXCLAMATION)

SetStatus(5)
    終了
End If

MsgBox("Icons and shortcut has been created, press OK to delete them
again","Desktop Management Scripting: Icon Confirm", MB_OK + MB_ICONINFORMATION)

'Delete shortcut
if Not(DeleteItem("DMS Editor", "", LNK_DESKTOP)) Then
    MsgBox("Failed to deinstall dmsedit from desktop", "Desktop Management
Scripting", MB_OK + MB_ICONEXCLAMATION)
    SetStatus(6)
        終了
End If

if Not(DeleteGroup("DMS Test", LNK_PROGRAMS)) Then
    MsgBox("Failed to delete icon group", "Desktop Management Scripting",
MB_OK + MB_ICONEXCLAMATION)

SetStatus(7)
    終了

End If
```

Print - テキスト文字列の画面への出力

UNIX および Windows で有効です。

Print 関数によって、テキスト文字列が画面に出力されます。

関数の形式

```
Print(text as String) as Boolean
```

テキスト

出力するテキスト。

この関数では、常に **TRUE** が返されます。

例：

```
Dim counter as integer

'Print 5 lines
For counter = 1 to 5
    Print("Line " + str(counter))
Next

'Wait 5 seconds
Sleep(5000)

'Clear the print screen
ClearScreen()
Print("Screen Cleared")
```

SetDirectory - 現行ディレクトリの変更

Windows でのみ有効です。

SetDirectory 関数によって、現行ディレクトリは変更されます。

関数の形式

```
SetDirectory(pathstring as String) as Boolean
```

pathstring

作成するディレクトリの名前を指定します。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

```
Rem switch to c:¥backup
If Not(SetDirectory(c:¥backup) Then
    Print("c:¥backup does not exist");
    exit
End If
```

SetDrive - 現行ドライブの変更

Windows でのみ有効です。

SetDrive 関数によって、現行ドライブは変更されます。

関数の形式

SetDrive(n1 as Integer) as Boolean

n1

ドライブ番号を示します。パラメータ n1 によって、A は 1、B は 2、C は 3 というようにドライブが指定されます。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

```
Rem ドライブ d に切り替えます。
If Not(SetDrive(4)) Then
    Print("Drive 4 not available");
    exit
End If
```

SetStatus - ステータス値を返す

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

SetStatus 関数によって、スクリプトからステータス値が返されます。

関数の形式

```
SetStatus(status as Integer) as Boolean
```

status

設定する値を識別します。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例 :

このスクリプトは、SetStatus 関数の使用法を示します。

```
if MsgBox("Should this job return an error?",MB_YESNO)=IDYES then
    SetStatus(NC_ERROR)
else
    SetStatus(NC_OK)
    Print ("The user selected OK when prompted...")
end if
```

Sleep - その他の関数

Sleep は、Delay 関数のエイリアスです。

SortArray--配列のソート

UNIX および Windows で有効です。

SortArray 関数は配列をソートします。

関数の形式

```
SortArray(array as array) as Boolean
```

```
SortArray(array as array, start as integer) as Boolean
```

```
SortArray(array as array, start as integer, end as integer) as Boolean
```

Array

ソートされる配列の名前を識別します。

Start

ソートの開始インデックスを指定します。配列は開始インデックスから配列の最後までソートされます。

End

ソートの終了インデックスを指定します。

この関数は基本タイプの配列のみを処理します。選択した形式によって、
(1) 配列全体をソート、(2) 開始インデックスから配列をソート、(3) 開始インデックスから終了インデックスまで配列をソートします。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

```
Dim a[10], b[10], i As Integer
Dim TAB As Char
TAB = chr(9)

For i=0 to 9
    a[i] = 10 - i
Next i

b = a
If SortArray(b) Then
    For i = 0 To 9
        Print("a("+Str(i)+")="+Str(a[i])+TAB+"b("+Str(i)+")="+Str(b[i]))
    Next i
Else
    Print("SortArray(b) failed!")
End If
```

後方互換性または不明な機能の処理

DMScript は、スクリプト内の古いインタープリタに対する新しい関数、または不明の関数を処理するための関数を提供します。以下の関数により、新しい関数または不明の関数を無視することにより、古い DSM エージェント上で新しいスクリプトが正常に実行されることが保証されます。

- ALLOW_UNRESOLVED_FUNCTIONS
- IsFunctionSupported

ALLOW_UNRESOLVED_FUNCTIONS

IsFunctionSupported 関数は、指定された関数が現在のバージョンの dmscript によって解釈されるかどうかを確認します。このディレクティブの形式は、以下のとおりです。

```
#pragma ALLOW_UNRESOLVED_FUNCTIONS
```

DMScript にこのディレクティブを追加した場合、コンパイラは、スクリプト内に未知の関数を検出すると、以下のアクションを実行します。

- コンパイル時にトレース ログを警告に追加します
- 実行時にスクリプトが不明の関数を実行しようとする、致命的なランタイムエラーを出力するコードを発行します。

IsFunctionSupported

IsFunctionSupported 関数は、関数がサポートされているかどうかチェックします。

関数の形式

IsFunctionSupported (name as string) as Boolean

例：

```
#pragma ALLOW_UNRESOLVED_FUNCTIONS
If IsFunctionSupported ("OpenDetectedSoftwareOutputFiles") then
  OpenDetectedSoftwareOutputFiles ("A7C1E14A-7C93-4E17-B4E5-45B796717F49", "V1", "OS
  Detection for Windows")
else
  dsmtrace("error","This version of dmscript does not support Intellisigs.")
  exit
end if
```

入力パラメータ

この関数には、以下のパラメータがあります。

name

関数の名前を指定します。

戻り値

指定された関数がスクリプトで定義されているか、または組み込み関数である場合、この関数は **true** を返します。その他の場合には **false** を返します。この関数は、スクリプトで下位互換性を処理するため、**ALLOW_UNRESOLVED_FUNCTIONS** プラグマと共に使用されます。スクリプトで、以前のリリースには存在せず、現在のリリースに存在する関数を使用すると、スクリプトはコンパイルに失敗し、古いリリースを実行するコンピュータ上でアクションを実行しません。**IsFunctionSupported** は、そのようなスクリプトを古いリリースでコンパイル可能にし、欠落している機能処理するために、実行時に特殊なアクションを実行します。欠落している関数を補完することはできませんが、ユーザに対してエラーメッセージが出力されます。

注：また、[GetFileInfo](#) (P. 131) 関数または [CompareVersions](#) (P. 134) 関数を使用して、特定のバージョンの **DMScript** 上でスクリプトが実行されていることを確認できます。

ネットワーク関数

NetGetMap - ネットワーク リソース名の取得

Windows および Windows CE で有効です。

NetGetMap 関数によって、デバイスに指定済みのリダイレクトされたデバイスに関連したネットワーク リソースの名前を取得できます。

関数の形式

```
NetGetMap(device as String) as String
```

device

ドライブ仕様またはプリンタ仕様を指定します。有効な値は、**A:** から **Z:** および **LPT1:** から **LPT3:** です。

この関数によって、デバイスがリダイレクトされた先のネットワーク リソースの名前を含む文字列が返されます。デバイスがリダイレクトされていない場合、空の文字列が返されます。

例：

この例では、AM セクタ共有をマップし、共有を表示して、再度マップを解除します。

