Peregrine

AssetCenter
Programmer's reference

Peregrine SYSTEMS

DAC-42-EN09

© Copyright 2002 Peregrine Systems, Inc. or its subsidiaries.

All Rights Reserved.

Information contained in this document is proprietary to Peregrine Systems, Incorporated, and may be used or disclosed only with written permission from Peregrine Systems, Inc. This manual, or any part thereof, may not be reproduced without the prior written permission of Peregrine Systems, Inc. This document refers to numerous products by their trade names. In most, if not all, cases these designations are claimed as Trademarks or Registered Trademarks by their respective companies.

Peregrine Systems ® and AssetCenter ® are trademarks of Peregrine Systems, Inc. or its subsidiaries.

This document and the related software described in this manual are supplied under license or nondisclosure agreement and may be used or copied only in accordance with the terms of the agreement. The information in this document is subject to change without notice and does not represent a commitment on the part of Peregrine Systems, Inc. Contact Peregrine Systems, Inc., Customer Support to verify the date of the latest version of this document.

The names of companies and individuals used in the sample database and in examples in the manuals are fictitious and are intended to illustrate the use of the software. Any resemblance to actual companies or individuals, whether past or present, is purely coincidental.

If you need technical support for this product, or would like to request documentation for a product for which you are licensed, contact Peregrine Systems, Inc. Customer Support by email at support@peregrine.com.

If you have comments or suggestions about this documentation, contact Peregrine Systems, Inc. Technical Publications by email at doc_comments@peregrine.com.

This edition applies to version 4.2.0 of the licensed program

AssetCenter

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

Programmer's reference

Table of Contents

I. Introduction 17

Chapter 1. Classification of functions 19

Families of functions 19
Scope of application of functions 20
Functional domains 20

Chapter 2. Conventions 23

Notation 23
Format of Date+Time constants in scripts 24
Format of Duration type constants in scripts 25

Chapter 3. Definitions 27

Definition of a function 27
Definition of the CurrentUser virtual link 28
Definition of a handle 28
Definition of an error code 29

Chapter 4. Function typing and parameters 31

List of types 31

Type of a function 32
Type of a parameter 32

II. Using the API 35

Chapter 5. Introduction 37

Warning 37
Installation 38

Chapter 6. Methodology 39

Chapter 7. Concepts and examples 41

Concepts 41
Handling dates 42
First example 42
Second example 43

III. Reference 45

Chapter 8. Reference 47

Abs() 47
AmActionDde() 48
AmActionExec() 50
AmActionMail() 51
AmActionPrint() 53
AmActionPrintPreview() 54
AmActionPrintTo() 55
AmAddAllPOLinesToInv() 56
AmAddCatRefAndCompositionToPOrder() 57
AmAddCatRefToPOrder() 59
AmAddEstimLinesToPO() 60
AmAddEstimLineToPO() 61
AmAddPOLineToInv() 63
AmAddPOrderLineToReceipt() 64
AmAddReceiptLineToInvoice() 65
AmAddReqLinesToEstim() 66
AmAddReqLinesToPO() 68
AmAddReqLineToEstim() 69
AmAddReqLineToPO() 70
AmAddRequestLineToPOrder() 71
AmAddTemplateToPOrder() 73
AmAddTemplateToRequest() 74

AmBusinessSecondsInDay() 75
AmCalcConsolidatedFeature() 76
AmCalcDepr() 77
AmCbkReplayEvent() 79
AmCheckTraceDone() 80
AmCleanup() 81
AmClearLastError() 82
AmCloseAllChildren() 83
AmCloseConnection() 83
AmCommit() 84
AmComputeAllLicAndInstallCounts() 85
AmComputeLicAndInstallCounts() 86
AmConnectTrace() 87
AmConvertCurrency() 89
AmConvertDateBasicToUnix() 91
AmConvertDateIntlToUnix() 92
AmConvertDateStringToUnix() 93
AmConvertDateUnixToBasic() 94
AmConvertDateUnixToIntl() 95
AmConvertDateUnixToString() 96
AmConvertDoubleToString() 97
AmConvertMonetaryToString() 98
AmConvertStringToDouble() 100
AmConvertStringToMonetary() 101
AmCounter() 102
AmCreateAssetPort() 103
AmCreateAssetsAwaitingDelivery() 105
AmCreateCable() 106
AmCreateCableBundle() 108
AmCreateCableLink() 109
AmCreateDelivFromPO() 111
AmCreateDevice() 112
AmCreateDeviceLink() 114
AmCreateEstimFromReq() 115
AmCreateEstimsFromAllReqLines() 117
AmCreateInvFromPO() 118
AmCreateLink() 119
AmCreatePOFromEstim() 120
AmCreatePOFromReq() 121
AmCreatePOrderFromRequest() 122
AmCreatePOrdersFromRequest() 124
AmCreatePOsFromAllReqLines() 125
AmCreateProjectCable() 126
AmCreateProjectDevice() 127

AmCreateProjectTrace() 128
AmCreateReceiptFromPOrder() 130
AmCreateRecord() 131
AmCreateRequestToInvoice() 132
AmCreateRequestToPOrder() 134
AmCreateRequestToReceipt() 135
AmCreateReturnFromReceipt() 137
AmCreateTraceHist() 138
AmCryptPassword() 140
AmCurrentDate() 141
AmCurrentIsoLang() 142
AmCurrentLanguage() 143
AmCurrentServerDate() 144
AmDateAdd() 145
AmDateAddLogical() 147
AmDateDiff() 148
AmDbGetDate() 150
AmDbGetDouble() 151
AmDbGetList() 152
AmDbGetListEx() 153
AmDbGetLong() 154
AmDbGetPk() 156
AmDbGetString() 157
AmDbGetStringEx() 159
AmDeadLine() 161
AmDecrementLogLevel() 162
AmDefAssignee() 163
AmDefaultCurrency() 164
AmDefEscalationScheme() 165
AmDefGroup() 167
AmDeleteLink() 169
AmDeleteRecord() 170
AmDisconnectTrace() 171
AmDuplicateRecord() 172
AmEndOfNthBusinessDay() 173
AmEnumValList() 175
AmEvalScript() 176
AmExecTransition() 177
AmExecuteActionById() 178
AmExecuteActionByName() 180
AmExportDocument() 181
AmFindCable() 182
AmFindDevice() 183
AmFindRootLink() 184

AmFindTermDevice() 185
AmFindTermField() 187
AmGenSqlName() 188
AmGetCatRef() 190
AmGetCatRefFromCatProduct() 191
AmGetComputeString() 192
AmGetCurrentNTDomain() 194
AmGetCurrentNTUser() 195
AmGetFeat() 196
AmGetFeatCount() 197
AmGetField() 198
AmGetFieldCount() 199
AmGetFieldDateValue() 200
AmGetFieldDescription() 201
AmGetFieldDoubleValue() 202
AmGetFieldFormat() 203
AmGetFieldFormatFromName() 205
AmGetFieldFromName() 206
AmGetFieldLabel() 207
AmGetFieldLabelFromName() 208
AmGetFieldLongValue() 209
AmGetFieldName() 211
AmGetFieldRights() 212
AmGetFieldSize() 213
AmGetFieldSqlName() 214
AmGetFieldStrValue() 216
AmGetFieldType() 217
AmGetFieldUserType() 219
AmGetForeignKey() 221
AmGetIndex() 222
AmGetIndexCount() 223
AmGetIndexField() 224
AmGetIndexFieldCount() 225
AmGetIndexFlags() 226
AmGetIndexName() 227
AmGetLink() 228
AmGetLinkCardinality() 229
AmGetLinkCount() 230
AmGetLinkDstField() 231
AmGetLinkFeatureValue() 232
AmGetLinkFromName() 234
AmGetLinkType() 235
AmGetMainField() 236
AmGetMemoField() 237

AmGetNextAssetPin() 237
AmGetNextAssetPort() 239
AmGetNextCableBundle() 241
AmGetNextCablePair() 242
AmGetNTDomains() 243
AmGetNTMachinesInDomain() 244
AmGetNTUsersInDomain() 245
AmGetPOLinePrice() 246
AmGetPOLinePriceCur() 247
AmGetPOLineReference() 249
AmGetRecordFromMainId() 250
AmGetRecordHandle() 251
AmGetRecordId() 252
AmGetRelDstField() 253
AmGetRelSrcField() 254
AmGetRelTable() 255
AmGetReverseLink() 256
AmGetSelfFromMainId() 256
AmGetSourceTable() 258
AmGetTable() 259
AmGetTableCount() 260
AmGetTableDescription() 261
AmGetTableFromName() 262
AmGetTableLabel() 263
AmGetTableName() 264
AmGetTableRights() 265
AmGetTableSqlName() 266
AmGetTargetTable() 267
AmGetTrace() 268
AmGetTraceFromHist() 270
AmGetTypedLinkField() 271
AmGetVersion() 272
AmHasAdminPrivilege() 273
AmHasRelTable() 274
AmImportDocument() 275
AmIncrementLogLevel() 277
AmInsertRecord() 278
AmInstantiateReqLine() 279
AmInstantiateRequest() 280
AmIsConnected() 281
AmIsFieldForeignKey() 282
AmIsFieldIndexed() 283
AmIsFieldPrimaryKey() 284
AmIsLink() 285

AmIsTypedLink() 286
AmLastError() 287
AmLastErrorMsg() 288
AmListToString() 289
AmLog() 290
AmLoginId() 291
AmLoginName() 292
AmMapSubReqLineAgent() 293
AmMoveCable() 294
AmMoveDevice() 296
AmMsgBox() 297
AmOpenConnection() 298
AmOpenScreen() 299
AmPagePath() 300
AmProgress() 301
AmQueryCreate() 302
AmQueryExec() 303
AmQueryGet() 304
AmQueryNext() 305
AmQuerySetAddMainField() 306
AmQuerySetFullMemo() 307
AmQueryStartTable() 308
AmQueryStop() 309
AmReceiveAllPOLines() 310
AmReceivePOLine() 312
AmRefreshAllCaches() 313
AmRefreshLabel() 314
AmRefreshProperty() 315
AmRefreshTraceHist() 316
AmReleaseHandle() 317
AmRemoveCable() 318
AmRemoveDevice() 319
AmReturnAsset() 321
AmReturnContract() 322
AmReturnPortfolioItem() 323
AmReturnTraining() 324
AmReturnWorkOrder() 325
AmRevCryptPassword() 327
AmRgbColor() 328
AmRollback() 329
AmSetFieldDateValue() 330
AmSetFieldDoubleValue() 331
AmSetFieldLongValue() 332
AmSetFieldStrValue() 334

AmSetLinkFeatureValue() 335
AmSetProperty() 336
AmShowCableCrossConnect() 337
AmShowDeviceCrossConnect() 338
AmSqlTextConst() 339
AmStartTransaction() 340
AmStartup() 341
AmTableDesc() 341
AmTaxRate() 343
AmUpdateDetail() 344
AmUpdateRecord() 345
AmValueOf() 346
AmWizChain() 347
AmWorkTimeSpanBetween() 348
AppendOperand() 350
ApplyNewVals() 351
Asc() 352
Atn() 353
BasicToLocalDate() 354
BasicToLocalTime() 355
BasicToLocalTimeStamp() 356
Beep() 357
CDbl() 358
ChDir() 359
ChDrive() 360
Chr() 361
CInt() 362
CLng() 363
Cos() 365
CountOccurences() 366
CountValues() 367
CSng() 368
CStr() 369
CurDir() 370
CVar() 371
DaContext() 372
DaCopy() 374
DaDbDeleteList() 376
DaDbGetList() 377
DaDbSetList() 379
DaDelete() 381
DaDownload() 382
DaDumpContext() 384
DaExec() 386

DaExecAction() 387
DaExecuteActionByName() 388
DaFileATime() 390
DaFileCRC() 391
DaFileCTime() 392
DaFileLanguage() 393
DaFileMTime() 394
DaFileSize() 395
DaFileType() 396
DaFileVersion() 397
DaFind() 398
DaFindNext() 399
DaFirstEnv() 400
DaGetEnv() 402
DaGetFileInfo() 404
DaImpersonate() 405
DaMkDir() 407
DaMove() 408
DaNetIpFromName() 410
DaNetNBTName() 411
DaNetPing() 412
DaNetWakeOnLan() 414
DaNetWinAddressByName() 415
DaNextEnv() 416
DaNTFileCopyTo() 417
DaNTFileCreateDir() 418
DaNTFileDelete() 419
DaNTFileDeleteDir() 420
DaNTFileDirCopyTo() 421
DaNTFileDirDownload() 422
DaNTFileDirUpload() 423
DaNTFileDownload() 424
DaNTFileUpload() 425
DaNTRegistryLMAddStringValue() 426
DaNTRegistryLMCreateKey() 427
DaNTRegistryLMDeleteKey() 427
DaNTRegistryLMDeleteValue() 428
DaNTRegistryLMGetLongValue() 429
DaNTRegistryLMGetStringValue() 430
DaNTRegistryLMSetLongValue() 431
DaNTRegistryLMSetStringValue() 432
DaNTServiceInstall() 433
DaNTServiceStart() 435
DaNTServiceStatus() 436

