HP Asset Manager

ソフトウェアバージョン：5.11

プログラマーズリファレンス

ドキュメントリリース日：01 October 2008
ソフトウェアリリース日：October 2008

利用規約

著作権

限定保証条項

機密コンピュータソフトウェア。

所有、使用、コピーには、HPによる有効なライセンスが必要です。

FAR 12.211および12.212準拠。商用コンピュータソフトウェア、コンピュータソフトウェアマニュアル、技術データは、ベンダの標準商用ライセンスに基づき、米国政府にライセンス供与されています。

保証

HP製品およびサービスに対する保証は、当該製品またはサービスに付帯する明示的保証条項でのみ規定されます。

本規定のいかなる部分も、他の保証を構成すると解釈されるものではありません。

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

本書に含まれる内容は、予告なく変更される場合があります。

商標

- Adobe®, Adobe logo®, Acrobat® and Acrobat Logo® are trademarks of Adobe Systems Incorporated.
- Corel® and Corel logo® are trademarks or registered trademarks of Corel Corporation or Corel Corporation Limited.
- Java™ is a US trademark of Sun Microsystems, Inc.
- Microsoft®, Windows®, Windows NT®, Windows® XP, Windows Mobile® and Windows Vista® are U.S. registered trademarks of Microsoft Corporation.
- Oracle® is a registered trademark of Oracle Corporation and/or its affiliates.
- UNIX® is a registered trademark of The Open Group.

目次

I はじめに

1 プログラミングの基本

この章では、Asset Managerで使用できるBASIC言語によるプログラミングの基本を説明します。プログラミングの経験があり、他の言語を使用したことがある方にとっては、この章の内容のほとんどは既知の事項のはずです。ただし、Asset ManagerのBASICでは従来の機能のいくつかが意図的に廃止されていたり制限されていたりするので、この章全体に目を通しておくことをお勧めします。

変数の概要

変数は、プログラムの実行中にデータを格納しておくために用いられます。変数は以下の情報によって識別されます。

- 変数名。変数に格納された値を参照するために用いられます。
- 変数の型。変数に格納できるデータの種類を決定します。

一般に、変数は以下の2種類に分けられます。

- 配列。
- スカラ変数。これは配列以外のすべての変数を指します。

変数の宣言

変数は、使用する前に明示的に宣言する必要があります。宣言のシンタックスは次の通りです。

```
Dim <変数名> [As <変数の型>]
```

注意:

Asset ManagerのBASICで変数の明示的宣言が必要なのは、Microsoft Visual Basicで*Option Explicit*キーワードを使用した場合と同じです。

変数名は以下の制約を満たす必要があります。

- 先頭は英大文字または英小文字。
- 長さは40文字以内。
- 使用できる文字は、英字A～Z、a～z、数字0～9、および下線文字（「_」）です。

  注意:

  アクセント付きの文字は使用可能ですが、なるべく使用しないでください。

- 予約されたキーワードは使用できません。例えば、BASICの関数や句の名前は予約されたキーワードです。

オプションの*As*句により、定義する変数の型を指定できます。型は変数に格納できる情報の種類を示します。使用できるデータ型には、*String*（文字列）、*Integer*（整数）、*Variant*（バリアント）などがあります。

*As*句を省略した場合、変数は*Variant*型と見なされます。

単一宣言

単一宣言とは、1つの宣言文で1つの変数を宣言するものです。次の例を参照してください。

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

複合宣言

複合宣言とは、1つの宣言文で任意の数の変数を宣言するものです。次の例を参照してください。

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

注意:

すでに説明したように、変数の型を指定しないと、変数はデフォルトで*Variant*型と見なされます。このため、上の例の2行目で、変数*A*と*B*の型は*Variant*であり、*C*の型は*Integer*です。

データ型

次の表は、関数またはパラメータで使用可能な型の一覧です。

型	意味
Integer	-32768～+32767の範囲の整数。
Long	-2147483647～+2147483646の範囲の整数。
Single	4バイトの浮動小数点数（単精度）。
Double	8バイトの浮動小数点数（倍精度）。
String	テキスト。すべての文字が使用可能。
Date	日付または日付+時刻
Variant	他のすべての型の代わりとなる汎用の型。

注意:

これらの型は外部ツールからは使用できません。使用できるのは*Long*、*Double*および*String*だけです。*Variant*は存在せず、*Integer*型と*Date*型のオブジェクトは*Long*で表されます。

数値型

Asset ManagerのBASIC言語では、数値型としてInteger（整数）、Long（長整数）、Single（単精度浮動小数点数）、Double（倍精度浮動小数点数）を用意しています。数値型は*Variant*型よりも使用メモリ量が少ないのが普通です。

変数に整数（123など）だけを格納し、小数（3.14など）を決して格納しない場合には、*Integer*または*Long*として宣言するのが適しています。これらのデータ型に対する演算は、他のデータ型よりも高速で、使用メモリ量も少なくてすみます。特にループのカウンタとしては、これらのデータ型が最適です。

変数に小数を格納する場合には、*Single*または*Double*として宣言します。

注意:

浮動小数点数（*Single*または*Double*）の演算では、丸め誤差が発生する場合があります。

文字列型

変数に文字列だけを格納する場合は、*String*として宣言します。

```
Dim MyString As String
```

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

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

デフォルトでは、*String*型の変数は可変サイズです。文字列を格納するために割り当てられるサイズは、変数に代入されたデータのサイズによって変化します。ただし、次のシンタックスを使って*String*型の変数を宣言することもできます。

```
Dim <変数名> As String * <格納する文字列のサイズ>
```

次の例では、20文字の文字列変数を宣言しています。

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

この変数に20文字よりも短い文字列を格納した場合、宣言したサイズになるまで文字列の末尾にスペースが追加されます。20文字よりも長い文字列を格納した場合、21文字目から後は切り捨てられます。

バリアント型

*Variant*型は、他のすべての型の代わりとなる汎用的な型です。他のデータ型と*Variant*型との間の変換は自動的に行われます。下の例を参照してください。

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

変換は自動的に行われますが、以下の規則を守る必要があります。

- *Variant*型に対して算術演算を実行する場合、変数の内容は数値（文字列で表現されるものを含む）でなければなりません。
- *Variant*型に対して文字列連結演算を行いたい場合は、+演算子でなく&演算子を使用します。

*Variant*には2つの特殊な値を格納できます。空値と*Null*値です。

空値

*Variant*変数に初めて値を代入するまでは、変数の内容は空値です。これは特別な値であり、0、空文字列、*Null*のどの値とも異なります。*Variant*変数の内容が空値かどうかをテストするには、BASIC関数**IsEmpty()**を使用します。次の例を参照してください。

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

空値を持つ*Variant*変数を式で使用することは可能です。状況に応じて、値は0または空文字列と解釈されます。*Variant*変数に空値を再代入するには、キーワード*Empty*を使用します。次の例を参照してください。

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

Null値

*Null*値は、データベースにおいて、存在しない値または未知の値を表すためによく用いられます。この値には以下のような特別な性質があります。

- *Null*値を含む式の値は常に*Null*値になります。これを、式の中で*Null*値が伝搬するといいます。式の一部が*Null*なら、式全体が*Null*となります。
- 一般的に、関数のパラメータを*Null*に設定すると、関数の戻り値は*Null*になります。

データ配列

配列は、複数の変数を1つの名前で格納して参照するためのものです。配列の中の1つの変数を識別するには、番号（添字）を使います。配列の要素はすべて同じ型になります。*String*と*Double*の両方の型を含む配列を作成することはできません。このような場合、*Variant*型を使用すれば、両方の値を格納する配列を作成できます。

配列の宣言

配列は変数の集合です。

配列に関して以下の規約が採用されています。

- 配列の下限：最初の要素の添字。

  注意:

  デフォルトでは、配列の下限は0です。

- 配列の上限：最後の要素の添字。

  注意:

  配列の上限は、*Long*型の最大値（2147483646要素）を超えることはできません。

配列の宣言は、変数の宣言に似ています。

```
Dim <配列名>(<配列の上限>) [As <配列に含まれる変数のデータ型>]
```

例：

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

次の宣言を使って、配列の下限を指定することもできます。

```
Dim <配列名>(<配列の下限> To <配列の上限>) [As <配列に含まれる変数のデータ型>]
```

例：

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

制限

Asset ManagerのBASICでは、配列に以下の制限があります。

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

制御構造

制御構造とは、その名の示すとおり、プログラムの実行を制御するためのものです。制御構造には以下の2種類があります。

- 決定構造：何らかの条件に基づいてプログラムの行き先を決定するもの。
- ループ構造：何らかの条件に基づいてプログラムの特定の部分を反復するもの。

決定構造

決定構造は、テストの結果に基づいて特定の命令を条件付きで実行します。以下の決定構造が使用できます。

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

If...Then

この構造は、1つまたは複数の命令を条件付きで実行するために使用します。この構造のシンタックスでは、1行または複数行の文が使用できます。1行の文は1つの命令だけを実行できます。

```
If <条件> Then <命令>
```

```
If <条件> Then
<命令群>
End If
```

条件は一般に比較式ですが、結果が数字で表される式はどれでも使用できます。この値は次にBasicにより**True**または**False**として解釈されます。**False**は数値0に該当し、その他の値は**True**とみなされます。

条件が**True**に評価されると、キーワード**Then**のあとの命令または命令群が実行されます。

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

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

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

最初の条件がテストされ、結果が**False**に評価された場合、2番目の条件がテストされ、以下同様に、どれかの条件が**True**に評価されるまで続きます。その条件の**Then**キーワードの後の命令群が実行されます。

Elseキーワードは省略可能です。これは、すべての条件が**False**に評価された場合に実行される命令群を指定するために使います。

注意:

この決定構造のElseIf**命令群はいくつでも重ねることができます。ただし、同じ式を連続的に異なる値と比較する場合、決定構造が無意味に複雑になり、読みにくくなるおそれがあります。このような場合は、**Select...Case**決定構造を使用することをお勧めします。**