```
If SectorService <> "" Then
  If NetMap("U:",SectorService) Then
    Is Mapped to " + NetGetMap("U:")
    NetUnMap("U:")
  Else
    MessageBox("Could not map Sector Share")
  Endif
Else
  MessageBox("Could not find Sector Share Name")
EndIf
```

NetMap--ローカル デバイスのネットワーク リソースへのマップ

Windows および Windows CE で有効です。

NetMap 関数はローカル デバイスをネットワーク リソースにマップします。

関数の形式

NetMap(device as String, service as String, password as String) as Boolean

NetMap(device as String, service as String) as Boolean

device

リダイレクトされるローカル ドライブまたはデバイスを指定します (有効な値は、A: - Z: および LPT1: - LPT3: です)。

service

ローカル デバイスをリダイレクトする先のネットワーク リソースの名前を指定します。

パスワード

ネットワーク リソースのパスワードを指定するオプション文字列を指定します。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

この例では、AM セクタ共有をマップし、共有を表示して、再度マップを解除します。

```
If SectorService <> "" Then
  If NetMap("U:",SectorService) Then
    Is Mapped to " + NetGetMap("U:")
    NetUnMap("U:")
  Else
    MessageBox("Could not map Sector Share")
  Endif
Else
  MessageBox("Could not find Sector Share Name")
EndIf
```

NetUnMap - リダイレクトのキャンセル**Windows で有効**

NetUnMap 関数はリダイレクトされたデバイスのリダイレクトをキャンセルします。

関数の形式

NetUnMap(device as String) as Boolean

NetUnMap(device as String, force as Boolean) as Boolean

device

リダイレクトされたドライブまたはデバイスを指定します（有効な値は、A: - Z: および LPT1: - LPT3: です）。

force

接続をキャンセルする前に、デバイスで開かれているすべてのファイルを閉じるかどうか指定します。force パラメータが FALSE で、リダイレクトされたデバイスに開かれているファイルがある場合、この関数は失敗します。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

この例では、**AM** セクタ共有をマップし、共有を表示して、再度マップを解除します。

```
If SectorService <> "" Then
  If NetMap("U:",SectorService) Then
    Is Mapped to " + NetGetMap("U:")
    NetUnMap("U:")
  Else
    MessageBox("Could not map Sector Share")
  Endif
Else
  MessageBox("Could not find Sector Share Name")
EndIf
```

レジストリ操作関数

RegCloseKey - レジストリ キーを閉じる

Windows で有効

RegCloseKey 関数は、**RegOpenKey** または **RegCreateKey** の呼び出しで取得したハンドルである、**hKey** パラメータに関連したレジストリ キーを閉じます。

関数の形式

RegCloseKey(hKey as Integer) as Boolean

hkey

レジストリ キーの **hKey** 値

この関数が正常に終了すると、**TRUE** が返されます。この関数が失敗した場合 (**hKey** が有効な開かれたレジストリ キーのハンドルではない場合)、関数によって **FALSE** が返されます。

例：

```
Dim hkey1, hkey2, hkey3 as integer

Dim cBuf as string
Dim ccBuf[47] as char
Dim bBuf[100] as byte
Dim i as integer

ClrScr()
for i = 0 to 99
    bBuf[i] = i
next i
cBuf = "1234567890sqwertzuiopu+asdfghjkl0a#<yxcvbnm,.-"
ccBuf = cBuf

hkey1 = RegOpenKey(HKEY_LOCAL_MACHINE, "SOFTWARE")

if hkey1 = 0 then
    Print("RegOpenKey failed.")
    exit
endif

hkey2 = RegCreateKey(hkey1, "CA")
if hkey2 = 0 then
    Print("RegCreateKey 1 failed.")
    exit
endif

if Not(RegSetValue(hkey1, "CA", "GmbH Paderborn - Germany")) then

Print("RegSetValue failed")
endif

hkey3 = RegCreateKey(hkey2, "4 test only")
if hkey3 = 0 then
    Print("RegCreateKey 2 failed.")
    exit
endif
```

```
if Not(RegSetVariable(hkey3, "var_1", 123)) then
    Print("RegSetVar 1 failed.")
endif
if Not(RegSetVariable(hkey3, "var_2", "i'm a string")) then
    Print("RegSetVar 2 failed.")
endif
if Not(RegSetVariable(hkey3, "var_3", cBuf, 46)) then
    Print("RegSetVar 3 failed.")
endif
if Not(RegSetVariable(hkey3, "var_4", bBuf, 100)) then

Print("RegSetVar 4 failed.")
endif
if Not(RegSetVariable(hkey3, "var_5", cBuf, 46, REG_SZ)) then
    Print("RegSetVar 5 failed.")
endif
if Not(RegSetVariable(hkey3, "var_6", ccBuf, 46, REG_SZ)) then
    Print("RegSetVar 5 failed.")
endif

if Not(RegCloseKey(hkey3)) then
    Print("RegCloseKey for hkey3 failed.")
endif
if Not(RegCloseKey(hkey2)) then
    Print("RegCloseKey for hkey2 failed.")
endif
if Not(RegCloseKey(hkey1)) then
    Print("RegCloseKey for hkey1 failed.")

endif
```

RegCreateKey--レジストリ キーの作成

Windows で有効

RegCreateKey 関数は特定のレジストリ キーを作成します。レジストリにキーがすでに存在する場合は、そのキーが開かれます。

関数の形式

RegCreateKey(hKey as Integer, subkey as String) as Integer

hKey

現在開いているキーまたは事前定義されたキーを指定します。**hKey** パラメータには、すべての **Windows** プラットフォームで使用可能な事前定義された以下の値を使用できます。

HKEY_CLASSES_ROOT

HKEY_CURRENT_USER

HKEY_LOCAL_MACHINE

HKEY_USERS

subkey

開くキーの名前を示す文字列を指定します。サブキーは、**hKey** によって指定されたキーのサブキーである必要があります。**subkey** パラメーターが空の文字列の場合、この関数は **hKey** パラメータに指定されたものと同じキーを返します。

この関数が正常に完了すると、新しく作成されたレジストリ キーへのハンドルが返されます。そうでない場合は、0 が返されます。

例：

```
Dim hkey1, hkey2, hkey3 as integer
Dim cBuf as string
Dim ccBuf[47] as char
Dim bBuf[100] as byte
Dim i as integer

ClrScr()
for i = 0 to 99
    bBuf[i] = i
next i
cBuf = "1234567890sqwertzuiopu+asdfghjkloa#<yxcvbnm,.-"
ccBuf = cBuf

hkey1 = RegOpenKey(HKEY_LOCAL_MACHINE, "SOFTWARE")

if hkey1 = 0 then
    Print("RegOpenKey failed.")
    exit
endif

hkey2 = RegCreateKey(hkey1, "CA Technology")
if hkey2 = 0 then
    Print("RegCreateKey 1 failed.")
    exit
endif

if Not(RegSetValue(hkey1, "CA", "International")) then
    Print("RegSetValue failed")
endif

hkey3 = RegCreateKey(hkey2, "4 test only")
if hkey3 = 0 then
    Print("RegCreateKey 2 failed.")
    exit
endif

if Not(RegSetVariable(hkey3, "var_1", 123)) then
    Print("RegSetVar 1 failed.")
endif
if Not(RegSetVariable(hkey3, "var_2", "i'm a string")) then
    Print("RegSetVar 2 failed.")
endif
if Not(RegSetVariable(hkey3, "var_3", cBuf, 46)) then
    Print("RegSetVar 3 failed.")
endif
```