DaNTServiceStop() 437
DaNTServiceUninstall() 438
DaNTWMIExecMethod() 439
DaNTWMIExecQuery() 440
DaNTWMIGetCurrentArrayValue() 441
DaNTWMIGetCurrentPropertyValue() 443
DaNTWMIGetInstanceCount() 444
DaNTWMIGetPropertyValue() 445
DaNTWMIGetTotalPropertiesValue() 447
DaNTWMINextItem() 449
DaNTWMIResetEnumeration() 450
DaRegCreateKey() 450
DaRegDeleteKey() 451
DaRegExec() 452
DaRegGetValue() 453
DaRegOutputValue() 454
DaRegSetValue() 455
DaRegStrValue() 457
DaRegVarValue() 458
DaRename() 459
DaReturnValue() 460
DaRmDir() 461
DaSendMail() 462
DaSetContext() 464
DaSetOption() 466
DaSetReturnValue() 468
Date() 469
DateSerial() 470
DateValue() 471
DaTrackingDelete() 473
DaTrackingGet() 474
DaTrackingSet() 475
DaTrackingTest() 476
DaUpload() 477
DaWait() 479
Day() 481
EscapeSeparators() 482
ExeDir() 483
Exp() 484
ExtractValue() 485
FileCopy() 487
FileDateTime() 487
FileExists() 488
FileLen() 489

Fix() 490
FormatDate() 491
FormatResString() 493
FV() 494
GetListItem() 496
Hex() 497
Hour() 498
InStr() 499
Int() 500
IPMT() 501
IsNumeric() 503
Kill() 504
LCase() 505
Left() 506
LeftPart() 508
LeftPartFromRight() 509
Len() 511
LocalToBasicDate() 512
LocalToBasicTime() 513
LocalToBasicTimeStamp() 514
LocalToUTCDate() 515
Log() 516
LTrim() 517
MakeInvertBool() 518
Mid() 519
Minute() 521
MkDir() 522
Month() 523
Name() 524
Now() 525
NPER() 526
Oct() 527
ParseDate() 528
ParseDMYDate() 530
ParseMDYDate() 531
ParseYMDDate() 532
PMT() 533
PPMT() 534
PV() 536
Randomize() 538
RATE() 539
RemoveRows() 541
Replace() 542
Right() 544

RightPart() 545
RightPartFromLeft() 546
RmDir() 548
Rnd() 549
RTrim() 550
Second() 552
SetSubList() 553
Sgn() 554
Shell() 555
Sin() 556
Space() 558
Sqr() 559
Str() 560
StrComp() 561
String() 562
SubList() 563
Tan() 565
Time() 566
Timer() 567
TimeSerial() 568
TimeValue() 569
ToSmart() 571
Trim() 572
UCase() 573
UnEscapeSeparators() 574
Union() 575
UTCToLocalDate() 577
Val() 578
WeekDay() 579
Year() 580

IV. Index 583

Available functions - Domain: All 585

Available functions - Domain: Procurement 593

Available functions - Domain: Cable 595

Available functions - Domain: Actions 597

Available functions - Domain: Builtin 599

Available functions - Domain: Wizards 601

Available functions - Domain: Authentication 603

Available functions - Domain: Remote execution 605

Available functions - Domain: File management 607

Available functions - Domain: Context 609

Available functions - Domain: Registry 611

Available functions - Domain: Database 613

Available functions - Domain: Tracking 615

Available functions - Domain: WMI 617

Available functions - Domain: Network 619

PART I Introduction

1 Classification of functions

CHAPTER

Functions are classified according to three different levels. A given function can be classified by:

- Families of functions
- Scope of application of functions
- Functional domains

Families of functions

Functions in the AssetCenter environment can be organized into several main families:

- Functions recognized by AssetCenter These are essentially functions that can be used in the scriptable parts (in Basic) of the software.
- Functions recognized by the AssetCenter API: These functions can be called by external tools or be a program written in a high-level language.

These main families of functions are not mutually exclusive. For example, certain AssetCenter API functions can be used in the Basic scripts in the software. Such a function, originating from the AssetCenter API is said to be

"exposed" in AssetCenter's internal Basic scripts. The syntax of such a function may change but its behavior remains the same.

Scope of application of functions

The functions described in this document can be used in at least one of the following contexts:

- AssetCenter API libraries. In particular, the functions are available for development of Get-It applications.
- Field or link configuration script (**Configure the object** popup menu item or AssetCenter Database Administrator) and by extension **Calculation script** (SQL name: memScript) of a calculated field:
  - Default value,
  - Mandatory nature,
  - Historization,
  - Read-only nature,
  - ...
- Script type action:
  - Script defined in the **Script of the action** (SQL name: Script) of a Script action.
- AssetCenter wizards:
  - "FINISH.DO" script of a wizard.
  - Value definition scripts for the properties of nodes.

Functional domains

Each function is associated with one or more functional domains. A functional domain describes the nature of operations carried out by the function. The different functional domains are listed below:

- Built-in: Classic Basic functions, conversion and string handling functions, etc.
- Technical: Connection to a database, handling of table, field, link, index, record and query objects.
- Functional: Generic, line of business functions.

- Cable.
- Procurement.
- Chargeback.
- Wizards.
- Actions.
- Graphics.

2 Conventions

CHAPTER

This chapter describes:

- Notation
- Format of Date+Time constants in scripts
- Format of Duration type constants in scripts

Notation

The following notation is used in the examples in this manual:

[]	Square brackets denote an optional parameter. Do not type these brackets in your command. Exception: In Basic scripts, when the square brackets denote the path to data in the database, they must appear in the script as shown below: `[Link.Link.Field]`
<>	Angle brackets denote a parameter in plain language. Do not type these brackets. Substitute the text with the appropriate information.
{}	Curly brackets surround the definition of a node or a script block spanning several lines for a property.

|	A pipe is used to separate a series of possible parameters contained within curly brackets.

The following text styles have specific meanings:

`Fixed width characters`	DOS command, function parameter are data formatting.
`Example`	Example of code or command.
...	Code or command omitted.
Object name	The names of fields, tabs, menus and files are shown in bold.
Note:	Important note.

Note

Format of Date+Time constants in scripts

Dates referenced in scripts are expressed in international format, independently of the display options specified by the user:

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

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

Note:

The hyphen ("-") can also be used as a date separator.

About dates

Dates are expressed differently in internal Basic and from external tools:

- In Basic, a date can be expressed in international format, or as a floating point number ("Double" type). In this case, the integer part of the number represents the number of days elapsed since 1899-12-30 at midnight, the decimal part represents the fraction of the current date (The number of seconds elapsed since the start of the day divided by 86400).
- Externally, dates are expressed as a long integer ("Long" type) that represents the number of seconds elapsed since 01/01/1970 at midnight, independent of time zones (UTC time).

Format of Duration type constants in scripts

In scripts, durations are stored and expressed in seconds. E.g. to set the default value for a "Duration" type field to 3 days, use the following script:

```
RetVal=259200
```

Likewise, functions that calculate durations, such as the "AmWorkTimeSpanBetween()" function, return a number of seconds.

Note:

In financial calculations, AssetCenter takes into account the most common simplifications used. In this case alone, a year is considered as being 12 months and 1 month as 30 days (thus: 1 year = 360 days).

3 Definitions

CHAPTER

This chapter groups together the definitions of several essential terms.

You will find the following definitions:

- Definition of a function
- Definition of the CurrentUser virtual link
- Definition of a handle
- Definition of an error code

Definition of a function

A function is a program that performs operations and returns a value to the user. This value is called the "return value" or "return code".

Here is an example of the syntax used to call an internal AssetCenter function:

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

Here is the syntax of the same function via the AssetCenter API:

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

Definition of the CurrentUser virtual link

Definition

"CurrentUser" can be considered as a link starting in all tables and pointing to the record in the table of departments and employees corresponding to the current user.

- In the "CurrentUser" format, it points to the record corresponding to the current user, and returns the user's ID number.
- In the "CurrentUser.Field" format, it returns the value of the field for the current user.

Note:

This virtual link is not displayed in the list of fields and links; therefore it is not directly accessible in AssetCenter's internal script builder. You must enter this expression manually.

Equivalencies

The "AmLoginName()" and "AmLoginId()" functions, which return the current user's "Name" (SQL name: Name) and ID (SQL name: lPersId), respectively, may be considered as functions derived from "CurrentUser". In effect, the following are equivalent:

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

Definition of a handle

A handle represents a unique identifier for an object. In the context of AssetCenter, this object can be a field, link, index, query, record, table or a connection. Handles are 32-bit integers ("Long" type).

Note:

The NULL value is not a valid handle.

From external tools, you can also access (database) connection handles.

Definition of an error code

When a function fails, it returns an error code.

From external tools

This error code and associated message can be recovered by external tools via the "AmLastError()" and "AmLastErrorMsg()" functions respectively. It can be cleared using the "AmClearLastError()" function.

Note:

Any new function call clears the error code and previous message.

Internally

Internally (in Basic scripts, for example), the last error code and its description can be recovered using the **Err.Number** and **Err.Description** functions.

Note:

Internally, you don't need to program your own error handling. A script with problems will stop and a database rollback will be performed if necessary.

You can raise an error on purpose using the Err.Raise function. Its syntax is as follows:

```
Err.Raise (<Error code>, <Error message>)
```

Note:

When the creation or modification of a record is invalidated by the value of the "Validity" field for the table in question, it is a good idea to raise an error message using the **Err.Raise** function in order to warn the user (code 12006 or 12007). If you do not do this, the user will not necessarily understand why the record cannot be modified or created.

The following table lists the most frequent error codes:

Error code	Meaning
12001	Undefined error
12002	Bad parameter for a function
12003	Invalid handle or object deleted
12004	No more data available. This error typically occurs when executing queries. When the query does not return data, this error is raised.
12005	Internal database server error
12006	Invalid value (incorrect type for a parameter, etc.)
12007	Non valid record (a mandatory field is not populated, for example)
12008	Problems with database access rights
12009	Obsolete or non implemented function
12010	Maximum number of database connections exceeded

4 Function typing and parameters

CHAPTER

This chapter contains information on the following:

- List of types
- Type of a function
- Type of a parameter

List of types

The following table summarizes the various types available for a function or a parameter:

Type	Description
Integer	Integer from -32,768 to +32,767.
Long	Integer from -2,147,483,647 to +2,147,483,646.
Double	8 byte floating-point number.
String	Text in which all characters are allowed.
Date	Date or Date+Time.
Variant	Generic type that can represent any type.

Note:

Not all of these types are available from external tools. Only Long, Double and String types are available. Variant is not used and Integer and Date objects are represented by a Long.

Type of a function

The type of a function corresponds to the type of the value returned by the function. We recommend paying close attention to this piece of information since it can be the cause of compilation and runtime errors in your programs.

For example, you cannot use a function returning a certain typed value in the definition of the default value of a differently typed field. Try, for example, assigning this default value script to any "Date" or "Date+Time" type field:

```
RetVal=AmLoginName()
```

The "AmLoginName()" function returns the name of the connected user in the form of a character string ("String" type). The type of this return value is therefore incompatible with "Date" type fields and AssetCenter displays an error message.

Type of a parameter

The parameters that can be used in functions also have a type that must be respected in order for the proper execution of the function. In the syntax of functions, parameters are prefixed according to their type. To avoid any possible confusion, the prefixes used in this reference differ according to the syntax (API or Basic) of the function. The following table resumes the equivalencies between the prefixes used in the API syntax and the Basic syntax:

Type	Prefix used in the API syntax	Prefix used in the Basic syntax
Integer	"i"	"i"
Long	"h" for a handle or "l" for a number	"l"
Double	"d"	"d"

Type	Prefix used in the API syntax	Prefix used in the Basic syntax
String	"char*psz"	"str"
Date	"ltm"	"dt"
Variant	"v"	"v"

II Using the API

PART

5 Introduction

CHAPTER

The AssetCenter API is provided as a 32-bit DLL, useable in Windows 95/98, Windows NT or Windows 2000.

It has been tested successfully in the following environments:

- Visual Basic 4.0, 5.0, and 6.0,
- Visual C++ 4.0, 5.0, and 6.0,
- All Microsoft products using VBA (Visual Basic for Applications)

Note:

The API should be compatible with all tools authorizing the use of third-party DLLs.

Warning

Before using the AssetCenter API, the user should be familiar with the terminology used in the AssetCenter conceptual model. In particular, a minimal knowledge of the database structure is required.

Information on the structure of the database can be found in the manual entitled "Reference guide: Administration and advanced use", chapter "Structure of the database" and in the "Database.txt" and "Tables.txt" files, which can be found in the "doc\infos" sub-folder of the AssetCenter installation folder.

Installation

Before using the AssetCenter API, it is highly recommended to install a fully functional version of AssetCenter. In this way, you can quickly test whether databases can be correctly accessed from a given computer and create or configure database connections. The API uses the same database layers and the same configuration information as AssetCenter to access data sources, so problems can often be investigated from within AssetCenter.

The typical steps for setting up a development environment with AssetCenter are as follows:

- Install a 32-bit version of AssetCenter with the AssetCenter API package.
- Use AssetCenter to configure the data source, and try to open a database.
- Use your development environment to call AssetCenter API functions.

To familiarize yourself with the AssetCenter API, we recommend using a demonstration database or any non critical source of data for which manipulation errors are not critical.