Select...Case

この構造は上の決定構造と働きは同じですが、一般的にコードが読みやすくなる効果があります。**Select...Case**機能は、構造の開始時に1回だけテストを実行し、テスト結果を各**Case**で指定された値と比較します。値が一致すると、その**Case**に対応する命令群が実行されます。

```
Select Case <テスト>
[Case <値リスト1>
<命令群1>]
[Case <値リスト2>
<命令群2>]
```

```
...
[Case Else
<命令群n>]
End Select
```

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

Case Elseキーワードに対応する命令群は、**Case**キーワード内に一致する値が見つからなかった場合に実行されます。

ループ構造

ループ構造は、一覧の命令を繰り返し実行するために使用します。以下のループ構造が使用できます。

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

Do...Loop

この構造は、一連の命令を不定の回数繰り返す場合に使用します。条件が満たされたとき、あるいは満たされなかったときにループが終了します。この条件は、**False**（0）または**True**（0以外）に評価される値または式です。


注意:

命令群の中でExit Do**を実行することにより、ループを強制的に終了させることができます。**

この構造にはいくつかの形式がありますが、最も普通に用いられるのは次のものです。

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

この場合、最初に条件が評価されます。値が**True**なら、命令群が実行され、プログラムは**Do While**キーワードに戻り、もう一度条件をテストし、以下同様に続きます。条件が**False**に評価されるとループは終了します。

次の例は、カウンタの値をテストし、ループの1回の反復ごとにカウンタを1ずつ増やします。カウンタが20になるとループは終了します。

```
Dim iCounter As Integer
iCounter = 0
Do While iCounter < 20
```

```
iCounter = iCounter +1
Loop
```

次の例は前の例と似ていますが、カウンタの値が10になったときに**Exit Do**キーワードでループを強制終了します。

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

この形式の**Do...Loop**構造では、命令群が実行される前に条件が評価されます。命令を実行してから条件をテストしたい場合は、次の**Do...Loop**構造を使用します。

```
Do
<命令群>
Loop While <条件>
```

注意:

この形式の構造では、命令群が少なくとも1回は必ず実行されます。

前の2つの**Do...Loop**構造では、条件が**True**である間反復が行われます。これに対して、条件が**False**の間反復したい場合、以下の構造を使用します。

```
Do Until <条件>
<命令群>
Loop

Do
<命令群>
Loop Until <条件>
```

この形式の構造を使うと、前の例は以下のように書くことができます。

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

For...Next

この構造は、一連の命令を不定の回数繰り返す場合に使用します。**Do...Loop**と異なり、**For...Next**ループではカウンタと呼ばれる変数を使用し、反復のたびにカウンタの値を加算または減算します。

注意:

命令群の中でExit For**を実行することにより、ループを強制的に終了させることができます。**

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

重要項目:

引数カウンタ、初期値、最終値、増分値は、すべて数値です。

注意:

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

For...Nextループが実行されると、以下の動作が行われます。

1. カウンタが初期化され、初期値が格納されます。
2. カウンタの値が最終値より大きいかどうかがテストされます。結果が真なら、ループは終了します。

   注意:

   増分値が負の場合は、カウンタの値が最終値より小さいかどうかがテストされます。
3. 命令群が実行されます。
4. カウンタに1または指定された増分値が加算されます。
5. 動作2から4までが繰り返されます。

次の動作は、1000までのすべての偶数を合計します。

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

次の例は前の例と似ていますが、カウンタの値が500になったときに**Exit For**キーワードでループを強制終了します。

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

演算子

演算子とは、単純な演算（加算、乗算等）を変数に対して実行したり、変数を比較したりするときに使用する記号です。演算子には以下の種類があります。

- 代入演算子
- 算術演算子
- 関係演算子（比較演算子とも呼ぶ）
- 論理演算子

代入演算子

この種類の演算子は、変数に値を代入するために使用します。Asset ManagerのBASICでは、代入演算子は「=」の1つだけです。代入のシンタックスは次の通りです。

```
<変数> = <値>
```

算術演算子

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

+演算子

この演算子は、2つの値を加算するために用いられます。シンタックスは次の通りです。

```
<結果> = <式1> + <式2>
```

注意:

この演算子は、2つの値の加算と、2つの文字列の連結の両方に用いられます。あいまいさを避けるため、この演算子は加算だけに使い、文字列の連結には&演算子を使用することをお勧めします。

-演算子

この演算子は、2つの値の差を求めるため、または1つの値の符号を反転するため（単項演算子）に用いられます。この演算子には2通りのシンタックスがあります。

```
<結果> = <式1> - <式2>
```

または

```
- <式>
```

*演算子

この演算子は、2つの値を乗算するために用いられます。シンタックスは次の通りです。

```
<結果> = <式1> * <式2>
```

/演算子

この演算子は、2つの値を除算するために用いられます。シンタックスは次の通りです。

```
<結果> = <式1> / <式2>
```

^演算子

この演算子は、値の累乗を求めるために用いられます。シンタックスは次の通りです。

```
<結果> = <式1> ^ <式2>
```

注意:

このシンタックスでは、式2（指数）が整数の場合は式1が負の値を取ることはできません。式の中で複数の指数演算が連続して行われる場合、左から右の順番で論理的に解釈されます。

Mod演算子

この演算子は、2つの値のユークリッド除算の剰余を計算します。シンタックスは次の通りです。

```
<結果> = <式1> Mod <式2>
```

注意:

浮動小数点数は自動的に整数に丸められます。

次の例は4を返します（6.8は最も近い整数である7に丸められます）。

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

関係演算子

関係演算子は、2つの値を比較するために使用します。下の表は関係演算子の一覧です。

演算子	名称	説明	シンタックス
=	等価演算子	2つの値を比較し、等しいかどうかを調べます。	<式1> = <式2>
<	「小なり」演算子	ある値が別の値より厳密に小さいかどうかをテストします。	<式1> < <式2>
<=	「小なりまたは等しい」演算子	ある値が別の値より小さいかまたは等しいかどうかをテストします。	<式1> <= <式2>
>	「大なり」演算子	ある値が別の値より厳密に大きいかどうかをテストします。	<式1> > <式2>
>=	「大なりまたは等しい」演算子	ある値が別の値より大きいかまたは等しいかどうかをテストします。	<式1> >= <式2>
<>	不等価演算子	ある値が別の値と異なるかどうかをテストします。	<式1> <> <式2>

論理演算子

論理演算子は、複数の条件を評価するために使用します。

And演算子

この演算子は、2つの式の論理積（両方の条件が真）演算を実行します。シンタックスは次の通りです。

```
<結果> = <式1> And <式2>
```

すべての式（オペランド）が*True*に評価された場合、結果は*True*です。どちらかの式が*False*に評価された場合、結果は*False*です。

Or演算子

この演算子は、2つの式の論理和（どちらかの条件が真）演算を実行します。シンタックスは次の通りです。

```
<結果> = <式1> Or <式2>
```

どちらかの式が*True*に評価された場合、結果は*True*です。

Xor演算子

この演算子は、2つの式の排他的論理和（2つの条件が一方だけが真）演算を実行します。シンタックスは次の通りです。

```
<結果> = <式1> Xor <式2>
```

一方の式だけが*True*に評価された場合、結果は*True*になります。

Not演算子

この演算子は、式の論理否定を実行するために用いられます。シンタックスは次の通りです。

```
<結果> = Not <式1>
```

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

演算子の優先順位

式の評価で複数の演算子を結合する場合、以下の優先順位が用いられます。演算子は優先順位の高いものから低いものへという順番で記載されています。

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

ファイル管理

Asset ManagerのBASICには、簡単なファイル管理機能があります。一般的な操作（読取り、書込みなど）が標準で用意されています。

ファイルに関する注意事項

ファイルとは、プログラムから外部のオブジェクトを参照する手段です。ファイルは論理的なレコードの集合です。レコードは構造化されている場合もされていない場合もあります。プログラムはレコードに対して基本的な操作（読取り、書込みなど）を実行できます。論理レコードは、1回の基本操作で処理できるデータの最小単位に相当します。

Asset Managerで処理できるのは、シーケンシャルファイルと呼ばれるものだけです。シーケンシャルファイルに対して実行できる主な操作は、次のレコードの読取りと、ファイルの末尾への新しいレコードの追加です。レコードの読取りと書込みを同時に行うことはできません。

読取りの場合、シーケンシャルファイルの最初の論理レコードにカーソルが置かれます。読取り操作を行うたびに、プログラムの内部領域（通常は変数）に1個のレコードが読み取られ、ファイル中の次のレコードにカーソルが移ります。読み取るレコードがまだ残っているかどうかを判定する操作（**EOF**（End Of File）句）が用意されています。

書込みの場合、シーケンシャルファイルは空か、ファイルの最後のレコードの後ろにカーソルが置かれます。書込み操作を行うたびに、プログラムの内部領域（通常は変数）のデータがファイル中のレコードに転送され、そのレコードの後ろにカーソルが移動します。


注意:

シーケンシャルファイルの主な特徴の1つは、レコードが書き込まれた順番で読み取られることです。

ファイルのオープンとクローズ

Open句

これはファイルを操作するための主要な機能です。ファイルの読取り、作成、書込みを実行できます。シンタックスは次の通りです。

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

この句のパラメータを下の表に示します。

パラメータ	説明
<ファイルのパス>	操作の対象となるファイルを指定する文字列。この文字列にはファイルのフルパスを指定できます。
<モード>	ファイルの処理モードを指定します。以下のいずれかの値を取ります。 ■ *Input*：ファイルは読取りモードでオープンされます。 ■ *Output*：ファイルは書込みモードでオープンされます。ファイルがすでに存在する場合、既存の内容は上書きされます。 ■ *Append*：ファイルは書込みモードでオープンされます。ファイルがすでに存在する場合、新しい内容はファイルの末尾に追加されます。 ■ *Binary*：ファイルはバイナリ読取りモードでオープンされます。
<アクセスタイプ>	オープンしたファイルに対して実行できる操作を指定します。ファイルが別のプロセスでオープンされており、指定したアクセスが許可されない場合、ファイルオープンコマンドは失敗します。このパラメータは以下のいずれかの値を取ります。 ■ *Read*：ファイルは読取り専用アクセスのためにオープンされます。 ■ *Write*ファイルは書込み専用アクセスのためにオープンされます。 ■ *Read Write*：ファイルは読取り／書込みモードでオープンされます。このアクセスタイプは、アクセスモードが*Binary*および*Append*のときだけ使用できます。
<ファイル番号>	ファイルを識別する1～511の固有の番号を指定します。**FreeFile()**関数を使えば、使用可能な次のファイル番号を知ることができます。