```
endif
if Not(RegSetVariable(hkey3, "var_4", bBuf, 100)) then
    Print("RegSetVar 4 failed.")
endif
if Not(RegSetVariable(hkey3, "var_5", cBuf, 46, REG_SZ)) then
    Print("RegSetVar 5 failed.")
endif
if Not(RegSetVariable(hkey3, "var_6", ccBuf, 46, REG_SZ)) then
    Print("RegSetVar 5 failed.")
endif

if Not(RegCloseKey(hkey3)) then
    Print("RegCloseKey for hkey3 failed.")
endif
if Not(RegCloseKey(hkey2)) then
    Print("RegCloseKey for hkey2 failed.")
endif
if Not(RegCloseKey(hkey1)) then
    Print("RegCloseKey for hkey1 failed.")
endif

endif
```

RegDeleteKey - レジストリ キーの削除

Windows で有効

RegDeleteKey 関数によって、特定のキーが削除されます。

関数の形式

RegDeleteKey(hKey as Integer, subkey as String) as Boolean

hKey

現在開かれているキーまたは事前定義されたキーを示します。すべての Windows プラットフォームで使用可能な事前定義された以下の値を使用できます。

HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS

subkey

パラメータで削除するキーの名前を示す文字列を指定します。サブキーは、**hKey** によって指定されたキーのサブキーである必要があります。 **subkey** パラメータに空の文字列は使用できません。

この関数が正常に終了すると、**TRUE** が返されます。キーが存在しないか、または現在別のアプリケーションによって開かれている場合、この関数によって **FALSE** が返されます。

subkeys

Windows 9x では、この関数はすべてのサブキーを削除します。**Windows NT** および **2000** では、関数を実行する前にサブキーを個別に削除する必要があります。

例：

テスト構造：

HKLM¥

¥ Software

¥ CA

¥ 4 test only

¥ var_6

¥ dummy

```
Dim hkey1 as integer
```

```
ClrScr()
```

```
hkey1 = RegOpenKey(HKEY_LOCAL_MACHINE, "SOFTWARE¥CA¥4 test only")
if hkey1 = 0 then
    Print("RegOpenKey failed.")
    exit
endif
```

```
if RegDeleteValue(hKey1) then
    Print("RegDeleteValue succeeded.")
else
    Print("RegDeleteValue failed.")
endif

if RegDeleteVariable(hKey1, "var_6") then
    Print("RegDeleteVariable for var_6 succeeded.")
else
    Print("RegDeleteVariable for var_6 failed.")
endif

RegCloseKey(hkey1)

hkey1 = RegOpenKey(HKEY_LOCAL_MACHINE, "SOFTWARE¥CA")

if hkey1 = 0 then
    Print("RegOpenKey failed.")
    exit
endif

if RegDeleteKey(hkey1, "dummy") then
    Print("Key dummy successfully deleted")
else
    Print("Key dummy deletion failed.")
endif

RegCloseKey(hkey1)

MessageBox("Check registry!", "desktop management scripting test")

hkey1 = RegOpenKey(HKEY_LOCAL_MACHINE, "SOFTWARE")
if hkey1 = 0 then
    Print("RegOpenKey failed.")
    exit
endif

if RegDeleteKey(hkey1, "CA¥4 test only") then
    Print("Key ""CA¥4 test only"" successfully deleted")
else
    Print("Key ""CA¥4 test only"" deletion failed.")
endif
```

```
if RegDeleteKey(hkey1, "CA") then
    Print("Key CA successfully deleted")
else
    Print("Key CA deletion failed.")
endif
```

```
RegCloseKey(hkey1)
```

RegDeleteValue - レジストリの値の削除

Windows で有効

RegDeleteValue 関数によってレジストリの値が削除されます。

関数の形式

```
RegDeleteValue(hkey as integer) as Boolean
```

hKey

関連した値を削除する先のレジストリ キーのハンドルを指定します。

戻り値は、関数が成功した場合は **TRUE**、そうでない場合は **FALSE** です。

例：

テスト構造：

```
HKLM¥
```

```
    ¥ Software
```

```
        ¥ CA
```

```
            ¥ 4 test only
```

```
                ¥ var_6
```

```
                    ¥ dummy
```

```
Dim hkey1 as integer
```

```
ClrScr()

hkey1 = RegOpenKey(HKEY_LOCAL_MACHINE, "SOFTWARE¥CA¥4 test only")
if hkey1 = 0 then
    Print("RegOpenKey failed.")
    exit
endif

if RegDeleteValue(hKey1) then
    Print("RegDeleteValue succeeded.")
else
    Print("RegDeleteValue failed.")
endif

if RegDeleteVariable(hKey1, "var_6") then
    Print("RegDeleteVariable for var_6 succeeded.")
else
    Print("RegDeleteVariable for var_6 failed.")
endif

RegCloseKey(hkey1)

hkey1 = RegOpenKey(HKEY_LOCAL_MACHINE, "SOFTWARE¥CA")
if hkey1 = 0 then
    Print("RegOpenKey failed.")
    exit
endif

if RegDeleteKey(hkey1, "dummy") then
    Print("Key dummy successfully deleted")
else
    Print("Key dummy deletion failed.")
endif

RegCloseKey(hkey1)

MessageBox("Check registry!", "desktop management scripting test")

hkey1 = RegOpenKey(HKEY_LOCAL_MACHINE, "SOFTWARE")
if hkey1 = 0 then
    Print("RegOpenKey failed.")
    exit
endif
```

```
if RegDeleteKey(hkey1, "CA¥4 test only") then
    Print("Key ""CA¥4 test only"" successfully deleted")

else
    Print("Key " "CA¥4 test only" " deletion failed.")
endif

if RegDeleteKey(hkey1, "CA") then
    Print("Key CA successfully deleted")
else
    Print("Key CA deletion failed.")
endif

RegCloseKey(hkey1)
```

RegDeleteVariable - 変数の削除

Windows で有効

RegDeleteVariable 関数によって、特定の変数が削除されます。

関数の形式

```
RegDeleteVariable(hKey as integer, name as string) as Boolean
```

hKey

現在開いているキーまたは事前定義されたキーを示します。**hKey** パラメータには、以下のいずれかの事前定義された値を指定できます。

```
HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS
```

name

削除する変数の名前を指定します。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

テスト構造：

HKLM¥

¥ Software

¥ CA

¥ 4 test only

¥ var_6

¥ dummy

Dim hkey1 as integer

ClrScr()

hkey1 = RegOpenKey(HKEY_LOCAL_MACHINE, "SOFTWARE¥CA¥4 test only")

if hkey1 = 0 then

Print("RegOpenKey failed.")

exit

endif

if RegDeleteValue(hKey1) then

Print("RegDeleteValue succeeded.")

else

Print("RegDeleteValue failed.")

endif

if RegDeleteVariable(hKey1, "var_6") then

Print("RegDeleteVariable for var_6 succeeded.")

else

Print("RegDeleteVariable for var_6 failed.")

endif

RegCloseKey(hkey1)

hkey1 = RegOpenKey(HKEY_LOCAL_MACHINE, "SOFTWARE¥CA")

if hkey1 = 0 then

Print("RegOpenKey failed.")

exit

endif