6 Methodology

CHAPTER

A typical sequence of operations using the AssetCenter API would be:

1. Create a query using an AQL statement:
   `SELECT AssetTag, User.Name, Supervisor.Name FROM amPortfolio`

   Note:

   You can also use AssetCenter Export to generate an AQL query.

2. Browse the query result set and retrieve any useful handles on specific items.
3. Use the retrieved handles to update the information in the corresponding objects.
4. Commit (accept) or rollback (cancel) the whole transaction.

7 Concepts and examples

CHAPTER

This section contains information on the following:

- Concepts
- Handling dates
- First example
- Second example

Concepts

AssetCenter is built around an object oriented design and the API maintains this structural view. To accommodate the limitation of Windows DLLs which imposes the use of a flat "C like API", the AssetCenter API work around this problem by using handles (32-bit integers) to identify every user-created object. This approach has the advantage of allowing non-object oriented languages to access the AssetCenter object model.

Before doing anything else, your program must call "AmStartUp()" in order to initialize the AssetCenter libraries. Your program must also terminate by calling the "AmCleanUp()" function.

Before accessing a database object, a connection should be established between the user and the database. This connection is identified by a "handle" on a "connection" object (this handle is then used in all the API functions that interact with the database. It corresponds to the parameter "hApiCnxBase". This object can then be used to create queries and gain access to records.

Note:

All database objects are linked to a connection so information about user privileges can be checked.

The first step is to open a connection using a valid data source name and a valid login/password combination.

Warning:

When you connect to the AssetCenter database via the AssetCenter API, a connection slot is used.

Handling dates

When reading dates, you have the choice of the following two functions for "Date" and "Date+Time" type fields:

- "AmGetFieldLongValue()" which returns the date as a Unix "Long" (UTC). We recommend using this function for calculations involving dates.
- "AmGetFieldStrValue()" which returns the date as a string in the same format as the Windows Control Panel. This date takes time zones into account. We recommend using this function when you need to display a date.

First example

The following example, written in C, declares a connection to the demonstration database:

```
long lCnx ;
lCnx = AmOpenConnection(ACDemo351ENG, Admin , ) ;
```

"lCnx" is a connection handle that can be used to identify the newly created connection.

This connection can now be used to create queries and access the database. The following example, written in C, defines a query on the table of assets and browses the results set:

```
#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("ACDemo300Eng","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'")
    /* Browses 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();
```

Second example

Queries are used to locate objects in the database. When you want to update a record, a handle on a "record" object must be obtained using a query. The record can then be processed using other AssetCenter API functions.

The next example shows how to modify a field in a specific record:

```
/* Handles for objects */
long lCnx ;
long lQuery ;
long lStatus ;
long lRecord ;
AmStartup();
lCnx = AmOpenConnection("ACDemo300Eng","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 = 'Compaq
' 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();
```

This example shows how to get a unique record handle using the query mechanism. In this sample code, the query is used to locate a single item, but it is also possible to use "AmQueryExec()" to get a set of records and then get a record handle for one or more records.

Note:

For reasons of simplicity, this example does not deal with all possible error codes.

III Reference

PART

8 Reference

CHAPTER

Abs()

Returns the absolute value of a number.

Internal BASIC syntax

Function Abs(dValue As Double) As Double

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓

	Available
FINISH.DO script of a wizard	✓

Input parameters

- **dValue**: Number for which you want to know the absolute value.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

AmActionDde()

This function sends a DDE request to a DDE server application. Using this function, AssetCenter can control another application using a DDE link. This function is equivalent to a DDE type action

API syntax

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

Internal BASIC syntax

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

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strService**: This parameter contains the name of the DDE service provided by the executable you want to call. Refer to the documentation of the executable to obtain the list of DDE services it provides.
- **strTopic**: This parameter contains the topic (i.e. the context) in which a DDE action must be executed.
- **strCommand**: This parameter contains the commands the external application must execute. You must follow the syntax defined by the external application.
- **strFileName**: If the service is not resident in memory, you must load it by specifying in this parameter the name of the executable (or the name of any file associated with an executable using the Windows File Manager) which activates the service.
- **strDirectory**: This parameter contains the path for the file defined in **strFileName**.
- **strParameters**: This parameter contains the various parameters to pass to the executable which activates the service when it is launched.

- **strTable**: Optional parameter used when the action is contextual. It indicates the SQL name of the table containing the record to which the action applies.
- **lRecordId**: Optional parameter used when the action is contextual. It indicates the identifier of the record to which the action applies.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmActionExec()

This function launches an ".exe", ".com", ".bat", ".pif" application. You can also refer to any type of document, as long as the document extension is associated with an executable via the Windows File Manager. This function is equivalent to an action of "Executable" type.

API syntax

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

Internal BASIC syntax

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

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	

	Available
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strFileName**: This parameter contains the name of the executable, or of any document (associated with an executable via the File Manager).
- **strDirectory**: This parameter contains the path for the file specified in the **strFileName** parameter.
- **strParameters**: This optional parameter contains the various parameters to be provided to the executable when it is launched.
- **strTable**: Optional parameter used when the action is contextual. It indicates the SQL name of the table containing the record to which the action applies.
- **lRecordId**: Optional parameter used when the action is contextual. It indicates the identifier of the record to which the action applies.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Example

This example executes the Windows NT explorer (situated in the "WinNT" folder of the "C" drive):

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

AmActionMail()

This function sends a message via one of the messaging systems managed by AssetCenter:

- Internal messaging system.

- External messaging system based on the VIM standard (Lotus Notes, etc.)
- External messaging system based on the MAPI standard (Microsoft Exchange, Microsoft Outlook, etc.)
- External messaging system based on the SMTP standard (Internet standard)

API syntax

long AmActionMail(char *strTo, char *strCc, char *strCcc, char *strSubject, char *strMessage, long iPriority, long bAcknowledge, char *strRefObject, char *strTable, long lRecordId);

Internal BASIC syntax

Function AmActionMail(strTo As String, strCc As String, strCcc As String, strSubject As String, strMessage As String, iPriority As Long, bAcknowledge As Long, strRefObject As String, strTable As String, lRecordId As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strTo**: This parameter contains the list of addresses for the message recipients in the form messaging_system:address. The semi-colon is used as a separator.
- **strCc**: This parameter contains the list of addresses for people who are copied in the message. The semi-colon is used as a separator.

- **strCcc**: This parameter contains the list of addresses for people who receive a blind copy of the message (they do not appear in the list of recipients). The semi-colon is used as a separator.
- **strSubject**: This parameter contains the message subject.
- **strMessage**: This parameter contains the message body.
- **iPriority**: This parameter defines the priority for sending the message:
  - 0: Low priority
  - 1: Normal priority.
  - 2: High priority.
- **bAcknowledge**: This parameter indicates whether the message sender will receive an acknowledgement:
  - 0: the sender does not receive an acknowledgement.
  - 1: the sender does receive an acknowledgement.
- **strRefObject**: This parameter is only used for messages sent via the AssetCenter internal messaging system. It contains the SQL name of the link to follow from the record corresponding to the context of execution to reach the referenced object. The CurrentUser virtual link can be used.
- **strTable**: Optional parameter used when the action is contextual. It indicates the SQL name of the table containing the record to which the action applies.
- **lRecordId**: Optional parameter used when the action is contextual. It indicates the identifier of the record to which the action applies.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmActionPrint()

This function prints a report on a given record in the database.

Internal BASIC syntax

Function AmActionPrint(lReportId As Long, lRecordId As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lReportId**: This parameter contains the identifier of the report to be printed.
- **lRecordId**: This parameter contains the identifier of the record concerned by the report. By default, this parameter is set to "0". The table concerned is implicitly defined by the report.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmActionPrintPreview()

This function triggers a print preview of a report concerning a given database record.

Internal BASIC syntax

Function AmActionPrintPreview(lReportId As Long, lRecordId As Long) As Long

Field of application

Version: 3.60

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lReportId**: This parameter contains the identifier of the report concerned.
- **lRecordId**: This parameter contains the identifier of the record concerned by the report. By default, this parameter is "0".

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmActionPrintTo()

This function prints a report on a given database record and on a given printer.

Internal BASIC syntax

Function AmActionPrintTo(strPrinterName As String, lReportId As Long, lRecordId As Long) As Long

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strPrinterName**: This parameter contains the name of the printer to use.
- **lReportId**: This parameter contains the identifier of the report to print.
- **lRecordId**: This parameter contains the identifier of the record concerned by the report. By default, this parameter is set to "0".

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmAddAllPOLinesToInv()

This function adds a purchase order in full to an existing supplier invoice.

API syntax

long AmAddAllPOLinesToInv(long hApiCnxBase, long lPOrdId, long lInvId);

Internal BASIC syntax

Function AmAddAllPOLinesToInv(lPOrdId As Long, lInvId As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lPOrdId**: This parameter contains the identifier of the order to be added to the supplier invoice.
- **lInvId**: This parameter contains the identifier of the supplier invoice to which the purchase order is added.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmAddCatRefAndCompositionToPOrder()

This function enables you to add the full contents of a catalog reference to a given purchase order.

API syntax

long AmAddCatRefAndCompositionToPOrder(long hApiCnxBase, long lPOrderId, long lCatRefId, float fCatRefQty, long lRequestId, double dUnitPrice, char *strCur);

Internal BASIC syntax

Function AmAddCatRefAndCompositionToPOrder(lPOrderId As Long, lCatRefId As Long, fCatRefQty As Single, lRequestId As Long, dUnitPrice As Double, strCur As String) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lPOrderId**: This parameter contains the identifier of the purchase order to complete.
- **lCatRefId**: This parameter contains the identifier of the catalog reference.
- **fCatRefQty**: This parameter contains the quantity (in the unit associated with the product) to add.
- **lRequestId**: This parameter contains the identifier of the request that this purchase order will satisfy.
- **dUnitPrice**: This parameter contains the unit price of the product of the catalog reference.
- **strCur**: This parameter contains the code of the currency in which the unit price is expressed

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Notes

Note:

In particular, this function can enable you to use the composition of the products of a catalog reference to fill out a purchase order.

AmAddCatRefToPOrder()

This function enables you to add a catalog reference to an existing purchase order.

API syntax

long AmAddCatRefToPOrder(long hApiCnxBase, long lRequestLineId, long lCatRefId, long lPOrderId, float fQty, long bCanMerge);

Internal BASIC syntax

Function AmAddCatRefToPOrder(lRequestLineId As Long, lCatRefId As Long, lPOrderId As Long, fQty As Single, bCanMerge As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lRequestLineId**: This parameter contains the identifier of the request line associated with the purchase order.
- **lCatRefId**: This parameter contains the identifier of the catalog reference to be added.
- **lPOrderId**: This parameter contains the identifier of the purchase order concerned by the operation.
- **fQty**: This parameter contains the quantity (in the unit associated with the product) to add.
- **bCanMerge**: This parameter enables you to specify whether the added line can be merged with an existing line in the purchase line.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmAddEstimLinesToPO()

This function adds all estimate lines of an estimates to an existing order.

API syntax

long AmAddEstimLinesToPO(long hApiCnxBase, long lEstimId, long lPOrdId, long bMergeLines);

Internal BASIC syntax

Function AmAddEstimLinesToPO(lEstimId As Long, lPOrdId As Long, bMergeLines As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lEstimId**: This parameter contains the identifier of the estimate to be added to the purchase order.
- **lPOrdId**: This parameter contains the identifier of the purchase order to which the all the estimate lines of the estimate are added.
- **bMergeLines**: This parameter enables you to specify if identical request lines are to be combined (**bMergeLines**=1) to give one single line. The quantities given for the lines to be combined are added together and a single line is created.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmAddEstimLineToPO()

This function adds an estimate line to an existing purchase order.

API syntax

long AmAddEstimLineToPO(long hApiCnxBase, long lEstimLineId, long lPOrdId, long bMergeLines);

Internal BASIC syntax

Function AmAddEstimLineToPO(lEstimLineId As Long, lPOrdId As Long, bMergeLines As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lEstimLineId**: This parameter contains the identifier of the estimate line to be added to the purchase order.
- **lPOrdId**: This parameter contains the identifier of the purchase order to which the estimate line is added.
- **bMergeLines**: This parameter enables you to specify if identical request lines are to be combined (**bMergeLines**=1) to given one single line. The quantities given for the lines to be combined are added together and a single line is created.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmAddPOLineToInv()

This function adds a given quantity of item(s) on an order line to a supplier invoice.

API syntax

long AmAddPOLineToInv(long hApiCnxBase, long lPOrdLineId, long lInvId, float fQty);

Internal BASIC syntax

Function AmAddPOLineToInv(lPOrdLineId As Long, lInvId As Long, fQty As Single) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lPOrdLineId**: This parameter contains the identifier of the order line to be added to the supplier invoice.
- **lInvId**: This parameter contains the identifier of the supplier invoice to which the items of the order line are added.
- **fQty**: This parameter contains the quantity of items on the order line to be added to the supplier invoice.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmAddPOrderLineToReceipt()

This function enables you to add an order line to a receipt. In this way you can receive an order line within the existing receipt.