注意:

以下の点に注意してください。

- ファイルに対して読取りまたは書込み操作を行う前に、必ず**Open**句でファイルをオープンする必要があります。
- *Append*、*Binary*、または*Output*モードでは、参照したファイルが存在しない場合、新規作成されます。
- *Binary*または*Input*モードの場合、ファイルをクローズせずに、別の番号を使って同じファイルをオープンすることが可能です。*Append*または*Output*モードの場合、ファイルをクローズしてから別の番号でオープンする必要があります。

Close句

Open()でオープンしたファイルをクローズします。シンタックスは次の通りです。

```
Close [<ファイルのリスト>]
```

オプションの**<ファイルのリスト>**引数は、1つまたは複数のファイル番号のリストです。このオプション引数のシンタックスは次の通りです。

```
[[#]<ファイル番号>][,[#]<ファイルのリスト>]...
```

注意:

このパラメータを省略すると、Open()句でオープンされたすべてのアクティブなファイルがクローズされます。

ファイルからのデータの読取り

ファイルからデータを読み取るには2つの句が使用できます。どちらを使用するかは、ファイルに対して指定されたアクセスモードによります。2つの句は以下の通りです。

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

Input句

*Binary*または*Input*モードでオープンしたファイルから一定数の文字を読み取るために使用します。シンタックスは次の通りです。

```
Input (<読取り文字数>,[#]<ファイル番号>)
```

Line Input句

シーケンシャルファイルから1行を読取り、*String*または*Variant*型の変数に格納します。シンタックスは次の通りです。

```
Line Input #<ファイル番号>, <変数名>
```

重要項目:

ファイル中の文字が1文字ずつ読み取られ、復帰文字または復帰文字+改行文字が読み取られると終了します。

ファイルへのデータの書込み

ファイルにデータを書き込むには、**Print**句だけが使用できます。シンタックスは次の通りです。

```
Print #<ファイル番号>, [<データ>]
```

2 関数の分類

関数は、次の3通りの方法で分類することができます。

関数の種類

Asset Manager環境で使用できる関数は、複数のグループに分類されます。

- Asset Managerで認識される関数：主にソフトウェアのスクリプト（BASIC）を記述できる部分で使います。
- Asset Manager APIで認識される関数：外部のツールから呼び出される関数、または高度言語で書くプログラムで使う関数です。

これらの関数は、それぞれ完全に独立しているわけではありません。例えば、Asset Manager API関数の中には、ソフトウェアのBASICスクリプトで使えるものがあります。このようにAsset Managerの内部BASICスクリプトでも使えるAsset Manager API関数のことを「公開されている」関数といいます。このような関数のシンタックスが変わることはありますが、関数で実行される処理は同じです。

関数の用途

本マニュアルで説明されている関数は、少なくとも次の場所の1つで使用可能です。

- Asset Manager APIライブラリ。これらの関数は、特にGet-Itアプリケーションの開発で使用可能です。
- フィールドやリンクの設定スクリプト（ポップアップメニューで［**オブジェクトの設定**］またはAsset Manager Application Designerを選択します）、または特殊フィールドの［**計算スクリプト**］（SQL名：memScript）内。
  - デフォルト値
  - 必須属性
  - 履歴の保持
  - 読み取り専用属性
  - ...
- 「スクリプト」タイプのアクション
  - スクリプトアクションの［**アクションのスクリプト**］（SQL名：Script）フィールドで定義されるスクリプト
- Asset Managerウィザード
  - ウィザードの「FINISH.DO」スクリプト
  - ノードのプロパティの値を定義するスクリプト

アプリケーションモジュール

各関数は1つまたは複数のアプリケーションモジュールに関連付けられています。アプリケーションモジュールは、関数が実行する処理の性質を表します。以下のアプリケーションモジュールがあります。

- 組み込み：標準のBasic関数や変換関数、文字列の操作など
- テクニカル：データベースへの接続、テーブル、フィールド、リンク、インデックス、レコード、クエリの管理
- 業務：専門分野に関する一般的な関数
- ケーブル
- 調達
- 経費付替え
- ウィザード
- アクション
- グラフィックス

3 表記法と形式

本章では、次の事項について説明します。

表記法

関数のシンタックスと用例の表記法は以下の表の通りです。

規則	説明
[]	大括弧は、オプションのパラメータを示します。実際にコマンドのパラメータを入力するときは、大括弧は必要ありません。 ただし、BASICスクリプトでは、次の例のように、データへのパスを大括弧で囲みます。 *[Link.Link.Field]*
<>	山形括弧は、パラメータの短い説明（英語の略語）を示します。実際にパラメータを入力する時は、山形括弧を使わずに、括弧内にあるテキストに該当する情報のみを入力してください。
{}	中括弧は、ノードの定義または1プロパティ用の複数言語のスクリプトブロックの定義を囲みます。

規則	説明
|	縦線（パイプ文字）は、中括弧に囲まれた複数のパラメータを区切る場合に使います。

特別な意味のあるテキストの表記法は以下の通りです。

規則	説明
固定幅の文字	DOSコマンド、関数のパラメータ、および日付形式
例	コードやコマンドの例
...	コードまたはコマンドの省略
［オブジェクト名］	フィールド、タブ、メニュー、キー名など

スクリプト内での「日付+時間」型定数のフォーマット

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

yyyy/mm/dd hh:mm:ss

例

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

注意:

年月日の区切りに、ハイフン（-）を使うこともできます。

日付について

内部BASICでの日付と外部ツールからの日付は形式が異なります。

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

Duration（時間）定数の形式

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

```
RetVal=259200
```

同様に、AmWorkTimeSpanBetween()などの時間を計算する関数も、秒単位の時間を返します。

注意:

Asset Managerの会計計算では、最も一般的な計算方法を使います。この方法では、1年は12ヶ月、1月は30日として計算するので、1年は360日になります。

4 定義

本章では、主要な用語の定義について説明します。

以下の用語の定義を説明します。

関数の定義

「関数」とは、なんらかの処理を実行し、値をユーザに返すプログラムです。返される値を「戻り値」または「戻りコード」といいます。

Asset Managerの内部関数を呼び出すシンタックスの例は次の通りです。

AmConvertCurrency(strSrcName As String, strDstName As String, dVal As Double) As Double

上記と同じ関数をAsset Manager APIから呼び出すシンタックスは次の通りです。

double AmConvertCurrency(long hApiCnxBase,long ltm, const char *pszSrcName, const char *pszDstName,double dVal)

CurrentUser仮想リンクの定義

定義

CurrentUserは、すべてのテーブルから出発し、現在のユーザに対応する部署と従業員テーブル内のレコードに到達するリンクと考えられます。

- **CurrentUser**という形式を使うと、仮想リンクは現在のユーザのレコードを検索し、［会社の部署と従業員テーブル］の記述文字列を返します。
- **CurrentUser.Field**という形式を使うと、現在のユーザのフィールドの値を返します。

注意:

仮想リンクは、フィールドとリンクのリストには表示されません。そのため、CurrentUserは、Asset Manager内のスクリプトの表示／編集用ウィンドウには直接表示されません。手動で「CurrentUser」と入力する必要があります。

等価関数

AmLoginName()関数と**AmLoginId()**関数は、それぞれ現在のユーザの**「名前（SQL名：Name）」**と「ID（SQL名：lPersId）」を返し、**CurrentUser**から派生した関数と見なされます。これらの関数には、次のような関係が成り立ちます。

- AmLoginName()=[CurrentUser.UserLogin]
- AmLoginId()=[CurrentUser.lEmplDeptId]

制限

CurrentUserはコンテキストが定義されている場合のみ機能します（コンテキストはテーブルであること）。

コンテキストがない場合、別の関数を使用してください。

例：

コンテキスト外アクションを作成し、Asset Managerデータベースに接続するユーザに依存するパスを持つファイルを実行することができます。

状況依存アクションの場合、［**フォルダ**］フィールドが例えば*c:\scripts\[CurrentUser.Name]*に設定された**実行可能**なタイプアクションを作成できます。

ただし、**実行可能**なタイプアクションにコンテキストがない場合、*[CurrentUser.Name]*は固定テキストとみなされます。

したがって、スクリプトを使用して**スクリプト**タイプのコンテキスト外アクションを作成するなど、別の解決策を見つける必要があります。

```
RetVal = amActionExec("program.exe","c:\scripts\" + amLoginName())
```

ハンドルの定義

「ハンドル」とは、オブジェクト固有の識別子です。Asset Managerでは、フィールド、リンク、インデックス、クエリ、レコード、テーブル、または接続のことをオブジェクトと呼びます。ハンドルは、32ビット整数型（Long型）の値を取ります。

注意:

NULLは、有効なハンドルの値ではありません。

外部ツールからも（データベースの）接続ハンドルにアクセスできます。

エラーコードの定義

関数を正しく実行できなかった場合は、エラーコードが返ります。

外部ツールの場合

外部ツールでは、AmLastError()関数とAmLastErrorMsg()関数を使って、それぞれエラーコードとそのコードに付いているエラーメッセージを取得することができます。エラーをクリアするには、AmClearLastError()関数を使います。