```
if RegDeleteKey(hkey1, "dummy") then
    Print("Key dummy successfully deleted")
else
    Print("Key dummy deletion failed.")
endif

RegCloseKey(hkey1)

MessageBox("Check registry!", "desktop management scripting test")

hkey1 = RegOpenKey(HKEY_LOCAL_MACHINE, "SOFTWARE")
if hkey1 = 0 then
    Print("RegOpenKey failed.")
    exit
endif

if RegDeleteKey(hkey1, "CA¥4 test only") then
    Print("Key ""CA¥4 test only"" successfully deleted")
else
    Print("Key ""CA¥4 test only"" deletion failed.")
endif

if RegDeleteKey(hkey1, "CA") then
    Print("Key CA successfully deleted")
else
    Print("Key CA deletion failed.")
endif

RegCloseKey(hkey1)
```

RegEnumKey - レジストリ キーのサブキーの列挙

Windows で有効

RegEnumKey 関数によって、指定されたレジストリ キーのサブキーが列挙されます。

関数の形式

RegEnumKey(hKey as Integer, index as Integer, subkey as String) as Boolean

hKey

現在開いているキーまたは事前定義されたキーを指定します。hKey パラメータには、以下のいずれかの事前定義された値を指定できます。

```
HKEY_CLASSES_ROOT  
HKEY_CURRENT_USER  
HKEY_LOCAL_MACHINE  
HKEY_USERS
```

index

取得するサブキーのインデックスを指定します。最初の関数呼び出しに対してゼロを使用します。

subkey

要求されたサブキーを保持する出力パラメータを指定します。

この関数が正常に完了すると、**TRUE** が返されます。また、サブキー変数にサブキー名が入力されます。キーが無効、またはサブキーが存在しない場合、**FALSE** が返されます。

例：

前提となるレジストリ構造：

HKLM

ソフトウェア

CA

test-1

...

test-9

dummy

4test only

var-0 "test 2"

...

```
Rem
Rem This example enumerates the subkeys of HKEY_CLASSES_ROOT,
Rem printing all filetype entries.
Rem

Dim SubKey as string
Dim Index as integer

Index=0
while RegEnumKey(HKEY_CLASSES_ROOT,Index,SubKey)
    if Left(SubKey,1)="." then print SubKey
    Index=Index+1
wend
```

RegEnumVariable - 変数の列挙

Windows で有効

RegEnumVariable 関数によって、特定のレジストリ キーの変数が列挙されます。

関数の形式

RegEnumVariable(hKey as Integer, index as Integer, name as String, strvalue as String, intvalue as Integer) as Integer

RegEnumVariable(hkey as integer, index as integer, dummy as string, name as string, bufsize as integer, buffer as void) as integer.

hKey

現在開いているキーまたは事前定義されたキーを指定します。**hKey** パラメータには、すべての **Windows** プラットフォームで使用可能な事前定義された以下の値を使用できます。

```
HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS
```

index

取得する変数のインデックスを指定します。最初の関数呼び出しに対してゼロを使用します。

name

要求された変数の名前を持つ出力パラメータを指定します。

strvalue

要求された変数の文字列値を持つ出力パラメータを指定します。

intvalue

要求された変数の整数値を持つ出力パラメータを指定します。

dummy

予約済み。

bufsize

バッファのタイプが **string** である場合は、受信側のバッファのサイズが **chars** 単位で指定されます。それ以外の場合は、バイト数が指定されます。変数である必要があります。正常に返された場合、取得したバイト数も含まれます。

buffer

取得された変数値を受信するためのバッファ。

この関数が正常に完了すると、関数の最初の変数で以下のいずれかの値が返されます。

REG_NODATA

値 0：変数が見つかりませんでした。これは、列挙の終了インデックスに達したか、またはハンドルが無効であることを意味します。

REG_INTEGER

値 1：整数タイプの変数が見つかりませんでした。値は **intvalue** 型の変数で保存されます。

REG_STRING

値 2：文字列タイプの変数が見つかりませんでした。値は **strvalue** 型の変数で保存されます。

REG_UNSUPPORTED

値 3：サポートされない変数タイプが見つかりました。値は **void** 型になります。

関数の 2 番目の変数が **Microsoft** の表記法で返されました。

REG_NONE

値 : 0

REG_SZ

値 : 1

REG_EXPAND_SZ

値 : 2

REG_BINARY

値 : 3

REG_DWORD

値 : 4

REG_DWORD_LITTLE_ENDIAN

値 : 4

REG_DWORD_BIG_ENDIAN

値 : 5

REG_LINK

値 : 6

REG_MULTI_SZ

値 : 7

例 :

```
Dim value as string
Dim hkey1 as integer
Dim name, str, dummy as string
Dim i, il, rtr, int as integer
Dim bBuf[100] as Byte
Dim cBuf[100] as char

ClrScr()

hkey1 = RegOpenKey(HKEY_LOCAL_MACHINE, "SOFTWARE\CA")
if hkey1 = 0 then
    Print("RegOpenKey failed.")
    exit
endif
```

```
i = 0
while
    RegEnumKey(hkey1, i, str)
    Print(Str(i) + " - " + str)

RegCloseKey(hkey1)

Print " "

hkey1 = RegOpenKey(HKEY_LOCAL_MACHINE, "SOFTWARE¥CA¥4 test only")
if hkey1 = 0 then
    Print("RegOpenKey failed.")
    exit
endif
for i = 0 to 1
    rtr = RegEnumVariable(hKey1, i, name, str, i1)
    if (REG_INTEGER = rtr) then
        Print(Str(rtr) + ": " + Str(i) + " - " + name + " = " + Str(i1))
    else
        if (REG_STRING = rtr) then
            Print(Str(rtr) + ": " + Str(i) + " - " + name + " = " + str)
        else
            Print(Str(rtr) + ": " + "unknown type");
        endif
    endif
endif
next i
i1 = 100
str = ""
rtr = RegEnumVariable(hKey1, 2, name, dummy, i1, bBuf)
for i = 0 to i1 - 1
    str = str + Str(bBuf[i])
next i
Print (Str(rtr) + ": " + "2 - " + name + "(" + Str(i1) + ") = " + str)
```

RegOpenKey -キーを開く

Windows で有効

RegOpenKey 関数によって、特定のレジストリ キーが開かれます。

関数の形式

RegOpenKey(hKey as Integer, subkey as String) as Integer

hKey

現在開いているキーまたは事前定義されたキーを指定します。**hKey** パラメータには、以下のいずれかの事前定義された値を指定できます。

```
HKEY_CLASSES_ROOT  
HKEY_CURRENT_USER  
HKEY_LOCAL_MACHINE  
HKEY_USERS
```

subkey

開くキーを指定します。サブキーのキーは、**hKey** パラメータによって指定されたキーのサブキーである必要があります。**subkey** パラメーターが空の文字列の場合、この関数は **hKey** パラメータに指定されたものと同じキーを返します。

この関数が正常に完了すると、新しく開かれたレジストリ キーへのハンドルが返されます。そうでない場合は、**0** が返されます。

例：

```
Dim hkey1, hkey2, hkey3 as integer  
  
Dim cBuf as string  
Dim ccBuf[47] as char  
Dim bBuf[100] as byte  
Dim i as integer  
  