API syntax

long AmAddPOrderLineToReceipt(long hApiCnxBase, long lPOrderLineId, long lRecptId, float fQty, long bCanMerge);

Internal BASIC syntax

Function AmAddPOrderLineToReceipt(lPOrderLineId As Long, lRecptId As Long, fQty As Single, bCanMerge As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lPOrderLineId**: This parameter contains the identifier of the order line.
- **lRecptId**: This parameter contains the identifier of the impacted receipt.
- **fQty**: This parameter contains the quantity to receive. In this way, you can limit the quantity received in relation to the la quantity ordered (in the unit of the product).
- **bCanMerge**: This parameter enables you to specify whether the line can be merged with an existing line in the receipt.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmAddReceiptLineToInvoice()

This function enables you to add a receipt line to an invoice. By doing so, you can invoice a receipt line within an existing invoice.

API syntax

long AmAddReceiptLineToInvoice(long hApiCnxBase, long lRecptLineId, long lInvoiceId, float fQty, long bCanMerge);

Internal BASIC syntax

Function AmAddReceiptLineToInvoice(lRecptLineId As Long, lInvoiceId As Long, fQty As Single, bCanMerge As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lRecptLineId**: This parameter contains the identifier of the receipt line.
- **lInvoiceId**: This parameter contains the identifier of the impacted invoice.
- **fQty**: This parameter contains the quantity to invoice. In this way, you can limit the quantity invoiced in relation to the la quantity received (in the unit of the product).
- **bCanMerge**: This parameter enables you to specify whether the line can be merged with an existing line in the invoice.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmAddReqLinesToEstim()

This function adds all request lines of a request to an existing estimate.

API syntax

long AmAddReqLinesToEstim(long hApiCnxBase, long lReqId, long lEstimId, long bMergeLines);

Internal BASIC syntax

Function AmAddReqLinesToEstim(lReqId As Long, lEstimId As Long, bMergeLines As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lReqId**: This parameter contains the identifier of the request to be added to the estimate.
- **lEstimId**: This parameter contains the identifier of the estimate to which all the request lines of the request are added.
- **bMergeLines**: This parameter enables you to specify if identical request lines are to be combined (**bMergeLines**=1) to given one single line. The quantities given for the lines to be combined are added together and a single line is created.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmAddReqLinesToPO()

This function adds all the request lines of a request to an existing purchase order. The supplier specified in the request must be identical to that in the purchase order concerned.

API syntax

long AmAddReqLinesToPO(long hApiCnxBase, long lReqId, long lPOrdId, long bMergeLines);

Internal BASIC syntax

Function AmAddReqLinesToPO(lReqId As Long, lPOrdId As Long, bMergeLines As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lReqId**: This parameter contains the identifier of the request to be added to the purchase order.
- **lPOrdId**: This parameter contains the identifier of the purchase order to which the request lines are to be added.
- **bMergeLines**: This parameter enables you to specify if identical request lines are to be combined (**bMergeLines**=1) to given one single line. The

quantities given for the lines to be combined are added together and a single line is created.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmAddReqLineToEstim()

This function adds a request line to an existing estimate.

API syntax

long AmAddReqLineToEstim(long hApiCnxBase, long lReqLineId, long lEstimId, long bMergeLines);

Internal BASIC syntax

Function AmAddReqLineToEstim(lReqLineId As Long, lEstimId As Long, bMergeLines As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lReqLineId**: This parameter contains the identifier of the request line to be added to the estimate.
- **lEstimId**: This parameter contains the identifier of the estimate to which the request line is added.
- **bMergeLines**: This parameter enables you to specify if identical request lines are to be combined (**bMergeLines**=1) to given one single line. The quantities given for the lines to be combined are added together and a single line is created.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmAddReqLineToPO()

This function adds a request line to an existing purchase order.

API syntax

long AmAddReqLineToPO(long hApiCnxBase, long lReqLineId, long lPOrdId, long bMergeLines);

Internal BASIC syntax

Function AmAddReqLineToPO(lReqLineId As Long, lPOrdId As Long, bMergeLines As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lReqLineId**: This parameter contains the identifier of the request line to be added to the purchase order.
- **lPOrdId**: This parameter contains the identifier of the purchase order to which the request line is to be added.
- **bMergeLines**: This parameter enables you to specify if identical request lines are to be combined (**bMergeLines**=1) to given one single line. The quantities given for the lines to be combined are added together and a single line is created.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmAddRequestLineToPOrder()

This function enables you to add a request line to a purchase order.

API syntax

long AmAddRequestLineToPOrder(long hApiCnxBase, long lRequestLineId, long lPOrderId, float fQty, long bCanMerge);

Internal BASIC syntax

Function AmAddRequestLineToPOrder(lRequestLineId As Long, lPOrderId As Long, fQty As Single, bCanMerge As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lRequestLineId**: This parameter contains the identifier of the request line.
- **lPOrderId**: This parameter contains the identifier of the impacted purchase order.
- **fQty**: This parameter contains the quantity to order. In this way, you can limit the quantity orderd in relation to the la quantity requested (in the unit of the product).
- **bCanMerge**: This parameter enables you to specify whether the line can be merged with an existing line in the purchase order.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmAddTemplateToPOrder()

This function enables you to add the full contents of a standard purchase order to a given purchase order.

API syntax

long AmAddTemplateToPOrder(long hApiCnxBase, long lRequestId, long lPOrderId, long lTemplateId, long lQty, long bCanMerge);

Internal BASIC syntax

Function AmAddTemplateToPOrder(lRequestId As Long, lPOrderId As Long, lTemplateId As Long, lQty As Long, bCanMerge As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lRequestId**: This parameter contains the identifier of the request line to satisfy for the order lines that will be added.
- **lPOrderId**: This parameter contains the identifier of the impacted purchase order.
- **lTemplateId**: This parameter contains the identifier of the standard purchase order to add.

- **lQty**: This parameter contains the quantity (in the unit of the product) to add.
- **bCanMerge**: This parameter enables you to specify whether the line can be merged with an existing line in the purchase order.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmAddTemplateToRequest()

This function enables you to add the full contents of a standard request to a given request.

API syntax

long AmAddTemplateToRequest(long hApiCnxBase, long lReqId, long lTemplateId, long lQty, long bCanMerge);

Internal BASIC syntax

Function AmAddTemplateToRequest(lReqId As Long, lTemplateId As Long, lQty As Long, bCanMerge As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lReqId**: This parameter contains the identifier of the affected request line.
- **lTemplateId**: This parameter contains the identifier of the standard request to add.
- **lQty**: This parameter contains the quantity (in the unit of the product) to add.
- **bCanMerge**: This parameter enables you to specify whether the line can be merged with an existing line in the request.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmBusinessSecondsInDay()

Calculates the number of business seconds in a day according to a calendar.

API syntax

long AmBusinessSecondsInDay(char *strCalendarSqlName, long tmDate);

Internal BASIC syntax

Function AmBusinessSecondsInDay(strCalendarSqlName As String, tmDate As Date) As Date

Field of application

Version: 3.00

	Available
AssetCenter API	✓

	Available
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strCalendarSqlName**: SQL name of the calendar used for the calculation.
- **tmDate**: Date for which the calculation is performed.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCalcConsolidatedFeature()

Calculates the value of consolidated feature on a table identified by its SQL name.

API syntax

long AmCalcConsolidatedFeature(long hApiCnxBase, long lCalcFeatId, char *strSQLTableName);

Internal BASIC syntax

Function AmCalcConsolidatedFeature(lCalcFeatId As Long, strSQLTableName As String) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lCalcFeatId**: Identifier of the consolidated feature.
- **strSQLTableName**: SQL name of the table for which the consolidated feature is calculated. The feature must be defined for this table.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmCalcDepr()

This function enables you calculate the depreciation amount for an asset on a given date. It returns the depreciation value on this date.

API syntax

double AmCalcDepr(long iType, long lDuration, double dCoeff, double dPrice, long tmStart, long tmDate);

Internal BASIC syntax

Function AmCalcDepr(iType As Long, lDuration As Long, dCoeff As Double, dPrice As Double, tmStart As Date, tmDate As Date) As Double

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **iType**: This parameter identifies the nature of the depreciation. The following values are possible:
  - 0: No depreciation
  - 1: Straight line
  - 2: Declining balance
- **lDuration**: This parameter contains the period over which the asset is depreciated. This period is expressed in seconds.
- **dCoeff**: This parameter contains the coefficient applied in the declining balance method. It is not interpreted in the case of the straight line depreciation method but must have a value.
- **dPrice**: This parameter contains the initial value of the asset concerned by the depreciation calculation.
- **tmStart**: This parameter contains the date from which the asset is depreciated.
- **tmDate**: This parameter contains the date on which the depreciation and residual value of the asset are calculated.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCbkReplayEvent()

This function enables you to replay the chargeback rule at the origin of an event, after correcting the record at the origin of the event.

API syntax

long AmCbkReplayEvent(long hApiCnxBase, long lCbkEventId);

Internal BASIC syntax

Function AmCbkReplayEvent(lCbkEventId As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lCbkEventId**: This parameter contains the identifier of the chargeback event concerned.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmCheckTraceDone()

The AmCheckTraceDone API determines if a port (lPortId) or bundle (lBundleId) is connected to an existing trace. The trace direction (iTraceDir) identifies if the trace should be checked in the direction of user-to-host (iTraceDir = 1) or host-to-user (iTraceDir = 0).

API syntax

long AmCheckTraceDone(long hApiCnxBase, long lPortId, long lBundleId, long iTraceDir);

Internal BASIC syntax

Function AmCheckTraceDone(lPortId As Long, lBundleId As Long, iTraceDir As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓

	Available
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lPortId**: This parameter is the port ID to check.
- **lBundleId**: This parameter is the bundle ID to check.
- **iTraceDir**: This parameter defines the direction to check.
  - 1: Check in the host direction
  - 0: Check in the host direction

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCleanup()

This function must be called at the end of scripts using the database modification functions. It frees all used resources.

API syntax

void AmCleanup();

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	

AmClearLastError()

This function clears the information concerning the last error message occurred during the last function call.

API syntax

long AmClearLastError(long hApiCnxBase);

Internal BASIC syntax

Function AmClearLastError() As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmCloseAllChildren()

This function destroys all the objects created during the current connection.

API syntax

long AmCloseAllChildren(long hApiCnxBase);

Internal BASIC syntax

Function AmCloseAllChildren() As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmCloseConnection()

Ends the AssetCenter session for a given connection. All objects (queries, records, tables, fields, etc.) created within this connection are automatically destroyed. Their handles become invalid. The connection handle no longer exists.

API syntax

long AmCloseConnection(long hApiCnxBase);

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmCommit()

This function commits all the modifications made to the database associated with the connection.

API syntax

long AmCommit(long hApiCnxBase);

Internal BASIC syntax

Function AmCommit() As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmComputeAllLicAndInstallCounts()

This function performs the count of software licenses and installations for all records.

API syntax

long AmComputeAllLicAndInstallCounts(long hApiCnxBase);

Internal BASIC syntax

Function AmComputeAllLicAndInstallCounts() As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmComputeLicAndInstallCounts()

This function performs the count of software licenses and installations for a record.

API syntax

long AmComputeLicAndInstallCounts(long hApiCnxBase, long lSLCountId);

Internal BASIC syntax

Function AmComputeLicAndInstallCounts(lSLCountId As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lSLCountId**: This parameter contains the identifier of the software license counter.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmConnectTrace()

The AmConnectTrace API is for connecting a source device/cable to a destination device/cable and creating a trace history and a trace operation.

API syntax

long AmConnectTrace(long hApiCnxBase, long iSrcLinkType, long lSrcPortBunId, long lSrcLabelRuleId, long iDestLinkType, long lDestPortBunId, long lDestLabelRuleId, long iTraceDir, long lDutyId, char *strComment, long lCabTraceOutId);

Internal BASIC syntax

Function AmConnectTrace(iSrcLinkType As Long, lSrcPortBunId As Long, lSrcLabelRuleId As Long, iDestLinkType As Long, lDestPortBunId As Long, lDestLabelRuleId As Long, iTraceDir As Long, lDutyId As Long, strComment As String, lCabTraceOutId As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓

	Available
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **iSrcLinkType**: This parameter determines the trace type for the source device/cable.
  - 8: Cable
  - 9: Device
- **lSrcPortBunId**: This parameter is the port or bundle to be connected on the source side.
- **lSrcLabelRuleId**: This parameter is the label rule used for the source link.
- **iDestLinkType**: This parameter determines the trace type for the destination device/cable.
  - 8: Cable
  - 9 Device
- **lDestPortBunId**: This parameter is the port or bundle to be connected on the destination side.
- **lDestLabelRuleId**: This parameter is the label rule used for the destination link.
- **iTraceDir**: This parameter defines the direction of the connection.
  - 1: user to host
  - 0: host to user
- **lDutyId**: This parameter is the duty for a cable type link.
- **strComment**: This parameter is the label for the trace operation.
- **lCabTraceOutId**: This parameter is the Cable Trace Output ID.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmConvertCurrency()

This function performs a conversion between two currencies at a given date.

API syntax

double AmConvertCurrency(long hApiCnxBase, long tmDate, char *strSrcName, char *strDstName, double dVal);

Internal BASIC syntax

Function AmConvertCurrency(tmDate As Date, strSrcName As String, strDstName As String, dVal As Double) As Double

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **tmDate**: This parameter contains the conversion date. It enables you to know the conversion rate in effect on this date.