注意:

新しい関数を呼び出すと、常にその前に発生したエラーとエラーメッセージがクリアされます。

内部

内部（Basicスクリプト内など）では、最後のエラーコードとエラーの説明を、**Err.Number**関数と**Err.Description**関数を使って取得します。

注意:

内部的にはエラーの管理をプログラムする必要はありません。問題のあるスクリプトは停止し、必要に応じてデータベースのロールバックが実行されます。

Err.Raise関数を使用すると、エラーメッセージを故意に発生させることができます。シンタックスは以下の通りです。

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

注意:

レコードの作成や変更が、テーブルの「有効性」に関するフィールド値により無効になる場合は、Err.Raise関数でエラーメッセージを発生させ、ユーザに知らせる方が賢明です（コード12006または12007）。エラーメッセージがないと、ユーザはレコードの変更や作成が不可能な理由を理解できません。

頻繁に使用されるエラーコードは以下の表の通りです。

エラーコード	説明
12001	未定義のエラー
12002	関数用の間違ったパラメータ
12003	無効なハンドルまたは削除されるオブジェクト
12004	使用可能なデータがない場合。このエラーは特にクエリの実行時に発生します。クエリの結果がデータを返さない場合に、このエラーが発生します。
12005	データベースサーバの内部エラー
12006	無効な値（パラメータに不適切なタイプ、など）
12007	無効なレコード（必須フィールドに値が入力されていない、など）
12008	データベースへのアクセス権限上の問題
12009	古い関数または実施されていない関数
12010	データベースへの接続最大数を超えた場合

5 関数とパラメータのデータ型

本章では、次の事項について説明します。

データ型のリスト

関数とパラメータで使用可能なデータ型とその説明は、以下の表の通りです。

データ型	説明
Integer	-32,768～+32,767の整数
Long	-2,147,483,647～+2,147,483,646の整数
Single	4バイトの浮動小数点数（単精度）。
Double	8バイトの浮動小数点数（倍精度）。
String	任意の文字からなるテキスト
Date（日付）	日付または日付+時刻

データ型	説明
Variant（可変）	汎用の任意の型

> **注意:**
>
> これらのデータ型の一部は外部ツールからは使用不可能です。Long、DoubleとString型のみが使用可能です。Variant型は存在しません。また、IntegerとDate型のオブジェクトはLong型で表現されます。

関数の型

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

例えば、フィールドのデフォルト値を定義する時に、そのフィールドのデータ型と異なる型の値を返す関数を使うことはできません。例えば、下の関数を使ってDate（日付）型やDate+Time（日付+時刻）型のフィールドのデフォルト値を定義すると、エラーが発生します。

```
RetVal=AmLoginName()
```

これは、「AmLoginName()」関数は、接続されているユーザの名前を文字列（String（文字列）型）で返すためです。文字列型の戻り値は、「Date（日付）」型や「Date+Time（日付+時刻）」型のフィールドでは使えないため、エラーメッセージが表示されます。

パラメータの型

関数で使うパラメータには型が決まっており、関数を正しく実行するためには、正しい型のパラメータを使用する必要があります。関数のシンタックス内では、パラメータの型を示す接頭辞がパラメータの先頭に付いています。このリファレンスに記載されている接頭辞は、関数のシンタックス（APIまたはBASIC）によって異なることに注意してください。それぞれAPIシンタックスとBASICシンタックスで使われている接頭辞は、次の表の通りです。

型	API関数のシンタックスで使われている接頭辞	BASIC関数のシンタックスで使われている接頭辞
Integer（整数）	"i"	"i"
Long（倍長整数）	h（ハンドル）l（数値）\n	"l"
Double（倍精度）	"d"	"d"
String（文字列）	"char*psz"	"str"

型	API関数のシンタックスで使われている接頭辞	BASIC関数のシンタックスで使われている接頭辞
Date（日付）	"ltm"	"dt"
Variant（可変）	"v"	"v"

II　APIの使用

6 はじめに

Asset Manager APIは、Windows 95/98、2000、XP、およびServer 2003上で使用可能な32ビットDLLとして用意されています。

次の環境がサポートされています。

- Visual Basic 4.0、5.0、および6.0
- Visual C++ 4.0、5.0、および6.0
- Visual Basic .NET 2002および2003
- Visual Studio .NET 2002および2003
- Visual C# .NET 2002および2003
- VBA（Visual Basic for Applications）を使用するMicrosoft全製品

警告:

ライブラリ（.dll）のエントリポイントは、.NET環境には用意されていません。これらの開発環境を使用する場合は、これらのエントリポイントをユーザ自身で定義する必要があります。

注意:

APIは、外部DLLの使用を許可するすべてのツールと互換性があります。

注意事項

Asset Manager APIを使う前に、Asset Managerの概念と用語を理解しておくことをお勧めします。特に、データベースの構造に関する知識が必要です。

データベースの構造の詳細は、Asset Managerのインストール先フォルダの「doc\infos」サブフォルダにあるマニュアル『管理』の、「データベースの標準記述ファイル」の章、および「Database.txt」と「Tables.txt」ファイルを参照してください。

インストール

Asset Manager APIを使用する前に、Asset Managerの完全インストールを実行してください。これにより、使用しているコンピュータからデータベースにアクセスできるかどうかを簡単に確認できるようになり、またデータベースの作成や設定が可能になります。APIは、Asset Managerと同じデータベース層と設定情報を使って、データソースにアクセスします。このため、APIで問題が発生した場合は、Asset ManagerのGUIで問題の原因を探すことができます。

Asset Managerでの開発環境を設定するための一般的な手順は、次の通りです。

- Asset Managerの32ビットバージョンとAsset Manager APIパッケージをインストールします。
- Asset Managerを使って、データソースを設定し、データベースを開きます。
- 開発環境で、Asset Manager API関数を呼び出します。

デモ用データベースか、エラーが発生してデータが壊れても差し支えないデータソースを使って、Asset Manager APIを試用してください。

DLLに関連付けられた「.ini」構成ファイル

▶『*Asset Manager* - **インストールとアップグレード**』マニュアルの「*.ini*および*.cfg*ファイル」の章

特に以下のセクションを参照してください。

- **使用可能な「*.ini*」および「*.cfg*」ファイル**
- ***.ini*ファイルの変更を管理する**

7 操作手順

Asset Manager APIを使用した一連のな処理手順は以下の通りです。

1 AQL文を使ってクエリを作成します。

```
SELECT AssetTag, User.Name, Supervisor.Name FROM amPortfolio
```

> **注意:**
>
> Asset Manager Export Toolを使ってAQLクエリを作成することもできます。

2 クエリの結果を検索し、特定の項目の必要なハンドルを取得します。

3 取得したハンドルを使って、レコード情報を更新します。

4 トランザクション全体をコミット（完了）するか、ロールバック（キャンセル）します。

8 基本概念とプログラムの例

本章では、次の事項について説明します。

基本概念

Asset Managerは、オブジェクト指向で設計されており、このためAPIもオブジェクト指向型の構造です。Windows DLLには、単層構造の「C言語型」APIを使用するという制約点がありますが、Asset Manager APIは、この問題を回避するためにハンドル（32ビット整数）を使ってユーザが作成した各オブジェクトを識別します。このため、オブジェクト指向言語以外のプログラムからでも、Asset Managerオブジェクトモデルにアクセスできるようになっています。

使用するプログラムは、Asset Manager DLLを初期化するために、最初にAmStartUp()関数を呼び出さなければなりません。また、最後にAmCleanUp()関数を呼び出して終了する必要があります。

データベースオブジェクトにアクセスする前に、ユーザとデータベースの接続を確立する必要があります。この接続は、「接続」オブジェクトの「ハンドル」で識別されます。このハンドルは、その後データベースと相互作用するすべてのAPI関数で使うことができます（ハンドルは「hApiCnxBase」パラメータで指定

します。）「接続」オブジェクトは、クエリを作成し、レコードにアクセスするのに使用されます。

注意:

すべてのデータベースオブジェクトが接続オブジェクトにリンクされるので、ユーザのアクセス権限情報をチェックできます。

接続が確立したら、まず有効なデータソース名とログイン名／パスワードを使って、接続を開きます。

警告:

Asset Manager APIを介してAsset Managerデータベースに接続するときは、スロットを使います。

日付と時刻の処理

Date（日付）型とDate+Time（日付+時刻）型のフィールドの値を読み取るには、次のいずれかの関数を使います。

- AmGetFieldLongValue()：UNIXで使うLong（倍長整数）型のUTC（協定世界時）を返します。
- AmGetFieldStrValue()：Windowsのコントロールパネルで設定されている形式と同じ文字列を返します。Windowsで指定しているタイムゾーンに合わせた日付と時刻が返ります。戻り値を表示する必要がある場合には、この関数を使うことをお勧めします。

プログラムの例（1）

デモ用データベースへの接続を作成するC言語プログラムの例は、次の通りです。

```
long lCnx ;
lCnx = AmOpenConnection(AMDemo51en, Admin , ) ;
```

この例のlCnxは、新しく作成する接続の識別に使用される接続ハンドルです。

次に、作成した接続を使ってクエリを作成したり、データベースにアクセスしたりできます。特定の資産を検索するC言語プログラムの例は、以下の通りです。

```
#include apiproto.h
#define SZ_MODEL_LEN 200
long lCnx ;
```

```
long lQuery ;
long lStatus ; /* to store error code */
char szModel[SZ_MODEL_LEN] ;
/* dll initialization */
AmStartup();
/* Open a connection */
lCnx = AmOpenConnection("AMDemo51en","Admin" ,"") ;
if( lCnx != 0 )
{
/* Creation of a query object */
lQuery = AmQueryCreate (lCnx)
if( lQuery != 0 )
{
/* Construction of  the result set : all assets from Compaq*/
lStatus = AmQueryExec(lQuery, "select AssetTag where brand = 'Compaq'")
/* Navigates through the result set */
while( !lStatus )
{
/* Read the first field (AssetTag) of the current item in the query */
lStatus = AmGetFieldStrValue(lQuery,0,szModel,SZ_MODEL_LEN-1);
if( lStatus == 0 )
{
printf(' Compaq AssetTag=%s\n',szModel);
lStatus = AmQueryNext(lQuery);
}
}
/* clean things up */
AmReleaseHandle(lQuery);
}
AmCloseConnection(lCnx);
}
AmCleanup();
```