ClrScr()  
for i = 0 to 99  
    bBuf[i] = i  
next i  
cBuf = "1234567890qwertyuiopu+asdfghjkl;a#<yxcvbnm,.-"  
ccBuf = cBuf  
  
hkey1 = RegOpenKey(HKEY_LOCAL_MACHINE, "SOFTWARE")
```

```
if hkey1 = 0 then
    Print("RegOpenKey failed.")
    exit
endif

hkey2 = RegCreateKey(hkey1, "CA")
if hkey2 = 0 then
    Print("RegCreateKey 1 failed.")
    exit
endif

if Not(RegSetValue(hkey1, "CA", "GmbH Paderborn - Germany")) then

Print("RegSetValue failed")
endif

hkey3 = RegCreateKey(hkey2, "4 test only")
if hkey3 = 0 then
    Print("RegCreateKey 2 failed.")
    exit
endif

if Not(RegSetVariable(hkey3, "var_1", 123)) then
    Print("RegSetVar 1 failed.")
endif
if Not(RegSetVariable(hkey3, "var_2", "i'm a string")) then
    Print("RegSetVar 2 failed.")
endif
if Not(RegSetVariable(hkey3, "var_3", cBuf, 46)) then
    Print("RegSetVar 3 failed.")
endif
if Not(RegSetVariable(hkey3, "var_4", bBuf, 100)) then

Print("RegSetVar 4 failed.")
endif
if Not(RegSetVariable(hkey3, "var_5", cBuf, 46, REG_SZ)) then
    Print("RegSetVar 5 failed.")
endif
if Not(RegSetVariable(hkey3, "var_6", ccBuf, 46, REG_SZ)) then
    Print("RegSetVar 5 failed.")
endif
endif
```

```
if Not(RegCloseKey(hkey3)) then
    Print("RegCloseKey for hkey3 failed.")
endif
if Not(RegCloseKey(hkey2)) then
    Print("RegCloseKey for hkey2 failed.")
endif
if Not(RegCloseKey(hkey1)) then
    Print("RegCloseKey for hkey1 failed.")
endif
```

RegQueryValue - テキスト文字列の取得

Windows で有効

開かれたキーで名前のないデフォルト値（データ）が抽出される場合、**RegQueryValue** 関数によって、指定されたレジストリ キーおよびサブキーに関連したテキスト文字列が取得されます。

関数の形式

RegQueryValue(hKey as Integer, subkey as String, value as String) as Boolean

hKey

現在開いているキーまたは事前定義されたキーを指定します。**hKey** パラメータには、以下のいずれかの事前定義された値を指定できます。

```
HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS
```

subkey

キーと共に、取得するデフォルト値を指定します。開かれたキーで名前のない値が抽出される場合、サブキーは任意のサブキーまたは空の文字列になります。サブキーは、**hKey** パラメータによって指定されたキーのサブキーである必要があります。パラメータ値は文字列変数で、サブキーの値を受け取ります。

Value

特定のキーおよびサブキーの戻り値（データ）を保持します。

この関数が正常に完了すると、**TRUE** が返されます。また、特定のキーおよびサブキーに関連した文字列が変数に入力されます。正常に完了しない場合は、**FALSE** が返されます。

RegQueryVariable - 変数のタイプおよび値の取得

Windows で有効

RegQueryVariable 関数は、指定されたレジストリ キーに対して指定された変数のタイプおよび値を取得します。

関数の形式

形式 1

RegQueryVariable(hKey as Integer, name as String, strvalue as String, intvalue as Integer) as Integer

hKey

現在開いているキーまたは事前定義されたキーを指定します。hKey パラメータには、以下のいずれかの事前定義された値を指定できます。

HKEY_CLASSES_ROOT

HKEY_CURRENT_USER

HKEY_LOCAL_MACHINE

HKEY_USERS

name

要求された変数の名前を指定します。

strvalue

要求された変数の文字列値を持つ出力パラメータを指定します。

intvalue

要求された変数の整数値を持つ出力パラメータを指定します。

この関数が正常に完了すると、以下の値のいずれかが返されます。

REG_NODATA

変数が見つかりませんでした。これはハンドルが無効であることを示します。

REG_INTEGER

整数タイプの変数が見つかりませんでした。値は `intvalue` 型の変数で保存されます。

REG_STRING

文字列タイプの変数が見つかりませんでした。値は `strvalue` 型の変数で保存されます。

REG_UNSUPPORTED

サポートされない変数タイプが見つかりました。値は `void` 型になります。

形式 2

`RegQueryVariable(hkey as integer, name as string, dummy as string, bufsize as integer, buffer as void) as integer.`

hKey

現在開いているキーまたは事前定義されたキーを指定します。`hKey` パラメータには、以下のいずれかの事前定義された値を指定できます。

`HKEY_CLASSES_ROOT`

`HKEY_CURRENT_USER`

`HKEY_LOCAL_MACHINE`

`HKEY_USERS`

name

要求された変数の名前を指定します。

dummy

予約済み。

bufsize

バッファのタイプが **string** である場合は、受信側のバッファのサイズが **chars** 単位で指定されます。それ以外の場合は、バイト数が指定されます。変数である必要があります。正常に返された場合、取得したバイト数も含まれます。

buffer

取得された変数値を受信するためのバッファ。

レジストリ エントリのタイプが **Microsoft** の表記法で返されました。

REG_NONE

値 : 0

REG_SZ

値 : 1

REG_EXPAND_SZ

値 : 2

REG_BINARY

値 : 3

REG_DWORD

値 : 4

REG_DWORD_LITTLE_ENDIAN

値 : 5

REG_DWORD_BIG_ENDIAN

値 : 6

REG_LINK

値 : 7

REG_MULTI_SZ

値 : 8

例：

以下の例を開始する前に、RegOpenKey の例を実行してください。

```
Dim value as string
Dim hkey1 as integer
Dim name, str, dummy as string
Dim i, il, rtr, int as integer
Dim bBuf[100] as Byte
Dim cBuf[100] as char

ClrScr()

hkey1 = RegOpenKey(HKEY_LOCAL_MACHINE, "SOFTWARE¥CA¥4 test only")
if hkey1 = 0 then
    Print("RegOpenKey failed.")
    exit
endif

rtr = RegQueryVariable(hkey1, "var_2", str, int)
Print(Str(rtr) + ": var_2 = " + str)
int = 100
str = ""
rtr = RegQueryVariable(hkey1, "var_2", dummy, int, str)
Print(Str(rtr) + ": var_2(" + Str(int) + ") = " + str)
int = 100
str = ""
rtr = RegQueryVariable(hkey1, "var_2", dummy, int, cBuf)
for il = 0 to int - 1
    str = str + cBuf[il]
next il
Print(Str(rtr) + ": var_2(" + Str(int) + ") = " + str)
int = 100
str = "| "
rtr = RegQueryVariable(hkey1, "var_2", dummy, int, bBuf)
for il = 0 to int - 1
    str = str + Str(bBuf[il]) + " | "
next il
Print(Str(rtr) + ": var_2(" + Str(int) + ") = " + str)

RegCloseKey(hkey1)
```

RegSetValue - レジストリ キーとテキスト文字列の関連付け

Windows で有効

RegSetValue 関数によって、指定されたレジストリ キーとテキスト文字列が関連付けられます。

関数の形式

RegSetValue(hKey as Integer, subkey as String, value as String) as Boolean

hKey

現在開いているキーまたは事前定義されたキーを指定します。**hKey** パラメータには、以下のいずれかの事前定義された値を指定できます。

HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS

subkey

設定するサブキーの名前を指定します。

value

変数に設定する文字列の値を指定します。

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