- **strSrcName**: This parameter contains the source currency for the conversion, i.e. the currency you want to convert.
- **strDstName**: This parameter contains the target currency for the conversion, i.e. the currency in which you want to express the source currency.
- **dVal**: This parameter contains the amount (expressed in the monetary unit of the source currency) to be converted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

Note:

The currency parameters (**strSrcName** and **strDstName**) for this function must be defined in AssetCenter. Furthermore, a valid exchange rate must exist for the date when you want to perform the conversion (**tmDate** parameter).

Example

The following example converts 5,000 FRF into dollars, on the date of November 02, 1998.

```
AmConvertCurrency("1998/11/02 00:00:00", "FRF", "$", 5000)
```

AmConvertDateBasicToUnix()

This function converts a Basic format date ("Date" type) to a Unix format date ("Long" type). This function does not work from external tools because the two types are equivalent.

API syntax

long AmConvertDateBasicToUnix(long hApiCnxBase, long tmTime);

Internal BASIC syntax

Function AmConvertDateBasicToUnix(tmTime As Date) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **tmTime**: This parameter contains the date to be converted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmConvertDateIntlToUnix()

This function converts an international format date ("Date" type) to a Unix format date ("Long" type).

API syntax

long AmConvertDateIntlToUnix(long hApiCnxBase, char *strDate);

Internal BASIC syntax

Function AmConvertDateIntlToUnix(strDate As String) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strDate:** This parameter contains the date to be converted in the international format (yyyy-mm-dd hh:mm:ss).

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmConvertDateStringToUnix()

Converts a date to a string (as displayed in the Windows Control Panel) to a Unix "Long".

API syntax

long AmConvertDateStringToUnix(long hApiCnxBase, char *strDate);

Internal BASIC syntax

Function AmConvertDateStringToUnix(strDate As String) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strDate**: Date as string to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmConvertDateUnixToBasic()

This function converts a Unix format date ("Long" type) to a Basic format date ("Date" type). This function does not work from external tools because the two types are equivalent.

API syntax

long AmConvertDateUnixToBasic(long hApiCnxBase, long lTime);

Internal BASIC syntax

Function AmConvertDateUnixToBasic(lTime As Long) As Date

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓

	Available
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lTime**: This parameter contains the date to be converted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmConvertDateUnixToIntl()

This function converts a Unix format date ("Long" type) to an international format date (yyyy-mm-dd hh:mm:ss).

API syntax

long AmConvertDateUnixToIntl(long hApiCnxBase, long lUnixDate, char *pstrDate, long lDate);

Internal BASIC syntax

Function AmConvertDateUnixToIntl(lUnixDate As Long) As String

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lUnixDate**: This parameter contains the date to be converted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmConvertDateUnixToString()

Converts a "Long" Unix format date to a string format date (as displayed in the Windows Control Panel).

API syntax

long AmConvertDateUnixToString(long hApiCnxBase, long lUnixDate, char *pstrDate, long lDate);

Internal BASIC syntax

Function AmConvertDateUnixToString(lUnixDate As Long) As String

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lUnixDate**: "Long" Unix format date to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmConvertDoubleToString()

This function converts a double precision number to a string. The string is formatted according to the regional options (number) defined in the Windows Control Panel.

API syntax

long AmConvertDoubleToString(double dSrc, char *pstrDst, long lDst);

Internal BASIC syntax

Function AmConvertDoubleToString(dSrc As Double) As String

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dSrc**: This parameter contains the double-precision number to be converted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmConvertMonetaryToString()

This function converts a monetary value to a character string. The string is formatted according to the regional options (currency) defined in the Windows Control Panel.

API syntax

long AmConvertMonetaryToString(double dSrc, char *pstrDst, long lDst);

Internal BASIC syntax

Function AmConvertMonetaryToString(dSrc As Double) As String

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dSrc**: This parameter contains the monetary value you want to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmConvertStringToDouble()

This function converts a character string (in a format corresponding to the one defined in the Windows Control Panel) to a double precision number.

API syntax

double AmConvertStringToDouble(char *strSrc);

Internal BASIC syntax

Function AmConvertStringToDouble(strSrc As String) As Double

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strSrc**: This parameter contains the character string to be converted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmConvertStringToMonetary()

This function converts a character string (in a format corresponding to the one defined in the Windows Control Panel) to a monetary value.

API syntax

double AmConvertStringToMonetary(char *strSrc);

Internal BASIC syntax

Function AmConvertStringToMonetary(strSrc As String) As Double

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strSrc**: This parameter contains the character string to be converted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCounter()

This function returns the value of the **strCounterName** counter, incremented by 1. Zeros are added as padding at the beginning if **iWidth** is greater than the number of digits of the counter. If the counter has more digits than the value stored in **iWidth**, the result will not be truncated.

Internal BASIC syntax

Function AmCounter(strCounterName As String, iWidth As Long) As String

Field of application

Version: 2.52

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	

Input parameters

- **strCounterName**: Name of the counter as it is defined in AssetCenter (access via the **Administration/ Counters** menu item).

- **iWidth:** The value of this parameter forces the output format of the function to be expressed over n digits. This parameter is only useful if the size of the counter is less than the value of this parameter.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following example returns the value of the "Delivery" counter expressed in 5 digits:

```
Dim strCounterName As String
strCounter = AmCounter("Delivery", 5)
```

For example, if the "Delivery" counter equals "18", the function returns:

```
00019
```

AmCreateAssetPort()

The AmCreateAssetPort API creates a new port on a device (lAssetId). The new port contains the given number of pins (iPinCount) of the given cable connector type (lCabCnxTypeId). The status of the pins must be "Available". The pins that will be added to the port are sorted by sequence number. Depending on the port direction (iPinPortDir), the available pins are sorted in ascending (iPinPortDir = 0) or descending (iPinPortDir = 1) order. This function assigns the given duty (lDutyId) to the new port.

API syntax

long AmCreateAssetPort(long hApiCnxBase, long lAssetId, long lCabCnxTypeId, long lDutyId, long iPinCount, long bPinPortDir, long iConnStatus, long bConsecutivePins, long iPrevPinSeq, long bLogError);

Internal BASIC syntax

Function AmCreateAssetPort(lAssetId As Long, lCabCnxTypeId As Long, lDutyId As Long, iPinCount As Long, bPinPortDir As Long, iConnStatus As Long, bConsecutivePins As Long, iPrevPinSeq As Long, bLogError As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lAssetId**: This parameter is the device ID.
- **lCabCnxTypeId**: This parameter is the cable connection type ID.
- **lDutyId**: This parameter is the duty type ID of the port.
- **iPinCount**: This parameter is the pin count that will be used in the new port.
- **bPinPortDir**: This parameter specifies the direction of the port.
- **iConnStatus**
- **bConsecutivePins**
- **iPrevPinSeq**
- **bLogError**

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCreateAssetsAwaitingDelivery()

This function enables you to create the assets that are awaiting receipt

API syntax

long AmCreateAssetsAwaitingDelivery(long hApiCnxBase, long lPOrdId);

Internal BASIC syntax

Function AmCreateAssetsAwaitingDelivery(lPOrdId As Long) As Long

Field of application

Version: 3.61

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lPOrdId**: This parameter contains the identifier of the purchase order concerned

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmCreateCable()

The AmCreateCable API creates a new cable. The cable is created using the given model type (lModelId), the role of the cable (strCableRole), its label rule (lLabelRuleId), its user location (lUserLoc), and its host location (lHostLoc). If the project (lProjectId) and work order (lWorkOrderId) have values, the new cable is added to the project and work order with the given comment (strComment). This comment describes the action that will be performed on the cable (i.e. "Install new cable").

API syntax

long AmCreateCable(long hApiCnxBase, long lModelId, long lUserId, long lHostId, char *strCableRole, long lProjectId, long lWorkOrderId, char *strComment, long lLabelRuleId, char *strLabel);

Internal BASIC syntax

Function AmCreateCable(lModelId As Long, lUserId As Long, lHostId As Long, strCableRole As String, lProjectId As Long, lWorkOrderId As Long, strComment As String, lLabelRuleId As Long, strLabel As String) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lModelId**: This parameter is the cable model ID.
- **lUserId**: This parameter is the user side location ID.
- **lHostId**: This parameter is the host side location ID.
- **strCableRole**: This parameter defines the role of the cable.
- **lProjectId**: This parameter defines the project associated with the placement of the cable.
- **lWorkOrderId**: This parameter defines the work order associated with the placement of the cable.
- **strComment**: This parameter is the comment used on the work order (defined by lWorkOrderId).
- **lLabelRuleId**: This parameter defines the label rule that will be applied to create the label for the cable.
- **strLabel**: This parameter specifies the label affixed to the cable.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCreateCableBundle()

The AmCreateCableBundle API creates a new bundle on a cable (lCableId). The new bundle contains the given number of cable pairs (iPairCount) of the given cable pair type (lPairType). The status of the pairs must be "Available". This function assigns the given duty (lDutyId) to the new bundle.

API syntax

long AmCreateCableBundle(long hApiCnxBase, long lCableId, long lPairTypeId, long lDutyId, long iPairCount, long iStartPairSeq, long bLogError);

Internal BASIC syntax

Function AmCreateCableBundle(lCableId As Long, lPairTypeId As Long, lDutyId As Long, iPairCount As Long, iStartPairSeq As Long, bLogError As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lCableId**: This parameter is the cable ID (it must exist in the cable table).
- **lPairTypeId**: This parameter is the cable pair type ID.
- **lDutyId**: This parameter is the duty of the cable bundle ID.

- **iPairCount**This parameter defines the pair count of the bundle.
- **iStartPairSeq**
- **bLogError**

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCreateCableLink()

The AmCreateCableLink API creates a new cable type cable link for a given cable (lCableId) and bundle (lNextBundle). The duty of the cable link is set using the given duty (lDutyId). The label rule of the cable link is set using the given label rule (lLabelRule).

Note:

The label is not updated using the given label rule, a separate call must be made to AmRefreshLabel().

If a previous link (lPrevLinkId) is specified, a parent link is made between the two records where the previous link is the child.

API syntax

long AmCreateCableLink(long hApiCnxBase, long lCableId, long lDutyId, long lBundleId, long lPrevLinkId, long iTraceDir, long lLabelRuleId);

Internal BASIC syntax

Function AmCreateCableLink(lCableId As Long, lDutyId As Long, lBundleId As Long, lPrevLinkId As Long, iTraceDir As Long, lLabelRuleId As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lCableId**: This parameter is the cable ID for the connection.
- **lDutyId**: This parameter is the connection duty.
- **lBundleId**: This parameter is the ID of the cable bundle to connect.
- **lPrevLinkId**: This parameter defines the cable link ID used to connect. This is optional by using a value of 0.
- **iTraceDir**: This parameter defines the connection direction.
  - 0=host to user
  - 1=user to host
- **lLabelRuleId**: This parameter is the label rule ID used.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCreateDelivFromPO()

This function receives a purchase order and returns the identifier of the receiving slip created.

API syntax

long AmCreateDelivFromPO(long hApiCnxBase, long lPOrdId);

Internal BASIC syntax

Function AmCreateDelivFromPO(lPOrdId As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lPOrdId**: This parameter contains the identifier of the purchase order to be received.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCreateDevice()

The AmCreateDevice API creates a new device. The device is created using the given model type (lProductId) and location (lLocId). The label rule of the asset is set to the given rule (lLabelRuleId).

Note:

The label is not updated using the given label rule, a separate call must be made to AmRefreshLabel.

If the project (lProjectId) and work order (lWorkOrderId) have values, the new asset is added to the project and work order with the comment contained in strComment. This comment describes the action that will be performed on the asset (i.e. "Install new asset").

API syntax

long AmCreateDevice(long hApiCnxBase, long lModelId, long lLocationId, long lProjectId, long lWorkOrderId, long lLabelRuleId, char *strComment);

Internal BASIC syntax

Function AmCreateDevice(lModelId As Long, lLocationId As Long, lProjectId As Long, lWorkOrderId As Long, lLabelRuleId As Long, strComment As String) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lModelId**: This parameter is the model ID for the new device.
- **lLocationId**: This parameter is the location ID for the new device.
- **lProjectId**: The parameter is the project ID. It can be 0.
- **lWorkOrderId**: This parameter is the work order ID. It can be 0.
- **lLabelRuleId**: This parameter defines the label rule ID that will be used for the asset.
- **strComment**: This parameter is the comment that will be used on the work order.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCreateDeviceLink()

The AmCreateDeviceLink API creates a new device type cable link for a given device (lAssetId) and port (lPortId). The label rule of the cable link is set using the given label rule (lLabelRule).

Note:

The label is not updated using the given label rule, a separate call must be made to AmRefreshLabel.

If a previous link (lPrevLinkId) is specified, a parent link is made between the two records. If the trace direction is user-to-host (iTraceDir = 1), then the previous link is the child. If the trace direction is host-to-user (iTraceDir = 0) then the previous link is the parent.