プログラムの例（2）

データベース内のオブジェクトを見つけるにはクエリを使います。レコードを更新するには、まずクエリを使って"record"オブジェクトのハンドルを取得する必要があります。次にそのレコードを他のAsset Manager API関数で処理します。

特定のレコードハンドルを取得して、フィールドを変更するプログラムの例は、以下の通りです。

```
/* Handles for objects */
long lCnx ;
long lQuery ;
long lStatus ;
long lRecord ;
AmStartup();
lCnx = AmOpenConnection("AMDemo51en","Admin" ,"") ;
/* Creation of a query object attached to lCnx */
lQuery = AmQueryCreate(lCnx);
/* Mark the starting point of the current transaction */
AmStartTransaction(lCnx);
/* Use a query that matches a single object */
lStatus = AmQueryGet(lQuery, "select model, AssetId where brand = 'Compa
q' and barcode='34234'") ;
/* Get a record handle to the matching object */
lRecord = AmGetRecordHandle(lQuery) ;
/* Change the field Field1 with new value spam */
lStatus = AmSetFieldStrValue(lRecord, "Field1", "Spam");
/* Update the change for the current session */
lStatus = AmUpdateRecord(lrecord);
/* Commit all modifications to the database */
lStatus = AmCommit(lCnx) ;
/* you can release here query and record objects */
/* but closing connection will do it */
/* Close the connection to the database */
AmCloseConnection(lCnx);
AmCleanup();
```

この例では、クエリを使ってレコード特有の識別子（ハンドル）を取得する方法を示しています。ここで使用したクエリでは、オブジェクトを1つだけ検索しますが、AmQueryExec()を使うと、複数のレコードを検索して1つまたは複数のレコードハンドルが返るようにすることもできます。

注意:

この例を簡単にするため、発生する可能性のあるエラーコードは含まれていません。

III Webサービス

9 Webサービス

Asset Manager Webサービスについて

Asset ManagerはWebサービスを公開できます。

これには、Asset ManagerはSOAPプロトコルを使用します。

公開されたWebサービスにより、Asset Managerサーバと容易に通信できます。

これにより、読み込みアクション（*retrieveAllPurchaseRequest*など）、および書き込みアクション（*savePurchaseRequest*など）を実行できます。

これらのアクションは、Webサービスと対話できるMicrosoft Studio 2003 ASP.Net、Java + Antやその他のあらゆるツールを介することで実行できます。


注意:

Asset Managerは、**サードパーティ製Webサービスを使用する（呼び出す）ことはできません。**

HP Connect-Itを使用して、**サードパーティ製Webサービスを呼び出すことはできません。**

Asset Manager Web Serviceが公開するWebサービスは、機能ドメインに従ってグループ化されます（**［WEBサービス］**（seWebService）フィールドが**［スタンドアロン］**である機能ドメインのみが保持されます）。これらには、**［seWebService］**フィールドが**［親ドメインから］**である機能サブドメインが含まれます。

Webサービスは、Asset Managerデータベースのオブジェクト（画面、アクションなど）を公開します。

公開されたWebサービスには、多数のAPIを含めることができます。

特定のWebサービスの定義にアクセスするには、次のようなURLを入力します。

```
http://<Asset Manager Web Serviceサーバの名前>:<Asset Manager Web Serviceのポート番号>/AssetManagerWebService/services/Head/<Webサービスの名前>?WSDL
```

<*Web***サービスの名前**>は、［**WEBサービス**］（seWebService）フィールドが［**スタンドアロン**］である機能ドメインのSQL名に対応します。

Webサービスの使用方法に関する詳細については、『▸ Webサービスの呼び出しに使用するコード例 [献 56]』を参照してください。

Webサービスの定義の確認

ある時点でのWebサービスによって操作されるオブジェクトのステータスを追跡するため、タグ付けを行います。タグ付けを行うことで、データベースにアクセスするのにWindowsクライアントを経由しないアプリケーションに、ある時点でのデータベースのステータスに対応するオブジェクトを使用させることができます。

Webサービスのタグ付けの詳細については、▸『**カスタマイズ**』の「*Customizing the database*（**データベースのカスタマイズ**）」の章、「*Development best practices*/ *Tag the Web services*（**開発のベストプラクティス**/*Web***サービスのタグ付け**）」を参照してください。

API命名規則

重要項目:

Webサービスが公開するAPIは、レコードではなくドキュメントでまとめられます。

***PurchaseRequest*ドキュメントには、関連依頼明細のすべてが含まれます。**

Webサービスが公開するAPIの命名規則のリストを以下に挙げます。

- *retrieveAllXxxListByYyy*

  *Yyy*でフィルタされる*Xxx*タイプドキュメントのリストを取得します。

  Xxxは画面のSQL名から生成されます。

  Yyyは次の項目から生成されます：

- インデックスを構成するフィールドとリンクのSQL名（*AssetAnddCntrIncluded*など）
- クエリのSQL名
- クエリウィザードの画面セット（QBEフィールド）

*retrieveAllPurchaseRequestListByUser*など

- *retrieveFirstXxxListByYyy*

  *Yyy*でフィルタされるn個の*Xxx*タイプドキュメントのリストを取得します（nはAPIのパラメータ）。
- *retrieveNextXxxList*

  ドキュメントをパラメータとして渡した後に、n個の*Xxx*タイプドキュメントのリストを取得します。
- *retrievePreviousXxxList*

  ドキュメントをパラメータとして渡す前に、n個の*Xxx*タイプドキュメントのリストを取得します。
- *retrieveLastXxxListByYyy*

  *Yyy*でフィルタされる、最後のn個の*Xxx*タイプドキュメントのリストを取得します（nはパラメータとして渡されます）。
- *retrieveXxxByYyy*

  *Yyy*でフィルタされる1個の*Xxx*タイプドキュメントを取得します。
- *retrieveXxx*

  パラメータとして渡されるAPI参照から、1個の*Xxx*タイプドキュメントを取得します。
- *saveXxx*

  1個の*Xxx*タイプドキュメントを保存します。
- *deleteXxx*

  1個の*Xxx*タイプドキュメントを削除します。
- *countXxx*

  パラメータとして渡されたリストに対応する*Xxx*タイプドキュメントの個数を数えます（メモリに読み込まれたリストのサイズには制限されません）。
- *retrieveXxxBreakdown*

  パラメータとして渡されたリストに対応する*Xxx*タイプドキュメントの内訳を生成します（メモリに読み込まれたリストのサイズには制限されません）。
- *executeZzz*

  *Zzz*アクションを実行します。

Webサービスの呼び出しに使用するコード例

バージョン5.01には、Asset Manager Webサービスを呼び出すコードがあるサンプルコードが備わっています。

これらのプロジェクトは、Asset Managerインストールフォルダの「samples\ws」フォルダに配置されています。

これらのプロジェクトは、以下の環境で設計されました。

Microsoft Studio 2003 ASP.Net

- *RequestSample*

  C# ASP.Netプロジェクト（購入依頼のリストを表示するため、および依頼を作成するために使用）。

- *ChartingSample*

  このVB.Net WindowsFormsプロジェクトは、コストタイプ別の経費明細の内訳を示すグラフを表示するのに使用されます。

  注意:

  このコードは、次の場所からダウンロードできる*DotNetCharting*をインストールする必要があります。

  http://www.dotnetcharting.com/download.aspx

- *ACPhoneListSample*

  このC# WindowsFormsプロジェクトは、ページング機能を使用してAsset Managerデータベースディレクトリを表示するのに使用されます（レコードは一括してすべてが返されるのではなく、グループ別に返されます）。

Java + Ant

- *RSS*

  このプロジェクトは、RSS（Really Simple Syndication）フィードを経由して接続したユーザに割り当てられている、ニュースとワークフロータスクを表示するのに使用されます。

注意:

RSSフォーマットとは、Webサイトのコンテンツ（記事、情報、イベント）を表示するための方法であり、通常は定期的に更新されるコンテンツを提供するあらゆるページのことを指します。

この機能により、Webサイトは他のサイトに公開されている最新情報を自動的に表示できます。

RSSフォーマットは、Webサイト間でコンテンツを共有するのに使用されます。

RSSコンテンツは、専門のRSSフィードリーダであるアグリゲータで読み取れます。

- *CoreServiceSample*
  このプロジェクトは、DOSコンソールでデモデータベースにある従業員と部署のリストを表示するのに使用されます。

WSDLの呼び出しを使用したFlashの開発の制限

注意:

ここで説明する回避策は、Flash 8でテストしています。

Asset Manager WSDLを呼び出すFlash 8アプリケーションを使用する場合、回避策を実装する必要があります。

以下の手順を実行してください。

1 Internet Explorerを起動します。

2 Asset Manager Web Serviceページにアクセスします（http://<Asset Manager Web Serviceサーバ名>:<Asset Manager Web Serviceのポート番号>/AssetManagerWebService）。

3 Flashアプリケーションを実装するWebサービスの対象リビジョンを表示します。

*R50*などです。

4 各ドメイン（［管理］など）で以下の手順を実行します。

a ［schema］リンクをクリックします。

表示されているドキュメントをローカルフォルダ（「C:\FlashDev\schema\R50\Administration\Administration.wsdl」など）に保存します。

b ［wsdl］リンクをクリックします。

表示されているドキュメントをローカルフォルダ（「C:\FlashDev\schema\R50\Administration\AdministrationTypes.xsd」など）に保存します。

c 「.wsdl」および「.xsd」ファイルを開きます。

先頭が*schemaLocation*=の行（*schemaLocation*="../../*schema*/*R50*/*Administration*/*AdministrationTypes.xsd*"など）を変更します。