```
Dim hkey1, hkey2, hkey3 as integer
Dim cBuf as string
Dim ccBuf[47] as char
Dim bBuf[100] as byte
Dim i as integer

ClrScr()
for i = 0 to 99
    bBuf[i] = i
next i
cBuf = "1234567890sqwertzuiopu+asdfghjkl0a#<yxcvbnm,.-"
ccBuf = cBuf

hkey1 = RegOpenKey(HKEY_LOCAL_MACHINE, "SOFTWARE")

if hkey1 = 0 then
    Print("RegOpenKey failed.")
    exit
endif

hkey2 = RegCreateKey(hkey1, "CA Technology")
if hkey2 = 0 then
    Print("RegCreateKey 1 failed.")
    exit
endif

if Not(RegSetValue(hkey1, "CA", "International")) then
    Print("RegSetValue failed")
endif

hkey3 = RegCreateKey(hkey2, "4 test only")
if hkey3 = 0 then
    Print("RegCreateKey 2 failed.")
    exit
endif

if Not(RegSetVariable(hkey3, "var_1", 123)) then
    Print("RegSetVar 1 failed.")
endif
if Not(RegSetVariable(hkey3, "var_2", "i'm a string")) then
    Print("RegSetVar 2 failed.")
endif
if Not(RegSetVariable(hkey3, "var_3", cBuf, 46)) then
    Print("RegSetVar 3 failed.")
endif
if Not(RegSetVariable(hkey3, "var_4", bBuf, 100)) then
    Print("RegSetVar 4 failed.")
endif
```

```
endif
if Not(RegSetVariable(hkey3, "var_5", cBuf, 46, REG_SZ)) then
    Print("RegSetVar 5 failed.")
endif
if Not(RegSetVariable(hkey3, "var_6", ccBuf, 46, REG_SZ)) then
    Print("RegSetVar 5 failed.")
endif

if Not(RegCloseKey(hkey3)) then
    Print("RegCloseKey for hkey3 failed.")
endif
if Not(RegCloseKey(hkey2)) then
    Print("RegCloseKey for hkey2 failed.")
endif
if Not(RegCloseKey(hkey1)) then
    Print("RegCloseKey for hkey1 failed.")
endif

endif
```

RegSetVariable - レジストリ キーへの変数の保存

Windows で有効

RegSetVariable 関数によって、指定されたレジストリ キーに変数が保存されます。

関数の形式

RegSetVariable(hKey as Integer, name as String, value as Integer) as Boolean

RegSetVariable(hKey as Integer, name as String, value as String) as Boolean

RegSetVariable(hkey as integer, name as string, buffer as void, bufsize as integer) as integer.

RegSetVariable(hkey as integer, name as string, buffer as void, bufsize as integer, type as integer) as integer.

hKey

現在開いているキーまたは事前定義されたキーを指定します。hKey パラメータには、以下のいずれかの事前定義された値を指定できます。

HKEY_CLASSES_ROOT

HKEY_CURRENT_USER

HKEY_LOCAL_MACHINE

HKEY_USERS

name

設定する変数の名前を指定します。

value

変数に設定する文字列または整数値を指定します。

buffer

変数に設定する値を含むバッファです。

bufsize

バッファのタイプが **string** である場合、タイプ **char** の配列である場合、バッファのサイズは **chars** 単位で指定されます。それ以外の場合は、バイト数が指定されます。

type

レジストリ エントリのタイプ。

レジストリ エントリのタイプが **Microsoft** の表記法で返されました。

REG_NONE

値： 0

REG_SZ

値： 1

REG_EXPAND_SZ

値： 2

REG_BINARY

値： 3

REG_DWORD

値： 4

REG_DWORD_LITTLE_ENDIAN

値： 5

REG_DWORD_BIG_ENDIAN

値： 6

REG_LINK

値： 7

REG_MULTI_SZ

値： 8

この関数が正常に完了すると、**TRUE** が返されます。それ以外の場合は、**FALSE** が返されます。

例：

以下の例を開始する前に、**RegOpenKey** の例を実行してください。

```
Dim value as string
Dim hkey1 as integer
Dim name, str, dummy as string
Dim i, il, rtr, int as integer
Dim bBuf[100] as Byte
Dim cBuf[100] as char

ClrScr()

hkey1 = RegOpenKey(HKEY_LOCAL_MACHINE, "SOFTWARE¥CA¥4 test only")
if hkey1 = 0 then
    Print("RegOpenKey failed.")
    exit
endif

rtr = RegQueryVariable(hkey1, "var_2", str, int)
Print(Str(rtr) + ": var_2 = " + str)
int = 100
str = ""
rtr = RegQueryVariable(hkey1, "var_2", dummy, int, str)
Print(Str(rtr) + ": var_2(" + Str(int) + ") = " + str)
int = 100
str = ""
rtr = RegQueryVariable(hkey1, "var_2", dummy, int, cBuf)
for il = 0 to int - 1
    str = str + cBuf[il]
next il
Print(Str(rtr) + ": var_2(" + Str(int) + ") = " + str)
int = 100
str = "| "
rtr = RegQueryVariable(hkey1, "var_2", dummy, int, bBuf)
for il = 0 to int - 1
    str = str + Str(bBuf[il]) + " | "
next il
Print(Str(rtr) + ": var_2(" + Str(int) + ") = " + str)

RegCloseKey(hkey1)
```

SetMode64 - Windows 64 ビット レジストリおよびファイル システムへのアクセス

DMScript 関数は、Windows レジストリにアクセスしてプログラムを起動できます。DMScript は 32 ビット アプリケーションであるため、DMScript が 64 ビット Windows 上で実行される場合、OS は特定のルールを適用します。特定のキーに関するレジストリ アクセスは、並列の 32 ビット レジストリ領域に自動的にリダイレクトされます。たとえば、HKLM¥SOFTWARE¥xxx へのアクセスは HKLM¥SOFTWARE¥Wow6432Node¥xxx にリダイレクトされます。リダイレクトされるキーの完全なリストについては、[http://msdn.microsoft.com/en-us/library/aa384253\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/aa384253(v=vs.85).aspx) を参照してください。

同様に、システム ディレクトリを参照する特定の環境変数を使用して、ファイル システムにアクセスすると、32 ビット バイナリが含まれるディレクトリに自動的にリダイレクトされます。たとえば、%windir%¥System32 ディレクトリは、64 ビット アプリケーション用に予約されています。DMScript は、このディレクトリへのアクセスを試行すると、%windir%¥SysWOW64 に自動的にリダイレクトされます。詳細については、[http://msdn.microsoft.com/en-us/library/aa384187\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/aa384187(v=vs.85).aspx) を参照してください。

スクリプトは、正しい 64 ビット レジストリ キーおよびファイル システムにアクセスすることにより、64 ビット アプリケーションを検出する必要があります。この動作は Intellisig スクリプトにとって重要です。たとえば、Intellisig が HKLM¥SOFTWARE¥7-zip 内で 64 ビット バージョンの 7-Zip に関してレジストリ キーの読み取りを試行する場合、実際には HKLM¥SOFTWARE¥Wow6432Node¥7-zip を検索しますが、それは存在しないため、スクリプトが失敗します。同様に、64 ビット バージョンの Notepad に関して %windir%¥System32 の検索を行うと、32 ビット バージョンが検索されます。

この動作を処理するために、DMScript は、64 ビット Windows OS への自動的なリダイレクトをオフにする以下の機能を提供しています。