API syntax

long AmCreateDeviceLink(long hApiCnxBase, long lAssetId, long lPortId, long lPrevLinkId, long iTraceDir, long lLabelRuleId);

Internal BASIC syntax

Function AmCreateDeviceLink(lAssetId As Long, lPortId As Long, lPrevLinkId As Long, iTraceDir As Long, lLabelRuleId As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lAssetId**: This parameter contains the identifier of the asset that will be connected.
- **lPortId**: This parameter contains the identifier of the port that will be connected.
- **lPrevLinkId**: This parameter contains the identifier of the device link enabling the connection.
- **iTraceDir**: This parameter specifies the direction of the connection.
  - 0=host to user
  - 1=user to host
- **lLabelRuleId**: This parameter contains the identifier of the label rule used for the new connection.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCreateEstimFromReq()

This function creates an estimate from a purchase request and returns the identifier of the estimate created.

API syntax

long AmCreateEstimFromReq(long hApiCnxBase, long lReqId, long lSuppId);

Internal BASIC syntax

Function AmCreateEstimFromReq(lReqId As Long, lSuppId As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lReqId**: This parameter contains the identifier of the purchase request used to create the estimate.
- **lSuppId**: This parameter contains the identifier of the supplier of the estimate that will be created by the function.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCreateEstimsFromAllReqLines()

This function creates an estimate from a request and returns the identifier of the estimate created.

API syntax

long AmCreateEstimsFromAllReqLines(long hApiCnxBase, long lReqId, long bMergeLines, long lDefSuppId);

Internal BASIC syntax

Function AmCreateEstimsFromAllReqLines(lReqId As Long, bMergeLines As Long, lDefSuppId As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lReqId**: This parameter contains the identifier of the request at the origin of the estimate.
- **bMergeLines**: This parameter enables you to specify if identical request lines are to be combined (**bMergeLines**=1) to given one single line. The quantities given for the lines to be combined are added together and a single line is created.

- **lDefSuppId**: This parameter contains the identifier of the default supplier for the estimate.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmCreateInvFromPO()

This function creates a supplier invoice from a purchase order and returns the identifier of the supplier invoice created.

API syntax

long AmCreateInvFromPO(long hApiCnxBase, long lPOrdId);

Internal BASIC syntax

Function AmCreateInvFromPO(lPOrdId As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lPOrdId**: This parameter contains the identifier of the purchase order at the origin of the invoice.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCreateLink()

This function modifies a link of a record and makes it point to a new record (**hApiRecDest**) in the target table. It therefore creates a link between two records.

API syntax

long AmCreateLink(long hApiRecord, char *strLinkName, long hApiRecDest);

Internal BASIC syntax

Function AmCreateLink(hApiRecord As Long, strLinkName As String, hApiRecDest As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **hApiRecord**: This parameter contains the handle of the record containing the link to be modified.
- **strLinkName**: This parameter contains the SQL name of the link to be modified.
- **hApiRecDest**: This parameter contains a handle of the target record of the link.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmCreatePOFromEstim()

This function creates a purchase order from an estimate and returns the identifier of the purchase order created.

API syntax

long AmCreatePOFromEstim(long hApiCnxBase, long lEstimId);

Internal BASIC syntax

Function AmCreatePOFromEstim(lEstimId As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lEstimId**: This parameter contains the identifier of the estimate used to create the order.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCreatePOFromReq()

This function creates a purchase order from a purchase request and returns the identifier of the PO created.

API syntax

long AmCreatePOFromReq(long hApiCnxBase, long lReqId, long lSuppId);

Internal BASIC syntax

Function AmCreatePOFromReq(lReqId As Long, lSuppId As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lReqId**: This parameter contains the identifier of the purchase request used to create the purchase order.
- **lSuppId**: This parameter contains the identifier of the supplier of the purchase order that will be created by the function.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCreatePOrderFromRequest()

This function enables you to create a purchase order from a request.

API syntax

long AmCreatePOrderFromRequest(long hApiCnxBase, long lRequestId, long lSupplierId);

Internal BASIC syntax

Function AmCreatePOrderFromRequest(lRequestId As Long, lSupplierId As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lRequestId**: This parameter contains the identifier of the request concerned.
- **lSupplierId**: This parameter contains the identifier of the supplier for the purchase order.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCreatePOrdersFromRequest()

This function enables you to create all the purchase orders necessary to satisfy a given request.

API syntax

long AmCreatePOrdersFromRequest(long hApiCnxBase, long lRequestId);

Internal BASIC syntax

Function AmCreatePOrdersFromRequest(lRequestId As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lRequestId**: This parameter contains the identifier of the request concerned

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmCreatePOsFromAllReqLines()

This function creates all the purchase orders from the request lines of a request.

API syntax

long AmCreatePOsFromAllReqLines(long hApiCnxBase, long lReqId, long bMergeLines, long lDefSuppId);

Internal BASIC syntax

Function AmCreatePOsFromAllReqLines(lReqId As Long, bMergeLines As Long, lDefSuppId As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lReqId**: This parameter contains the identifier of the request from which the purchase orders are to be created.
- **bMergeLines**: This parameter enables you to specify if identical request lines are to be combined (**bMergeLines**=1) to given one single line. The quantities given for the lines to be combined are added together and a single line is created.

- **lDefSuppId**: This parameter contains the identifier of the default supplier for the requested items. This parameter is optional and is set to "0" by default.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmCreateProjectCable()

The AmCreateProjectCable API adds a cable (lCableId) to a project (lProjectId) and work order (lWorkOrderId). A comment (strComment) explains the action being performed (i.e. "Install new cable").

API syntax

long AmCreateProjectCable(long hApiCnxBase, long lProjectId, long lWorkOrderId, long lCableId, char *strComment);

Internal BASIC syntax

Function AmCreateProjectCable(lProjectId As Long, lWorkOrderId As Long, lCableId As Long, strComment As String) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lProjectId**: This parameter is the ID of the project that gets the cable.
- **lWorkOrderId**: This parameter is the ID of the work order for the cable.
- **lCableId**: This parameter is the cable ID.
- **strComment**: This parameter is the comment that will be used on the work order.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCreateProjectDevice()

The AmCreateProjectDevice API adds a device (lAssetId) to a project (lProjectId) and work order (lWorkOrderId). A comment (strComment) explains the action being performed (i.e. "Install new device").

API syntax

long AmCreateProjectDevice(long hApiCnxBase, long lProjectId, long lWorkOrderId, long lAssetId, char *strComment);

Internal BASIC syntax

Function AmCreateProjectDevice(lProjectId As Long, lWorkOrderId As Long, lAssetId As Long, strComment As String) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lProjectId**: This parameter defines the ID of the project to get the new device.
- **lWorkOrderId**: This parameter defines the ID of the work order to get the new device.
- **lAssetId**: This parameter is the new device asset ID.
- **strComment**: This parameter is the comment that will be used on the work order.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCreateProjectTrace()

The AmCreateProjectTrace API adds a trace (strTrace) to a project (lProjectId) and work order (lWorkOrderId). The service of the trace is set using the given duty (lDutyId). The trace type (lTraceType) indicates if the trace is a connection (iTraceType = 1) or a disconnection (lTraceType = 2). The label of the user

link being modified (strModLinkLabel) identifies what part of the trace is being modified. A comment (strComment) explains the action being performed (i.e. "Connect these devices").

API syntax

long AmCreateProjectTrace(long hApiCnxBase, long lProjectId, long lWorkOrderId, long iTraceType, long lDutyId, char *strModLinkLabel, char *strTrace, char *strComment);

Internal BASIC syntax

Function AmCreateProjectTrace(lProjectId As Long, lWorkOrderId As Long, iTraceType As Long, lDutyId As Long, strModLinkLabel As String, strTrace As String, strComment As String) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lProjectId**: This parameter defines the project ID to get the trace information.
- **lWorkOrderId**: This parameter defines the work order ID to get the trace information.
- **iTraceType**: This parameter defines the trace type.
  - 1=connection
  - 2=disconnection

- **lDutyId**: This parameter defines the duty. This appears in the work order.
- **strModLinkLabel**: This parameter defines a comment that will be used on the work order.
- **strTrace**: This parameter defines the trace output string that will be used on the work order.
- **strComment**: This parameter is the comment that will be used on the work order.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCreateReceiptFromPOrder()

This function enables you to create a receipt from a purchase order.

API syntax

long AmCreateReceiptFromPOrder(long hApiCnxBase, long lPOrderId);

Internal BASIC syntax

Function AmCreateReceiptFromPOrder(lPOrderId As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lPOrderId**: This parameter contains the identifier of the purchase order concerned.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCreateRecord()

This function creates an empty record in a table taking the default values into account. This new record does not exist in the database until it has been inserted.

API syntax

long AmCreateRecord(long hApiCnxBase, char *strTable);

Internal BASIC syntax

Function AmCreateRecord(strTable As String) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **strTable**: This parameter contains the SQL name of the table in which you want to create the record.

Example

The following example creates an employee in the database:

```
Dim lErr As Long
Dim hRecord As Long
hRecord = amCreateRecord("amEmplDept")
lErr = amSetFieldStrValue(hRecord, "Name", "Doe")
lErr = amSetFieldStrValue(hRecord, "FirstName", "John")
lErr = amInsertRecord(hRecord)
```

AmCreateRequestToInvoice()

This function enables you to create all objects in the procurement cycle: Request, Purchase order, Receipt, Invoice.

API syntax

long AmCreateRequestToInvoice(long hApiCnxBase, float fQty, long lCatRefId, double dUnitPrice, char *strCur, long lRequesterId, long lCostId, long lUserId, long lStockId);

Internal BASIC syntax

Function AmCreateRequestToInvoice(fQty As Single, lCatRefId As Long, dUnitPrice As Double, strCur As String, lRequesterId As Long, lCostId As Long, lUserId As Long, lStockId As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **fQty**: This parameter contains the quantity (in packaged units) to be ordered, then received, then invoiced.
- **lCatRefId**: This parameter contains the identifier of the catalog reference.
- **dUnitPrice**: This parameter contains the unit price of the catalog reference.
- **strCur**: This parameter contains the currency code for the catalog reference price.
- **lRequesterId**: This parameter contains the identifier of the requester.
- **lCostId**: This parameter contains the identifier of the impacted cost center.
- **lUserId**: This parameter contains the identifier of the user of the ordered item.
- **lStockId**: This parameter contains the identifier of the delivery stock of the item.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

Equivalent to the sequence of calls: amCreateRequestToReceipt, amCreateOrUpdateInvoiceFromReceipt.

AmCreateRequestToPOrder()

This function enables you to create the objects in the procurement cycle: Request, Purchase order.

API syntax

long AmCreateRequestToPOrder(long hApiCnxBase, float fQty, long lCatRefId, double dUnitPrice, char *strCur, long lRequesterId, long lCostId, long lUserId, long lStockId);

Internal BASIC syntax

Function AmCreateRequestToPOrder(fQty As Single, lCatRefId As Long, dUnitPrice As Double, strCur As String, lRequesterId As Long, lCostId As Long, lUserId As Long, lStockId As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	

	Available
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **fQty**: This parameter contains the quantity (in packaged units) to be ordered.
- **lCatRefId**: This parameter contains the identifier of the catalog reference.
- **dUnitPrice**: This parameter contains the unit price of the catalog reference.
- **strCur**: This parameter contains the currency code for the catalog reference price.
- **lRequesterId**: This parameter contains the identifier of the requester.
- **lCostId**: This parameter contains the identifier of the impacted cost center.
- **lUserId**: This parameter contains the identifier of the user of the ordered item.
- **lStockId**: This parameter contains the identifier of the delivery stock of the item.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCreateRequestToReceipt()

This function enables you to create the objects in the procurement cycle: Request, Purchase order, Receipt.

API syntax

long AmCreateRequestToReceipt(long hApiCnxBase, float fQty, long lCatRefId, double dUnitPrice, char *strCur, long lRequesterId, long lCostId, long lUserId, long lStockId);

Internal BASIC syntax

Function AmCreateRequestToReceipt(fQty As Single, lCatRefId As Long, dUnitPrice As Double, strCur As String, lRequesterId As Long, lCostId As Long, lUserId As Long, lStockId As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **fQty**: This parameter contains the quantity (in packaged units) to be ordered, then received.
- **lCatRefId**: This parameter contains the identifier of the catalog reference.
- **dUnitPrice**: This parameter contains the unit price of the catalog reference.
- **strCur**: This parameter contains the currency code for the catalog reference price.
- **lRequesterId**: This parameter contains the identifier of the requester.
- **lCostId**: This parameter contains the identifier of the impacted cost center.
- **lUserId**: This parameter contains the identifier of the user of the ordered item.

- **lStockId**: This parameter contains the identifier of the delivery stock of the item.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

Equivalent to the sequence of calls: amCreateRequestToPOrder, amCreateReceiptFromPOrder.

AmCreateReturnFromReceipt()

This function enables you to create a return slip from a receiving slip.

API syntax

long AmCreateReturnFromReceipt(long hApiCnxBase, long lRecptId);

Internal BASIC syntax

Function AmCreateReturnFromReceipt(lRecptId As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lRecptId**: This parameter contains the identifier of the receipt line.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCreateTraceHist()

The AmCreateTraceHist API is for creating trace history and trace operation based on an existing connection from a source device/cable to a destination device/cable.