相対パス../../を絶対パスに置き換えます。

例：

```
schemaLocation="file:///C|/FlashDev/schema/R50/Administration/AdministrationTypes.xsd"
```

d ローカルWSDLを使用してFlashオブジェクトを開発します。

HP Service ManagerからのAsset Manager WSDLの呼び出し：制限

Asset Manager WSDLには、HP Service Managerでサポートされていない定義が含まれています。

これにより、HP Service Managerからの呼び出しを防ぎます。

接続プールの再初期化

次のいずれかの操作を実行した場合：

- WindowsまたはWebクライアントで、**［リストデータの値］**（amItemListVal）、または**［リストデータ］**（amItemizedList）テーブルで、レコードの追加、変更、削除を実行
- ウィザードの*DBLISTBOX*コントロールの*ColName*プロパティにソーステーブルの画面に使用されるデフォルト列の一部ではない列（Asset Manager Application Designer／Detail of the source table（ソーステーブルの詳細）／Detail of the screens（画面の詳細）／**［リスト／詳細］**タブ／**［Columns of the list（リストの列）］**、および**［Other columns（その他の列）］**フィールド）を追加

Asset Manager Web Serviceを使用して接続プールを再初期化し、これらの操作がWebクライアントに反映されるようにする必要があります。

1 Asset Manager Web Serviceを開始します。

```
http://<Asset Manager Web Serviceサーバの名前、またはIPアドレス>:<Asset Manager Web Serviceのポート番号>/AssetManagerWebService
```

2 Reset the connection poolリンクをクリックします。
この操作を実行するには、管理者権限が必要です。

ヒント:

これらのレコードはリフレッシュする必要があるキャッシュに格納されるため、この操作を実行する必要があります。

IV 関数の説明

10 関数の説明

Abs()

数値の絶対値を返します。

内部Basicシンタックス

Function Abs(dValue As Double) As Double

用途

バージョン：*3.00*

	使用
AssetManager API	
リンクまたはフィールドの設定スクリプト	✓
「スクリプト」タイプのアクションの設定	✓
ウィザードスクリプト	✓
ウィザードの「*FINISH.DO*」スクリプト	✓

パラメータ

- **dValue**：絶対値を取得する数値

戻りコード

エラーの場合は、次の2つの処理方法を用意します。

- Asset Manager内で、この関数を含むスクリプトを中断し、ユーザの画面にエラーメッセージを表示します。
- 外部プログラムからこの関数を呼び出した場合は、AmLastError() [献 ?]関数（必要に応じてAmLastErrorMsg() [献 ?]関数も）を呼び出し、エラーが発生したかどうかを確認し、エラーが発生した場合はエラーメッセージを返すようにします。

例

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

AmActionDde()

DDEサーバアプリケーションにDDE要求を送ります。この関数を使うと、DDEリンクにより、Asset Managerから他のアプリケーションをコントロールできます。この関数はDDEタイプのアクションに相当します。

APIシンタックス

long AmActionDde(long hApiCnxBase, char *strService, char *strTopic, char *strCommand, char *strFileName, char *strDirectory, char *strParameters, char *strTable, long lRecordId);

内部Basicシンタックス

Function AmActionDde(strService As String, strTopic As String, strCommand As String, strFileName As String, strDirectory As String, strParameters As String, strTable As String, lRecordId As Long) As Long

用途

バージョン：*3.00*

	使用
AssetManager API	✓
リンクまたはフィールドの設定スクリプト	
「スクリプト」タイプのアクションの設定	✓
ウィザードスクリプト	✓
ウィザードの「*FINISH.DO*」スクリプト	✓

パラメータ

- **strService**：呼び出す実行可能ファイルのDDEサービスの名前。実行可能ファイルのDDEサービスの一覧については、その実行可能ファイルのマニュアルを参照してください。
- **strTopic**：実行するDDEアクションのトピック（コンテキスト）
- **strCommand**：実行する外部アプリケーションのコマンド。外部アプリケーションで定義されているシンタックスに従って指定する必要があります。
- **strFileName**：サービスがメモリにない場合は、サービスを起動する実行可能ファイルの名前（または、Windowsのファイルマネージャで実行可能ファイルとして設定されているファイルの名前）をこのパラメータで指定して、サービスを起動する必要があります。
- **strDirectory**：**strFileName**で指定したファイルのパス
- **strParameters**：サービスの起動時にサービスをアクティブにする実行可能ファイルに渡す様々なパラメータ
- **strTable**：アクションのコンテキストを指定する時に追加するパラメータ。アクションを適用するレコードが入っているテーブルのSQL名を示します。
- **lRecordId**：アクションが状況依存である場合に使用する、オプションのパラメータ。アクションを適用するレコードの識別子を指定します。

戻りコード

- 0：成功
- 0以外：エラーコード

AmActionExec()

「.exe」、「.com」、「.bat」、「.pif」のいずれかのアプリケーションを起動します。また、Windowsのファイルマネージャで実行可能ファイルとして設定している拡張子の文書を参照できます。この関数は、「実行可能ファイル」タイプのアクションに相当します。

APIシンタックス

long AmActionExec(long hApiCnxBase, char *strFileName, char *strDirectory, char *strParameters, char *strTable, long lRecordId);

内部Basicシンタックス

Function AmActionExec(strFileName As String, strDirectory As String, strParameters As String, strTable As String, lRecordId As Long) As Long

用途

バージョン：*3.00*

	使用
AssetManager API	✓
リンクまたはフィールドの設定スクリプト	
「スクリプト」タイプのアクションの設定	✓
ウィザードスクリプト	✓
ウィザードの「*FINISH.DO*」スクリプト	✓

パラメータ

- **strFileName**：以下のリストは、WindowsとWebの各クライアント用の機能が、状況に応じて取り得る動作をまとめたものです。：
  - *http*または*https*：
    - *Windows*クライアントがトリガする機能：クライアントのWindowsワークステーション上でデフォルトインターネットブラウザを起動し、**strFileName**パラメータが与えるURLアドレスにアクセスします。
    - *Web*クライアントがトリガする機能：Webクライアントのワークスペース内に、**strFileName**パラメータが指定するURLに対応するページを表示します。
  - *ftp*：
    - *Windows*クライアントがトリガする機能： クライアントのWindowsワークステーション上でエクスプローラを起動し、**strFileName**パラメータが与えるFTPサイトに接続します。
    - *Web*クライアントがトリガする機能：Webクライアントのワークスペース内に、**strFileName**パラメータが指定するFTPサイトを表示します。
  - *mailto*：

- *Windows*クライアントがトリガする機能：クライアントのWindowsワークステーション上でデフォルトのメッセージングアプリケーションを起動し、**strFileName**パラメータのemphasis>mailto:

<listitem> priority="2">
*Web*クライアントがトリガする機能：
</listitem>

<listitem> priority="2">
その他の値：

- *Windows*クライアントがトリガする機能：**strFileName**パラメータが指定するファイルを実行します。
- *Web*クライアントがトリガする機能：アクションの動作は［アクション／実行可能タイプのアクションの実行］（ExecuteAction）データベースオプション（Windowsクライアントの［管理／データベースオプション...］メニュー）の値によって異なります：
  - **なし**：ファイルはAsset Manager Web Serviceステーション上では実行されず、エラーが返されます。
  - **サーバ**：ファイルはAsset Manager Web Serviceステーション上で実行されます。
  - **クライアント**：Webクライアント上で実行できるアクションを説明するメッセージが表示されます。

</listitem>

<listitem> priority="2">
strDirectory：**strFileName**に指定されているファイルへのパス。
</listitem>
<listitem> priority="2">
strParameters：実行可能ファイルの起動時に実行可能ファイルに提供する各種のパラメータ（必要な場合のみ）
</listitem>
<listitem> priority="2">
strTable：アクションのコンテキストを指定する時に追加するパラメータ。アクションを適用するレコードが入っているテーブルのSQL名を指定します。
</listitem>
<listitem> priority="2">
lRecordId：アクションが状況依存である場合に使用する、オプションのパラメータ。アクションを適用するレコードの識別子を指定します。
</listitem>

10. 戻りコード

- 0：成功
- 0以外：エラーコード

10. 例

「C」ドライブの「WinNT」ディレクトリにあるWindows NT Explorerを実行します。

```
RetVal = AmActionExec("explorer.exe", "c:\winnt\")
```

V インデックス

使用可能な関数 - 全関数のリスト

- Abs
- AmActionDde
- AmActionExec
- AmActionMail
- AmActionPrint
- AmActionPrintPreview
- AmActionPrintTo
- AmAddAllPOLinesToInv
- AmAddCatRefAndCompositionToPOrder
- AmAddCatRefToPOrder
- AmAddEstimLinesToPO
- AmAddEstimLineToPO
- AmAddLicContentToRequest
- AmAddPOLineToInv
- AmAddPOrderLineToReceipt
- AmAddReceiptLineToInvoice
- AmAddReqLinesToEstim
- AmAddReqLinesToPO
- AmAddReqLineToEstim
- AmAddReqLineToPO
- AmAddRequestLineToPOrder
- AmAddTemplateToPOrder
- AmAddTemplateToRequest
- AmArchiveRecord
- AmAttribCmdAvailability
- AmBackupRecord
- AmBuildNumber
- AmBusinessSecondsInDay
- AmCalcConsolidatedFeature
- AmCalcDepr
- AmCalculateAndStoreStatistic
- AmCalculateCatRefQty
- AmCalculateReqLineQty
- AmCalculateStatistic
- AmCalculateStatisticFromSQLName
- AmCbkReplayEvent
- AmCheckTraceDone
- AmCleanup
- AmClearLastError
- AmCloseAllChildren