```
setmode64(mode)
```

mode

リダイレクトをオフまたはオンにする必要があるかどうかを指定します。値 1 はリダイレクトをオフにし、値 0 はオンにします。

例：setmode64(1)

注：64 ビット モードの場合、HKLM¥SOFTWARE¥Wow6432Node¥xxx などの物理名を使用して 32 ビット レジストリにアクセスします。ただし、Windows の今後のバージョンでリダイレクト方法が変化する可能性があるため、Microsoft はこの方法を推奨していません。代わりに、スクリプトを 32 ビット モードに戻す必要があります。

重要： より長い期間ファイル システム リダイレクトを無効にすると、dmscript がシステム DLL をロードできなくなり、失敗する場合があります。

文字列操作関数

文字列操作関数は、サポート対象のすべてのオペレーティング システムで使用できます。

重要： Linux コンピュータ上では、文字列操作関数に入力される文字列に日本語を含めることはできません。

Asc--文字の対応する ASCII 値への変換

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

asc 関数によって、1 つの文字を対応する ASCII 値、または対応するグリフの全角文字コードに変換することができます。パラメータ s1 が文字列で複数の文字を含む場合、関数では最初の文字のみが評価されます。

関数の形式

Asc(s1 as String) as Integer

Asc(s1 as Char) as Integer

s1

文字の名前を指定します。

戻り値は入力文字の全角文字コード値または入力文字列の先頭文字になります。

例：

```
Dim OutStr as string

OutStr = Str(Asc("A"))+" "+Str(Asc("B"))+" "+Str(Asc("C"))

Print "AscII Values of the letters in "ABC" is: " + OutStr
```

Chr - 整数の全角文字への変換

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

Chr 関数によって、整数値は対応する全角文字に変換されます。

関数の形式

```
Chr(n1 as Integer) as Char
```

n1

整数値を指定します。

この関数によって、n1 パラメータで指定された全角文字が 1 つ返されます。

例：

```
Print "LITTLE "+Chr(65)+Chr(66)+Chr(67)
```

InStr - サブ文字列があるかどうか文字列をスキャン

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

InStr 関数によって、指定されたサブ文字列があるかどうか文字列がスキャンされます。この関数では、最初にサブ文字列 **substr** が存在する場所を探して文字列 **str** がスキャンされます。

関数の形式

```
Instr(str as String, substr as String) as Integer
```

str

検索を実行する対象の文字列を指定します。

substr

検索を実行する対象のサブ文字列を指定します。

この関数によって、**substr** が始まる **str** 内の文字位置が返されます。 **str** に **substr** がない場合、関数によって 0 が返されます。

例：

以下の例では、**InStr** 関数を使用して、「**Script**」が「**Unicenter Asset Management Script Interpreter**」という文にあるかどうかを判別します。

```
Dim MyStr as String ' Contains string to evaluate
Dim Pos as Integer ' Holds a string index

MyStr="Unicenter Asset Management Script Interpreter"

Pos=InStr(MyStr,"Script")

if Pos>0 then
    Print "The word ""Script"" was found at character position "+str(Pos)+" in"
    Print "the sentence """+MyStr+""". "
else
    Print "The word ""Script"" was NOT found in"
    Print "the sentence """+MyStr+""". "
End if
```

LCase - 大文字の小文字への変換

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

LCase 関数によって、**s1** 文字列の大文字が小文字に変換されます。その他の文字は影響を受けません。ASCII コードテーブルに対する制限はありません。

関数の形式

Lcase (s1 as String) as String

s1

変換する文字列を指定します。

この関数では、変換された文字列が返されます。

例：

以下の例では、テキストが大文字および小文字で表示されます。

```
Rem
Dim text As String
Text="Little ABC"

Print("LCase : " + LCase(text)) 'little abc

Print("UCase : " + UCase(text)) 'LITTLE ABC
```

Left - 文字列の最も左の文字を返す

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

Left 関数によって、文字列 **str** の最も左の文字を含むサブ文字列が返されます。

関数の形式

```
Left(str as String, n as Integer) as String
```

str

左のサブ文字列を取得する先の文字列を示します。

n

左のサブ文字列の文字数を示します。

文字列 **str** で最も左の **n** 文字を含むサブ文字列が返されます。

例： Left 関数

以下の例では、文字列操作の方法が示されます。

```
Dim text As String
Text="Unicenter TNG"

Print("Left(text,3) : " + Left(text,3)) 'Uni
Print("Mid(text,4,6) : " + Mid(text,4,6)) 'center
Print("Right(text,3) : " + Right(text,3)) 'TNG
Print("Length : " + Str(Len(text))) '13
```

Len - 文字列の長さを取得

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

Len 関数によって、文字列 **str** の長さが取得できます。

関数の形式

```
Len(str as String) as Integer
```

str

特定の文字列を指定します。

文字列 **str** の文字数が返されます。

例：

以下の例では、文字列操作の方法が示されます。

```
Dim text As String
Text="Unicenter TNG"

Print("Left(text,3) : " + Left(text,3)) 'Uni
Print("Mid(text,4,6) : " + Mid(text,4,6)) 'center
Print("Right(text,3) : " + Right(text,3)) 'TNG
Print("Length : " + Str(Len(text))) '13
```

Mid - 文字列からサブ文字列を返す

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

Mid 関数によって、文字列からサブ文字列が返されます。

関数の形式

```
Mid(str as String, start as Integer, length as Integer) as String
```

```
Mid (str as String, start as Integer) as String
```

str

サブ文字列を取得する先の文字列を指定します。

start

str におけるサブ文字列の開始位置を示します。

len

サブ文字列の長さを示します。

注： 引数の開始位置および長さは **1** から **65,535** までである必要があります。長さが省略された場合、または **str** における長さが *n* 文字よりも短い場合、関数によって開始位置から文字列の最後まですべての文字が返されます。開始位置が **str** の文字数よりも大きい場合、長さが **0** の文字列が返されます。

関数の戻り値は、前述の文字列 **str** のサブ文字列です。

例：

以下の例では、文字列操作の方法が示されます。

```
Dim text As String
Text="Unicenter TNG"

Print("Left(text,3) : " + Left(text,3)) 'Uni
Print("Mid(text,4,6) : " + Mid(text,4,6)) 'center
Print("Right(text,3) : " + Right(text,3)) 'TNG
Print("Length      : " + Str(Len(text))) '13
```

Right - 最も右の文字が n のサブ文字列を返す

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

Right 関数によって、文字列 **str** の最も右の文字を含むサブ文字列が返されます。

関数の形式

```
Right(str as String, n as Integer) as String
```

str

右のサブ文字列を取得する先の文字列を示します。

n

サブ文字列の文字数を示します。

この関数によって、文字列 **str** の最も右の文字を含むサブ文字列が返されます。

例：

以下の例では、文字列操作の方法が示されます。

```
Dim text As String
Text="Unicenter TNG"