API syntax

long AmCreateTraceHist(long hApiCnxBase, long lSrcLinkId, long lDestLinkId, long iTraceDir, long lCabTraceOutId, char *strComment);

Internal BASIC syntax

Function AmCreateTraceHist(lSrcLinkId As Long, lDestLinkId As Long, iTraceDir As Long, lCabTraceOutId As Long, strComment As String) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lSrcLinkId**: This parameter is the device/cable used for the source link.
- **lDestLinkId**: This parameter is the device/cable used for the destination link.
- **iTraceDir**: This parameter specifies the direction of the connection.
  - 0=host to user
  - 1=user to host
- **lCabTraceOutId**: This parameter is the cable trace output ID.
- **strComment**: This parameter is the comment to be associated with the trace operation.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmCryptPassword()

This function encrypts the password of a user, identified by a login and password.

API syntax

long AmCryptPassword(char *strUser, char *strPasswd, char *pStrCrypted, long lpStrCrypted);

Internal BASIC syntax

Function AmCryptPassword(strUser As String, strPasswd As String) As String

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strUser**: This parameter contains the login of the user whose password you want to encrypt.
- **strPasswd**: This parameter contains, in plaintext, the password to be encrypted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCurrentDate()

This function returns the current date on the client workstation.

API syntax

long AmCurrentDate();

Internal BASIC syntax

Function AmCurrentDate() As Date

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

If the database is configured to use time zones, the behavior of this function differs depending on whether it is called directly from AssetCenter or from an external program. In AssetCenter, this function behaves in the same way as the Now() function in Basic. From external programs, the value returned by this function is expressed as GMT+0 and does not take daylight savings into account.

AmCurrentIsoLang()

This function returns the ISO language code of the language used in AssetCenter ("en" for English, "fr" for French, etc.).

API syntax

long AmCurrentIsoLang(char *pstrLanguage, long lLanguage);

Internal BASIC syntax

Function AmCurrentIsoLang() As String

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓

	Available
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCurrentLanguage()

This function returns the language version of AssetCenter ("US" for English, "FR" for French, etc.).

API syntax

long AmCurrentLanguage(char *pstrLanguage, long lLanguage);

Internal BASIC syntax

Function AmCurrentLanguage() As String

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓

	Available
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmCurrentServerDate()

This function returns the current date on the server.

API syntax

long AmCurrentServerDate(long hApiCnxBase);

Internal BASIC syntax

Function AmCurrentServerDate() As Date

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓

	Available
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmDateAdd()

This function calculates a new date according to a start date to which a real duration is added.

API syntax

long AmDateAdd(long tmStart, long tsDuration);

Internal BASIC syntax

Function AmDateAdd(tmStart As Date, tsDuration As Long) As Date

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓

	Available
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **tmStart**: This parameter contains the date to which the duration is added.
- **tsDuration**: This parameter contains the duration, expressed in seconds, to be added to the date **tmStart**.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following example illustrates the difference between the amDateAdd() and amDateAddLogical() functions. A duration of 30 days will be added to the date 1/1/1999 (January 1, 1999) using each of these functions.

AmDateAdd adds a real duration, 30 days in this case:

```
RetVal=AmDateAdd("1999/01/01", 2592000)
```

The function returns the value:

```
1999/01/31
```

AmDateAddLogical adds a logical duration, in this case 30 days (=1 month):

```
RetVal=AmDateAddLogical("1999/01/01", 2592000)
```

The function returns the value:

```
1999/02/01
```

AmDateAddLogical()

This function calculates a new date according to a start date to which a logical duration is added (1 month contains 30 days).

API syntax

long AmDateAddLogical(long tmStart, long tsDuration);

Internal BASIC syntax

Function AmDateAddLogical(tmStart As Date, tsDuration As Long) As Date

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **tmStart**: This parameter contains the date to which the duration is added.
- **tsDuration**: This parameter contains the duration, expressed in seconds, to be added to the date **tmStart**.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following example illustrates the difference between the amDateAdd() and amDateAddLogical() functions. A duration of 30 days will be added to the date 1/1/1999 (January 1, 1999) using each of these functions.

AmDateAdd adds a real duration, 30 days in this case:

```
RetVal=AmDateAdd("1999/01/01", 2592000)
```

The function returns the value:

```
1999/01/31
```

AmDateAddLogical adds a logical duration, in this case 30 days (=1 month):

```
RetVal=AmDateAddLogical("1999/01/01", 2592000)
```

The function returns the value:

```
1999/02/01
```

AmDateDiff()

This function calculates in the seconds the duration (or timespan) between two dates.

API syntax

long AmDateDiff(long tmEnd, long tmStart);

Internal BASIC syntax

Function AmDateDiff(tmEnd As Date, tmStart As Date) As Date

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **tmEnd**: This parameter contains the end date of the period for which the calculation is carried out.
- **tmStart**: This parameter contains the start date of the period for which the calculation is carried out.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following example calculates the time elapsed between 01/01/98 and 01/01/99.

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

AmDbGetDate()

This function returns the result, in date format, of the AQL query. If the query does not return a result, the value 0 is returned without triggering an error.

API syntax

long AmDbGetDate(long hApiCnxBase, char *strQuery);

Internal BASIC syntax

Function AmDbGetDate(strQuery As String) As Date

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strQuery**: This parameter contains the full AQL query whose result you want to recover.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmDbGetDouble()

This function returns the result (as a double-precision number), of the AQL query. If the query does not return a result, the value 0 is returned without triggering an error.

API syntax

double AmDbGetDouble(long hApiCnxBase, char *strQuery);

Internal BASIC syntax

Function AmDbGetDouble(strQuery As String) As Double

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strQuery**: This parameter contains the full AQL query whose result you want to recover.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmDbGetList()

This function returns, as a list, the result of an AQL query. The number of elements selected by the AQL query is limited to 99.

API syntax

long AmDbGetList(long hApiCnxBase, char *strQuery, char *pstrResult, long lResult, char *strColSep, char *strLineSep, char *strIdSep);

Internal BASIC syntax

Function AmDbGetList(strQuery As String, strColSep As String, strLineSep As String, strIdSep As String) As String

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strQuery**: This parameter contains the AQL query you want to execute.
- **strColSep**: This parameter contains the character used as column separator in the result given by the function.
- **strLineSep**: This parameter contains the character used as line separator in the result given by the function.
- **strIdSep**: This parameter contains the character used as identifier separator in the result given by the function.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmDbGetListEx()

This function returns, as a list, the result of an AQL query. Unlike the **AmDbGetList** function, this function is not limited in the number of elements selected by the AQL query.

API syntax

long AmDbGetListEx(long hApiCnxBase, char *strQuery, char *pstrResult, long lResult, char *strColSep, char *strLineSep, char *strIdSep);

Internal BASIC syntax

Function AmDbGetListEx(strQuery As String, strColSep As String, strLineSep As String, strIdSep As String) As String

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strQuery**: This parameter contains the AQL query you want to execute.
- **strColSep**: This parameter contains the character used as column separator in the result given by the function.
- **strLineSep**: This parameter contains the character used as line separator in the result given by the function.
- **strIdSep**: This parameter contains the character used as identifier separator in the result given by the function.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmDbGetLong()

This function returns the result of an AQL query. If the query does not return a result, the value 0 is returned without triggering an error.

API syntax

long AmDbGetLong(long hApiCnxBase, char *strQuery);

Internal BASIC syntax

Function AmDbGetLong(strQuery As String) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strQuery**: This parameter contains the full AQL query whose result you want to recover.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following example returns the identifier of a product supplier:

```
AmDbGetLong("SELECT lSuppId FROM amProdSupp WHERE lProdId="+Str([ProdId])
+")
```

AmDbGetPk()

This function returns the primary key of a table according to the WHERE clause in an AQL query. If the query does not return a result, the value 0 is returned without triggering an error.

API syntax

long AmDbGetPk(long hApiCnxBase, char *strTableName, char *strWhere);

Internal BASIC syntax

Function AmDbGetPk(strTableName As String, strWhere As String) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strTableName:** SQL name of the table whose primary key you want to recover.
- **strWhere:** WHERE clause in an AQL query.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmDbGetString()

This function returns the result of an AQL query as a formatted string. The number of elements selected by the AQL query is limited to 99.

Warning:

Do not use this function to recover the value of a single string type field. This function is similar to the AmDbGetList and AmDbGetListEx functions.

API syntax

long AmDbGetString(long hApiCnxBase, char *strQuery, char *pstrResult, long lResult, char *strColSep, char *strLineSep);

Internal BASIC syntax

Function AmDbGetString(strQuery As String, strColSep As String, strLineSep As String) As String

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strQuery**: This parameter contains the AQL query you want to execute.
- **strColSep**: This parameter contains the character used as column separator in the final string.
- **strLineSep**: This parameter contains the character used as line separator in the final string.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

In the API syntax, the lResult parameter must contain the expected size of the resulting value.

Example

```
Dim strList As String
strList = amDbGetList("Select Name, FullName from amEmplDept Where Name L
ike 'C%'", "|", ",", "=")
```

returns the string:

```
Carpenter|/Taltek/I.S. Department/Carpenter\, Jerome\, DEMO-M016/=23459,C
havez|/Taltek/I.S. Department/Chavez\, Philip\, DEMO-M014/=23460,Chouraqu
i|/Taltek/Sales/Los Angeles Agency/Chouraqui\, Thomas\, DEMO-M017/=23491,
Cipriani|/Taltek/Sales/Los Angeles Agency/Cipriani\, Fred\, DEMO-M018/=23
492,Clech|/Taltek/Sales/Burbank Agency/Clech\, Richard\, DEMO-M021/=23482
,Colombo|/Taltek/Finance/Colombo\, Gerald\, DEMO-M022/=23441
```

The escape character \ is used before commas.

The same query with **amDbGetString()** does not add escape characters, which makes it inappropriate to fill a list. For example:

```
amDbGetString("Select FullName from amEmplDept Where Name Like 'C%'", "|"
, chr(10), "")
```

displays:

```
/Taltek/I.S. Department/Carpenter, Jerome, DEMO-M016/
/Taltek/I.S. Department/Chavez, Philip, DEMO-M014/
/Taltek/Sales/Los Angeles Agency/Chouraqui, Thomas, DEMO-M017/
/Taltek/Sales/Los Angeles Agency/Cipriani, Fred, DEMO-M018/
/Taltek/Sales/Burbank Agency/Clech, Richard, DEMO-M021/
/Taltek/Finance/Colombo, Gerald, DEMO-M022/
```

AmDbGetStringEx()

This function returns, as a character string , the result of an AQL query. The difference with the **AmDbGetString** function is that this function is not limited in the number of elements selected by the AQL query.

Warning:

Do not use this function to recover the value of a single string type field. This function is similar to the AmDbGetList and AmDbGetListEx functions.

API syntax

long AmDbGetStringEx(long hApiCnxBase, char *strQuery, char *pstrResult, long lResult, char *strColSep, char *strLineSep);

Internal BASIC syntax

Function AmDbGetStringEx(strQuery As String, strColSep As String, strLineSep As String) As String

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strQuery**: This parameter contains the AQL query you want to execute.
- **strColSep**: This parameter contains the character used as column separator in the final string.
- **strLineSep**: This parameter contains the character used as line separator in the final string.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmDeadLine()

This function calculates a deadline according to a calendar, a start date and a number of working seconds elapsed.

API syntax

long AmDeadLine(char *strCalendarSqlName, long tmStart, long tsDuration);

Internal BASIC syntax

Function AmDeadLine(strCalendarSqlName As String, tmStart As Date, tsDuration As Long) As Date

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strCalendarSqlName**: This parameter contains the SQL name of the calendar of working periods used as a basis for calculating the deadline.

- **tmStart**: This parameter contains the start date of the period.
- **tsDuration**: This parameter contains the number of working seconds since the start date of the period.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following example calculates the deadline according to the calendar whose SQL name is "Calendar_Paris", from a period start date set to 01/09/1998 at 8 a.m. and for a number of seconds equal to 450,000.

```
AmDeadLine("Calendar_Paris", "1998/09/01 08:00:00", 450000)
```

This example returns the deadline, i.e. 22/09/1998 at 6 p.m.

AmDecrementLogLevel()

This function enables you to go up one level in the hierarchy of a log window in the final page of a wizard.

Internal BASIC syntax

Function AmDecrementLogLevel() As Long

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmDefAssignee()

This function searches for the ID number of the default ticket supervisor for a given employee group.

API syntax

long AmDefAssignee(long hApiCnxBase, long lGroupId);

Internal BASIC syntax

Function AmDefAssignee(lGroupId As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓

	Available
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lGroupId**: This parameter contains the ID number of an employee group.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following generic example returns the identifier of the default supervisor for an employee group:

```
AmDefAssignee([lGroupId])
```

You can directly enter the numeric value of the identifier, as in the following example:

```
AmDefAssignee(24)
```

AmDefaultCurrency()

Returns the default currency used in AssetCenter.

API syntax

long AmDefaultCurrency(long hApiCnxBase, char *return, long lreturn);

Internal BASIC syntax

Function AmDefaultCurrency() As String

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmDefEscalationScheme()

This function searches for the default escalation scheme according to the location and severity of the helpdesk ticket.