- AmCloseConnection
- AmCommit
- AmComputeAllLicAndInstallCounts
- AmComputeLicAndInstallCounts
- AmConnectionName
- AmConnectTrace
- AmConvertCurrency
- AmConvertDateBasicToUnix
- AmConvertDateIntlToUnix
- AmConvertDateStringToUnix
- AmConvertDateUnixToBasic
- AmConvertDateUnixToIntl
- AmConvertDateUnixToString
- AmConvertDoubleToString
- AmConvertMonetaryToString
- AmConvertStringToDouble
- AmConvertStringToMonetary
- AmCounter
- AmCreateAssetPort
- AmCreateAssetsAwaitingDelivery
- AmCreateCable
- AmCreateCableBundle
- AmCreateCableLink
- AmCreateDelivFromPO
- AmCreateDevice
- AmCreateDeviceLink
- AmCreateEstimFromReq
- AmCreateEstimsFromAllReqLines
- AmCreateInvFromPO
- AmCreateLink
- AmCreateOrUpdateInvoiceFromReceipt
- AmCreatePOFromEstim
- AmCreatePOFromReq
- AmCreatePOrderFromRequest
- AmCreatePOrdersFromRequest
- AmCreatePOsFromAllReqLines
- AmCreateProjectCable
- AmCreateProjectDevice
- AmCreateProjectTrace
- AmCreateReceiptFromPOrder
- AmCreateRecord
- AmCreateRequestToInvoice
- AmCreateRequestToPOrder
- AmCreateRequestToReceipt
- AmCreateReturnFromReceipt
- AmCreateTraceHist
- AmCreateTraceLink
- AmCryptPassword
- AmCurrentDate
- AmCurrentIsoLang
- AmCurrentLanguage
- AmCurrentServerDate
- AmDateAdd
- AmDateAddLogical
- AmDateDiff
- AmDbExecAql
- AmDbGetDate
- AmDbGetDouble
- AmDbGetLimitedList
- AmDbGetList
- AmDbGetListEx
- AmDbGetLong
- AmDbGetPk
- AmDbGetString
- AmDbGetStringEx
- AmDeadLine
- AmDecrementLogLevel
- AmDefAssignee
- AmDefaultCurrency
- AmDefEscalationScheme

- AmDefGroup
- AmDeleteLink
- AmDeleteRecord
- AmDisconnectTrace
- AmDuplicateRecord
- AmEndOfNthBusinessDay
- AmEnumValList
- AmESDAddComputers
- AmESDCreateTask
- AmEvalScript
- AmExecTransition
- AmExecuteActionById
- AmExecuteActionByName
- AmExportDocument
- AmExportReport
- AmFindCable
- AmFindDevice
- AmFindRootLink
- AmFindTermDevice
- AmFindTermField
- AmFlushTransaction
- AmFormatCurrency
- AmFormatLong
- AmGeneratePlanningData
- AmGenSqlName
- AmGetCatRef
- AmGetCatRefFromCatProduct
- AmGetComputeString
- AmGetCurrentNTDomain
- AmGetCurrentNTUser
- AmGetFeat
- AmGetFeatCount
- AmGetField
- AmGetFieldCount
- AmGetFieldDateOnlyValue
- AmGetFieldDateValue
- AmGetFieldDescription
- AmGetFieldDoubleValue
- AmGetFieldFormat
- AmGetFieldFormatFromName
- AmGetFieldFromName
- AmGetFieldLabel
- AmGetFieldLabelFromName
- AmGetFieldLongValue
- AmGetFieldName
- AmGetFieldRights
- AmGetFieldSize
- AmGetFieldSqlName
- AmGetFieldStrValue
- AmGetFieldType
- AmGetFieldUserType
- AmGetForeignKey
- AmGetIndex
- AmGetIndexCount
- AmGetIndexField
- AmGetIndexFieldCount
- AmGetIndexFlags
- AmGetIndexName
- AmGetLink
- AmGetLinkCardinality
- AmGetLinkCount
- AmGetLinkDstField
- AmGetLinkFeatureValue
- AmGetLinkFromName
- AmGetLinkType
- AmGetMainField
- AmGetMemoField
- AmGetNextAssetPin
- AmGetNextAssetPort
- AmGetNextCableBundle

- AmGetNextCablePair
- AmGetNTDomains
- AmGetNTMachinesInDomain
- AmGetNTUsersInDomain
- AmGetPackageNames
- AmGetPOLinePrice
- AmGetPOLinePriceCur
- AmGetPOLineReference
- AmGetRecordFromMainId
- AmGetRecordHandle
- AmGetRecordId
- AmGetRelDstField
- AmGetRelSrcField
- AmGetRelTable
- AmGetReverseLink
- AmGetScreenSetsNames
- AmGetScriptValue
- AmGetSelfFromMainId
- AmGetSerialModifiedPages
- AmGetSerialNbFilters
- AmGetSourceTable
- AmGetTable
- AmGetTableCount
- AmGetTableDescription
- AmGetTableFromName
- AmGetTableLabel
- AmGetTableName
- AmGetTableRights
- AmGetTableSqlName
- AmGetTargetTable
- AmGetTrace
- AmGetTraceFromHist
- AmGetTypedLinkField
- AmGetUserEnvSessionItem
- AmGetVersion
- AmGetViewModifiedPages
- AmGetViewNbFilters
- AmHasAdminPrivilege
- AmHasRelTable
- AmHasRightsForCreation
- AmHasRightsForDeletion
- AmHasRightsForFieldUpdate
- AmHelpdeskCanCloseFile
- AmHelpdeskCanProceed
- AmHelpdeskCanSaveCall
- AmImportDocument
- AmImportReport
- AmIncrementLogLevel
- AmInsertRecord
- AmInstantiateReqLine
- AmInstantiateRequest
- AmIsConnected
- AmIsExistingPage
- AmIsExistingScreen
- AmIsFieldForeignKey
- AmIsFieldIndexed
- AmIsFieldPrimaryKey
- AmIsFilterModifInSerial
- AmIsFilterModifInView
- AmIsHelpdeskAdmin
- AmIsHelpdeskMember
- AmIsHelpdeskSuper
- AmIsLink
- AmIsModuleAuthorized
- AmIsTypedLink
- AmLastError
- AmLastErrorMsg
- AmListToString
- AmLog
- AmLoginId

- AmLoginName
- AmMapSubReqLineAgent
- AmMoveCable
- AmMoveDevice
- AmMsgBox
- AmNbLanguages
- AmOpenConnection
- AmOpenScreen
- AmOverflowTables
- AmPagePath
- AmProgress
- AmPurgeRecord
- AmQueryCreate
- AmQueryExec
- AmQueryGet
- AmQueryNext
- AmQuerySetAddMainField
- AmQuerySetFullMemo
- AmQueryStartTable
- AmQueryStop
- AmReceiveAllPOLines
- AmReceivePOLine
- AmRefreshAllCaches
- AmRefreshLabel
- AmRefreshProperty
- AmRefreshTraceHist
- AmReleaseHandle
- AmRemoveCable
- AmRemoveDevice
- AmResetPassword
- AmResetUserEnvSession
- AmResetUserPassword
- AmRestoreRecord
- AmReturnAsset
- AmReturnContract
- AmReturnPortfolioItem
- AmReturnTraining
- AmReturnWorkOrder
- AmRevCryptPassword
- AmRgbColor
- AmRollback
- AmSetFieldDateOnlyValue
- AmSetFieldDateValue
- AmSetFieldDoubleValue
- AmSetFieldLongValue
- AmSetFieldStrValue
- AmSetLinkFeatureValue
- AmSetProperty
- AmSetUserEnvSessionItem
- AmShowCableCrossConnect
- AmShowDeviceCrossConnect
- AmSqlTextConst
- AmStandIn
- AmStandInGroup
- AmStartTransaction
- AmStartup
- AmTableDesc
- AmTaxRate
- AmTransferSerialFilterToQueryTable
- AmTransferSerialPropsToScreen
- AmTransferViewFilterToQueryTable
- AmTransferViewPropsToScreen
- AmUpdateDetail
- AmUpdateLossLines
- AmUpdateRecord
- AmUpdateUser
- AmValueOf
- AmWizChain
- AmWorkTimeSpanBetween
- AppendOperand

- ApplyNewVals
- Asc
- Atn
- BasicToLocalDate
- BasicToLocalTime
- BasicToLocalTimeStamp
- Beep
- CDbl
- ChDir
- ChDrive
- Chr
- CInt
- CLng
- Cos
- CountOccurences
- CountValues
- CSng
- CStr
- CurDir
- CVar
- Date
- DateAdd
- DateAddLogical
- DateDiff
- DateSerial
- DateValue
- Day
- EnumToComboBox
- EscapeSeparators
- ExeDir
- Exp
- ExtractValue
- FileCopy
- FileDateTime
- FileExists
- FileLen
- Fix
- FormatDate
- FormatResString
- FV
- GetEnvVar
- GetListItem
- Hex
- Hour
- InStr
- Int
- IPMT
- IsNumeric
- Kill
- LCase
- Left
- LeftPart
- LeftPartFromRight
- Len
- LocalToBasicDate
- LocalToBasicTime
- LocalToBasicTimeStamp
- LocalToUTCDate
- Log
- LTrim
- MakeInvertBool
- Mid
- Minute
- MkDir
- Month
- Name
- Now
- NPER
- Oct
- ParseDate

- ParseDMYDate
- ParseMDYDate
- ParseYMDDate
- PMT
- PPMT
- PV
- Randomize
- RATE
- RemoveRows
- Replace
- Right
- RightPart
- RightPartFromLeft
- RmAllInDir
- RmDir
- Rnd
- RoundValue
- RTrim
- Second
- SetMaxInst
- SetSubList
- Sgn
- Shell
- Sin
- Space
- Sqr
- Str
- StrComp
- String
- SubList
- SysEnumToComboBox
- Tan
- Time
- Timer
- TimeSerial
- TimeValue
- ToSmart
- Trim
- UCase
- UnEscapeSeparators
- Union
- UTCToLocalDate
- Val
- WeekDay
- XmlAttribute
- Year