Print("Left(text,3) : " + Left(text,3)) 'Uni
Print("Mid(text,4,6) : " + Mid(text,4,6)) 'center
Print("Right(text,3) : " + Right(text,3)) 'TNG
Print("Length      : " + Str(Len(text))) '13
```

Str - 数値表記の文字列への変換

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

Str 関数によって、数値表記の値が文字列に変換されます。

関数の形式

```
Str(n as Integer) as String
```

n

変換対象の整数を示します。

この関数によって、パラメータ n で指定された値を表す文字列が返されます。

例：

以下の例では、文字「ABC」の ASCII 値が画面に表示されます。

```
Dim OutStr as string
```

```
OutStr = Str(Asc("A"))+" "+Str(Asc("B"))+" "+Str(Asc("C"))
```

```
Print "AscII Values of the letters in ""ABC"" is: " + OutStr
```

Trim--文字列からの先頭スペースおよび後続スペースの削除

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

Trim 関数によって、文字列から先頭スペースおよび後続スペースが削除されます。

関数の形式

```
Trim(str as String) as String
```

str

削除対象の文字列を指定します。

この関数では、削除された文字列が返されます。

例：

```
Dim tStr, str As String

str = "  abcd "
tStr = Trim(str)
Print(">>>" + Str + "<<<")
Print(">>>" + tStr + "<<<")
```

これによって以下が出力されます：

```
>>>  abcd <<<
>>>abcd<<<
```

UCase - 小文字の大文字への変換

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

UCase 関数によって、s1 文字列の小文字が大文字に変換されます。その他の文字は影響を受けません。ASCII コードテーブルに対する制限はありません。

この文字列操作関数の形式は、以下のとおりです。

```
UCase(s1 as String) as String
```

s1

変換対象の文字列を識別します。

この関数では、変換された文字列が返されます。

例：

以下の例では、テキストが大文字および小文字で表示されます。

```
Rem
Dim text As String
Text="Little ABC"

Print("LCase : " + LCase(text)) 'little abc

Print("UCase : " + UCase(text)) 'LITTLE ABC
```

Val - 文字列の整数への変換

NetWare、Symbian OS、UNIX、Windows、および Windows CE で有効です。

Val 関数によって、入力文字列が整数に変換されます。

関数の形式

Val(str as String) as Integer

str

変換対象の文字列を識別します。

この関数では、あらゆる先頭タブまたはスペースが無視され、後続に文字列があるオプション記号が識別されます。

この関数によって、入力文字列を変換した値が返されます。文字列を数字に変換できない場合、この関数では 0 が返されます。

例：

```
Dim i As Integer
i = Val("2735")
Print("Val(""2735"")+Str(i))
i=ValEx("2735")
Print("ValEx(2735)")
Print("ValEx("""2735""")+str(i))
i=ValEx("01414")
Print("ValEx("""01414""")+str(i))
i=ValEx("0x1414")
Print("ValEx("""0x1414""")+str(i))
```

ValEx - 10 進、16 進、または 8 進文字列の整数への変換

UNIX および Windows で有効です。

ValEx によって、10 進、16 進、または 8 進の入力文字列が整数に変換されます。

関数の形式

ValEx(str as String) as Integer

str

文字列では、10 進、16 進、または 8 進数が指定され、正負記号を含むことができます。指定された数字は整数に変換されます。

この関数によって、入力文字列を変換した整数値が返されます。文字列を数字に変換できない場合、この関数では 0 が返されます。

例：

```
Dim i As Integer
i = Val("2735")
Print("Val(""2735"")"+Str(i))
i=ValEx("2735")
Print("ValEx(2735")
Print("ValEx(""2735"")="+str(i))
i=ValEx("01414")
Print("ValEx(""01414"")="+str(i))
i=ValEx("0x1414")
Print("ValEx(""0x1414"")="+str(i))
```

文字列を分割する関数

ファイルまたは別のソースから読み取られた情報の解析を支援するには、以下の **DMScript** 関数を使用して、文字列を個別のトークンまたはサブ文字列に分割できます。トークンは、区切り文字（通常はスペース）によって別のサブ文字列から分離されたサブ文字列です。単引用符または二重引用符でトークンを囲むことができます。

- SetTokeniserInput
- GetToken

SetTokenizerInput

SetTokenizerInput 関数は、トークナイザを文字列で初期化し、また文字列を分割する区切り文字を定義します。別の文字列または区切り文字を指定してこの関数を呼び出すまで、スクリプトの別の部分に対してもこの文字列と区切り文字が有効です。

関数の形式

```
SetTokenizerInput(str as string, delimiters as string)
```

入力パラメータ

この関数には、以下の入力パラメータがあります。

str

分割対象の入力文字列を指定します。

注：一度に指定できる入力文字列は **1** つのみです。

delimiters

文字列を分割するための、区切り文字のセットを指定します。デフォルトでは、区切り文字としてスペース文字が使用されますが、任意の文字のセットを指定できます。たとえば、「.,/」は、**1** つのドット、カンマ、またはスラッシュによって分離されたサブ文字列のリストをトークン化します。

戻り値

なし

GetToken

GetToken 関数は、**SetTokenizerInput** によって設定された現在の入力文字列から、次の文字列トークンを取得します。

関数の形式

GetToken(token as string)as boolean

例：

```
SetTokenizerInput("This is a test program", " ")
while GetToken(token)
    print token
wend
```

入力パラメータ

この関数には、以下の入力パラメータがあります。

token

関数によって返される次のトークンを保持する文字列変数を指定します。

戻り値

次の文字列トークンを返した場合は、**True** を返します。現在の入力文字列から取得するトークンがこれ以上ない場合、**False** を返します。

出力

This is a test program.

ユーザ定義関数

このスクリプトでは、AM スクリプトで関数を使用する方法が説明されています。

```
REM
REM このスクリプトでは、AM スクリプトで関数を使用する方法が説明されています。
REM スクリプト
REM

'*****
'*
'* この関数では、2 つの整数が追加され、合計が返されます
'*
'*****

Function sum( a as Integer, b as Integer) as integer
  DIM result AS Integer

  result = a + b

  '関数値として結果が返されます
  sum = result
End Function 'sum

'*****
'*****-Main --*****
'*****

これは実際のプログラムです
Print str(Sum(3,4))           '7
Print str(Sum(234,567))      '801
```

例：スクリプト関数からの DLL の使用

このスクリプトでは、Asset Management スクリプトで関数を使用する方法が説明されています。

Rem

以下の例では、Windows NT または Windows 95/98 に壁紙を設定します。

Rem でシステム DLL 呼び出しを使用します。

Rem

#if Win32

 '32 ビットの関数の宣言

 Function SystemParametersInfo Lib "USER32.DLL" Alias "SystemParametersInfoA"
 (uiAction as Word,uiParam as Word,pvParam as String, fWinIni as Word) as Boolean
#EndIf

 'フラグの定義

Dim SPI_SETDESKWALLPAPER as integer

Dim SPIF_UPDATEINIFILE as integer

Dim SPIF_SENDWININICHANGE as integer

 'システム フラグの設定

SPI_SETDESKWALLPAPER = 20

SPIF_UPDATEINIFILE = 0x0001

SPIF_SENDWININICHANGE = 0x0002

 'システム関数の呼び出し

SystemParametersInfo(SPI_SETDESKWALLPAPER,0,"c:\%winnt%\winnt.bmp",
SPIF_UPDATEINIFILE OR SPIF_SENDWININICHANGE)