使用可能な関数 - データの変換 - 計算の実行

- AmBusinessSecondsInDay
- AmCalcConsolidatedFeature
- AmConvertDateBasicToUnix
- AmConvertDateIntlToUnix
- AmConvertDateStringToUnix
- AmConvertDateUnixToBasic
- AmConvertDateUnixToIntl
- AmConvertDateUnixToString
- AmConvertDoubleToString
- AmConvertMonetaryToString
- AmConvertStringToDouble
- AmConvertStringToMonetary
- AmCounter
- AmCryptPassword
- AmDateAdd
- AmDateAddLogical
- AmDateDiff
- AmDbGetDate
- AmDbGetDouble
- AmDbGetLimitedList
- AmDbGetList
- AmDbGetListEx
- AmDbGetLong
- AmDbGetPk
- AmDbGetString
- AmDbGetStringEx
- AmDeadLine
- AmEndOfNthBusinessDay
- AmEnumValList
- AmFormatLong
- AmGenSqlName
- AmGetComputeString
- AmListToString
- AmRevCryptPassword
- AmSqlTextConst
- AmTableDesc
- AmWorkTimeSpanBetween
- EnumToComboBox
- SysEnumToComboBox
- XmlAttribute

使用可能な関数 - 情報の取得

- AmBuildNumber
- AmConnectionName
- AmCurrentDate
- AmCurrentIsoLang
- AmCurrentLanguage
- AmCurrentServerDate
- AmGetCurrentNTDomain
- AmGetCurrentNTUser
- AmGetFeat
- AmGetFeatCount
- AmGetField
- AmGetFieldCount
- AmGetFieldDateOnlyValue
- AmGetFieldDateValue
- AmGetFieldDescription
- AmGetFieldDoubleValue
- AmGetFieldFormat
- AmGetFieldFormatFromName
- AmGetFieldFromName
- AmGetFieldLabel
- AmGetFieldLabelFromName
- AmGetFieldLongValue
- AmGetFieldName
- AmGetFieldRights
- AmGetFieldSize
- AmGetFieldSqlName
- AmGetFieldStrValue
- AmGetFieldType
- AmGetFieldUserType
- AmGetForeignKey
- AmGetIndex
- AmGetIndexCount
- AmGetIndexField
- AmGetIndexFieldCount
- AmGetIndexFlags
- AmGetIndexName
- AmGetLink
- AmGetLinkCardinality
- AmGetLinkCount
- AmGetLinkDstField

- AmGetLinkFeatureValue
- AmGetLinkFromName
- AmGetLinkType
- AmGetMainField
- AmGetMemoField
- AmGetNTDomains
- AmGetNTMachinesInDomain
- AmGetNTUsersInDomain
- AmGetPackageNames
- AmGetRecordFromMainId
- AmGetRecordHandle
- AmGetRecordId
- AmGetRelDstField
- AmGetRelSrcField
- AmGetRelTable
- AmGetReverseLink
- AmGetScreenSetsNames
- AmGetSelfFromMainId
- AmGetSerialModifiedPages
- AmGetSerialNbFilters
- AmGetSourceTable
- AmGetTable
- AmGetTableCount
- AmGetTableDescription
- AmGetTableFromName
- AmGetTableLabel
- AmGetTableName
- AmGetTableRights
- AmGetTableSqlName
- AmGetTargetTable
- AmGetTypedLinkField
- AmGetVersion
- AmGetViewModifiedPages
- AmGetViewNbFilters
- AmHasAdminPrivilege
- AmHasRelTable
- AmHasRightsForCreation
- AmHasRightsForDeletion
- AmHasRightsForFieldUpdate
- AmIsConnected
- AmIsExistingPage
- AmIsExistingScreen
- AmIsFieldForeignKey
- AmIsFieldIndexed
- AmIsFieldPrimaryKey
- AmIsFilterModifInSerial
- AmIsFilterModifInView
- AmIsLink
- AmIsModuleAuthorized
- AmIsTypedLink
- AmLastError
- AmLastErrorMsg
- AmLoginId
- AmLoginName
- AmNbLanguages
- AmOverflowTables
- AmPagePath
- AmProgress
- AmQueryNext
- AmQueryStartTable
- AmRgbColor
- AmValueOf

使用可能な関数 - Asset Managerでの内部動作のトリガ

- **AmCalculateAndStoreStatistic**
- **AmCalculateStatistic**
- **AmCalculateStatisticFromSQLName**
- **AmCleanup**
- **AmClearLastError**
- **AmCloseAllChildren**
- **AmCloseConnection**
- **AmDbExecAql**
- **AmDecrementLogLevel**
- **AmEvalScript**
- **AmExecTransition**
- **AmExecuteActionById**
- **AmExecuteActionByName**
- **AmExportDocument**
- **AmExportReport**
- **AmGeneratePlanningData**
- **AmIncrementLogLevel**
- **AmLog**
- **AmMsgBox**
- **AmOpenConnection**
- **AmOpenScreen**
- **AmQueryExec**
- **AmQueryGet**
- **AmQuerySetAddMainField**
- **AmQuerySetFullMemo**
- **AmQueryStop**
- **AmRefreshAllCaches**
- **AmRefreshProperty**
- **AmReleaseHandle**
- **AmStartup**
- **AmUpdateDetail**
- **AmWizChain**

使用可能な関数 - 「ファイナンス」モジュール

- **AmCalcDepr**
- **AmCbkReplayEvent**
- **AmConvertCurrency**
- **AmDefaultCurrency**
- **AmFormatCurrency**
- **AmTaxRate**

使用可能な関数 - データベースのデータの変更

- AmArchiveRecord
- AmBackupRecord
- AmCommit
- AmCreateLink
- AmCreateRecord
- AmDeleteLink
- AmDeleteRecord
- AmDuplicateRecord
- AmFlushTransaction
- AmImportDocument
- AmImportReport
- AmInsertRecord
- AmPurgeRecord
- AmRestoreRecord
- AmRollback
- AmSetFieldDateOnlyValue
- AmSetFieldDateValue
- AmSetFieldDoubleValue
- AmSetFieldLongValue
- AmSetFieldStrValue
- AmSetLinkFeatureValue
- AmSetProperty
- AmStartTransaction
- AmTransferSerialFilterToQueryTable
- AmTransferSerialPropsToScreen
- AmTransferViewFilterToQueryTable
- AmTransferViewPropsToScreen
- AmUpdateRecord
- AmUpdateUser

使用可能な関数 - 「調達」モジュール

- AmAddAllPOLinesToInv
- AmAddCatRefAndCompositionToPOrder
- AmAddCatRefToPOrder
- AmAddEstimLinesToPO
- AmAddEstimLineToPO
- AmAddLicContentToRequest
- AmAddPOLineToInv
- AmAddPOrderLineToReceipt
- AmAddReceiptLineToInvoice
- AmAddReqLinesToEstim
- AmAddReqLinesToPO
- AmAddReqLineToEstim
- AmAddReqLineToPO
- AmAddRequestLineToPOrder
- AmAddTemplateToPOrder
- AmAddTemplateToRequest
- AmCalculateCatRefQty
- AmCalculateReqLineQty
- AmCreateAssetsAwaitingDelivery
- AmCreateDelivFromPO
- AmCreateEstimFromReq
- AmCreateEstimsFromAllReqLines
- AmCreateInvFromPO
- AmCreateOrUpdateInvoiceFromReceipt
- AmCreatePOFromEstim
- AmCreatePOFromReq
- AmCreatePOrderFromRequest
- AmCreatePOrdersFromRequest
- AmCreatePOsFromAllReqLines
- AmCreateReceiptFromPOrder
- AmCreateRequestToInvoice
- AmCreateRequestToPOrder
- AmCreateRequestToReceipt
- AmCreateReturnFromReceipt
- AmGetCatRef
- AmGetCatRefFromCatProduct
- AmGetPOLinePrice
- AmGetPOLinePriceCur
- AmGetPOLineReference
- AmInstantiateReqLine

- **AmInstantiateRequest**
- **AmMapSubReqLineAgent**
- **AmReceiveAllPOLines**
- **AmReceivePOLine**
- **AmReturnAsset**
- **AmReturnContract**
- **AmReturnPortfolioItem**
- **AmReturnTraining**
- **AmReturnWorkOrder**

使用可能な関数 - 「契約」モジュール

- **AmUpdateLossLines**

使用可能な関数 - 「ケーブル」モジュール

- AmCheckTraceDone
- AmConnectTrace
- AmCreateAssetPort
- AmCreateCable
- AmCreateCableBundle
- AmCreateCableLink
- AmCreateDevice
- AmCreateDeviceLink
- AmCreateProjectCable
- AmCreateProjectDevice
- AmCreateProjectTrace
- AmCreateTraceHist
- AmCreateTraceLink
- AmDisconnectTrace
- AmFindCable
- AmFindDevice
- AmFindRootLink
- AmFindTermDevice
- AmFindTermField
- AmGetNextAssetPin
- AmGetNextAssetPort
- AmGetNextCableBundle
- AmGetNextCablePair
- AmGetTrace
- AmGetTraceFromHist
- AmMoveCable
- AmMoveDevice
- AmRefreshLabel
- AmRefreshTraceHist
- AmRemoveCable
- AmRemoveDevice
- AmShowCableCrossConnect
- AmShowDeviceCrossConnect

使用可能な関数 - 'ソフトウェア配布'モジュール

- **AmESDAddComputers**
- **AmESDCreateTask**

使用可能な関数 - 「ポートフォリオ」モジュール

- **AmStandIn**
- **AmStandInGroup**

使用可能な関数 - Asset Managerからの外部動作のトリガ

- **AmActionDde**
- **AmActionExec**
- **AmActionMail**
- **AmActionPrint**
- **AmActionPrintPreview**
- **AmActionPrintTo**