API syntax

long AmDefEscalationScheme(long hApiCnxBase, char *strLocFullName, long lSeverityLvl);

Internal BASIC syntax

Function AmDefEscalationScheme(strLocFullName As String, lSeverityLvl As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strLocFullName**: This parameter contains the full name of the location.
- **lSeverityLvl**: This parameter contains the value of the severity.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following generic example returns the identifier of the default escalation scheme according to the location and the severity:

```
AmDefEscalationScheme([Asset.Location.FullName], [Severity.lSeverityLvl])
```

You can directly enter the value of the parameters, as in the following example:

```
AmDefEscalationScheme ( "/Location/", 24)
```

AmDefGroup()

This function returns the ID number of the default helpdesk group according to the type of problem, the location, and the maintenance contract.

API syntax

long AmDefGroup(long hApiCnxBase, long lProblemClassId, char *strLocFullName, long lAssetMainCntId);

Internal BASIC syntax

Function AmDefGroup(lProblemClassId As Long, strLocFullName As String, lAssetMainCntId As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lProblemClassId**: This parameter contains the ID number for a problem type.
- **strLocFullName**: This parameter contains the full name of a location.
- **lAssetMainCntId**: This parameter contains the ID number of a maintenance contract.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

The method used to define the default helpdesk group is the following:

1. The function searches the helpdesk groups associated with the problem type of the ticket.
2. From these groups, the function searches the helpdesk groups associated with the "nearest" location to the asset: direct location, else parent location, and so on until the root location.
3. If no group is found, and if the DBMS supports double outer joins, the function searches groups that aren't associated with a location.

   For the list of DBMSs that support double external joints, refer to the **Helpdesk** guide, chapter **References (Helpdesk)**, section **DBMSs supporting double outer-joins**.
4. If the DBMS supports double outer joins, the function selects, from the groups found previously, the helpdesk group linked to maintenance contracts covering the asset.
5. If no group is found, the function repeats steps 1, 2, 3 and 4 starting from the problem type a level up in the hierarchy of problems until it reaches the root of the problem-type tree.

Example

The following generic example calculates the ID number of the default helpdesk group according to three parameters: the type of problem, the location, and the maintenance contract.

```
AmDefGroup([ProblemClass.lPbClassId],[Asset.Location.FullName],[Asset.lMa
intCntrId])
```

You can directly enter the numeric value of the parameters using the ID numbers, as shown in the following example:

```
AmDefGroup(0, [Asset.Location.FullName], 0)
```

AmDeleteLink()

This function deletes a links of a record.

API syntax

long AmDeleteLink(long hApiRecord, char *strLinkName, long hApiRecDest);

Internal BASIC syntax

Function AmDeleteLink(hApiRecord As Long, strLinkName As String, hApiRecDest As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **hApiRecord**: This parameter contains the handle of the record containing the link to be deleted.
- **strLinkName**: This parameter contains the SQL name of the link to be deleted.
- **hApiRecDest**: This parameter contains a handle of the target record of the link.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmDeleteRecord()

This function deletes a record in the database.

API syntax

long AmDeleteRecord(long hApiRecord);

Internal BASIC syntax

Function AmDeleteRecord(hApiRecord As Long) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	

	Available
FINISH.DO script of a wizard	✓

Input parameters

- **hApiRecord**: This parameter contains a handle of the record you want to delete.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmDisconnectTrace()

The AmDisconnectTrace API disconnects the trace between a user node (lEndId) and host node (lStartId) in the cable link table. If either node is at the end of a trace, it will be deleted from the cable link table. It also creates trace history and trace operations entries based on the disconnect.

API syntax

long AmDisconnectTrace(long hApiCnxBase, long lStartId, long lEndId, char *strComment, long lCabTraceOutId);

Internal BASIC syntax

Function AmDisconnectTrace(lStartId As Long, lEndId As Long, strComment As String, lCabTraceOutId As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lStartId**: This parameter defines the host connection ID that will be disconnected.
- **lEndId**: This parameter defines the user connection ID that will be disconnected.
- **strComment**: This parameter is the string operation to show new connects and disconnects.
- **lCabTraceOutId**: This parameter is the cable trace output ID.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmDuplicateRecord()

This function enables you to duplicate a record.

API syntax

long AmDuplicateRecord(long hApiRecord, long bInsert);

Internal BASIC syntax

Function AmDuplicateRecord(hApiRecord As Long, bInsert As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **hApiRecord**: This parameter contains the handle of the record to duplicate.
- **bInsert**: This parameter enables you to specify whether you want to insert the duplicated record immediately (=1) or not (=0).

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmEndOfNthBusinessDay()

Gives the last business hour of the nth day (identified by the integer **lDayCount**) from a given date according to a calendar.

API syntax

long AmEndOfNthBusinessDay(char *strCalendarSqlName, long tmStart, long lDayCount);

Internal BASIC syntax

Function AmEndOfNthBusinessDay(strCalendarSqlName As String, tmStart As Date, lDayCount As Long) As Date

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strCalendarSqlName**: Name of the calendar used for the calculation.
- **tmStart**: Start date for the calculation.
- **lDayCount**: Number of full business days to add to dStart for the calculation.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmEnumValList()

This function returns a string containing all the values of a system itemized list. The different values are sorted alphabetically and are delimited by the separator indicated in the **strLineSep** parameter.

If an itemized list value contains the character used as the separator or a "\", the "\" prefix is used.

API syntax

long AmEnumValList(long hApiCnxBase, char *strEnumName, char *pstrValList, long lValList, long bNoCase, char *strLineSep);

Internal BASIC syntax

Function AmEnumValList(strEnumName As String, bNoCase As Long, strLineSep As String) As String

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strEnumName**: This parameter contains the SQL name of the system-itemized list for which you want to recover the values.
- **bNoCase**: This parameter enables you to specify whether the sort is case sensitive (=1) or not (=0).

- **strLineSep**: This parameter contains the character used to delimit the itemized-list values.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmEvalScript()

This function enables you to evaluate a script by its name from the current context. This function has two uses:

- Evaluate a system script (Default value, Mandatory, etc,)
- Call a function from a script library.

Internal BASIC syntax

Function AmEvalScript(strScriptName As String, strObject As String, strPath As String, ...) As Variant

Field of application

Version: 4.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strScriptName**: This parameter contains the name of the script to evaluate. In the first case, it is a system name (DefVal, etc.). In the second case, it is the name of a function belonging to a script library (the name of the library is then specified by the strObject parameter).
- **strObject**: This parameter contains the object concerned by the script. It can be the SQL name of a field or the name of a library.
- **strPath**: This optional parameter enables you to specify a path (link.link.link...) to shift the context of evaluation of a script. This parameter does not work in the second case.
- **...**: When calling a function from a script library, enables you to pass parameters to the function called.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

The following is a list of usable system script names:

- For a table: IsValidScript, IsRelevantScript
- For a field: DefValScript, MandatoryScript, HistorizedScript, ReadOnlyScript, IrrelevantScript
- For a link: HistorizedScript, FilterScript, IrrelevantScript
- For a feature: DefValScript, MandatoryScript, AvailableScript, HistorizedScript

AmExecTransition()

This function triggers a valid transition from the current page.

Internal BASIC syntax

Function AmExecTransition(strTransName As String) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **strTransName**: This parameter contains the name of the transition as defined in the wizard script. An error is returned if the transition is not found. The function does not work (and does not return an error) if the transition is not valid.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmExecuteActionById()

This function executes an action as identified by its identifier.

API syntax

long AmExecuteActionById(long lActionId, char *strTableName, long lRecordId);

Internal BASIC syntax

Function AmExecuteActionById(lActionId As Long, strTableName As String, lRecordId As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lActionId**: This parameter contains the identifier of the action to be executed.
- **strTableName**: In the case of a contextual action, this parameter contains the SQL name of the table on which the action is executed. If this parameter is omitted, in the case of a contextual action, the function will fail. For non-contextual actions, this parameter is not interpreted and therefore optional.
- **lRecordId**: This parameter contains the identifier of a possible record concerned by the action. For non-contextual actions, this parameter is not interpreted and therefore optional.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmExecuteActionByName()

This function executes an action as identified by its SQL name.

API syntax

long AmExecuteActionByName(char *strSqlName, char *strTableName, long lRecordId);

Internal BASIC syntax

Function AmExecuteActionByName(strSqlName As String, strTableName As String, lRecordId As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **strSqlName**: This parameter contains the SQL name of the action to be executed.
- **strTableName**: In the case of a contextual action, this parameter contains the SQL name of the table on which the action is executed. If this parameter is omitted, in the case of a contextual action, the function will fail. For non-contextual actions, this parameter is not interpreted and therefore optional.

- **lRecordId**: This parameter contains the identifier of a possible record concerned by the action. For non-contextual actions, this parameter is not interpreted and therefore optional.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmExportDocument()

This function enables you to export a document attached to a record.

API syntax

long AmExportDocument(long hApiCnxBase, long lDocId, char *strFileName);

Internal BASIC syntax

Function AmExportDocument(lDocId As Long, strFileName As String) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lDocId**: This parameter contains the identifier of the document to export.
- **strFileName**: This parameter contains the name of the document to export, as it is stored in the FileName field of the Documents table.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmFindCable()

The AmFindCable API finds the next available cable that runs between a given user location (lUserId) and host location (lHostId). The cable must be of the specified cable type (strCabType) and cable role (strCableRole). The cable must also have a status of "Available". The cables are sorted in ascending order by cable ID and only cables greater than the previous cable ID (lPrevCabId) are selected.

API syntax

long AmFindCable(long hApiCnxBase, long lPrevCableId, char *strCabType, long lUserId, long lHostId, char *strCableRole);

Internal BASIC syntax

Function AmFindCable(lPrevCableId As Long, strCabType As String, lUserId As Long, lHostId As Long, strCableRole As String) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lPrevCableId**: This parameter is the ID of the previous cable.
- **strCabType**: This parameter defines the cable type for searching.
- **lUserId**: This parameter defines the user location ID.
- **lHostId**: This parameter defines the host location ID.
- **strCableRole**: This parameter is the cable role to locate.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmFindDevice()

The AmFindDevice API finds a device of a given type (strDevType) in a given location (lLocId). The devices are sorted in ascending order by device ID and only devices greater than the previous device ID (lPrevDeviceId) are selected.

API syntax

long AmFindDevice(long hApiCnxBase, long lPrevDeviceId, char *strDeviceType, long lLocationId);

Internal BASIC syntax

Function AmFindDevice(lPrevDeviceId As Long, strDeviceType As String, lLocationId As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lPrevDeviceId**: This parameter defines the previous device ID searched. The value of 0 is used to start a search.
- **strDeviceType**: This parameter defines the device type to locate.
- **lLocationId**: This parameter is the location ID to search.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmFindRootLink()

This function enables you to recover the root link of a trace.

API syntax

long AmFindRootLink(long hApiCnxBase, long lLinkId);

Internal BASIC syntax

Function AmFindRootLink(lLinkId As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lLinkId**: This parameter contains the identifier of the link concerned by the operation.

Output parameters

The function returns the identifier of the root link.

AmFindTermDevice()

The AmFindTermDevice API finds the next available device in a given termination field (lTermField) for a given cable role (strCableRole). The devices are sorted in ascending order by sequence number and only assets greater than the previous sequence number (strPrevTermSeq) are selected. Also, for pin-based devices (bPinBased = 1), we check the total number of pins needed

(iPinPortCount) against the total number of pins remaining on the device. For port-based devices (bPinBased = 0) we check to make sure there is at least one port remaining on the device and that the remaining port has the host or user side available by the checking the flag (bCheckAvail = 0 - user device, bCheckAvail = 1 - host device).

API syntax

long AmFindTermDevice(long hApiCnxBase, long iPrevTermSeq, long lTermFieldId, char *strCableRole, long bPinBased, long iPinPortCount, long bCheckAvail);

Internal BASIC syntax

Function AmFindTermDevice(iPrevTermSeq As Long, lTermFieldId As Long, strCableRole As String, bPinBased As Long, iPinPortCount As Long, bCheckAvail As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **iPrevTermSeq**: This parameter is the previous termination field's sequence searched. The value of 0 is used to start a search.
- **lTermFieldId**: This parameter is the termination field ID.
- **strCableRole**: This parameter is the cable role to locate.

- **bPinBased**: This parameter determines whether the device is pin-based or port-based.
- **iPinPortCount**: For pin-based devices, this parameter is the total number of pins needed to create a virtual port. For port-based devices, this parameter is 1 since this API is called per port needed.
- **bCheckAvail**: This parameter is used to determine what side of the port needs to be available.
  - 0=user device, check host available
  - 1=host device, check user available

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmFindTermField()

The AmFindTermField API finds a termination field that provides the given duty (lDutyId) from the given location (lLocId). It will continue to find additional termination fields in a given location for a given duty if lTermFieldId is greater than 0.

API syntax

long AmFindTermField(long hApiCnxBase, long lDutyId, long lLocationId, long lPrevTermFieldId);

Internal BASIC syntax

Function AmFindTermField(lDutyId As Long, lLocationId As Long, lPrevTermFieldId As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lDutyId**: This parameter defines the duty to locate.
- **lLocationId**: This parameter is the location ID to search.
- **lPrevTermFieldId**: This parameter is the termination field ID.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGenSqlName()

This function generates a valid SQL name from a classic string. Spaces are replaced by underscores ("_"). This function is especially useful for defining the default value of a SQL name for a feature based on its name.

API syntax

long AmGenSqlName(char *return, long lreturn, char *strText);

Internal BASIC syntax

Function AmGenSqlName(strText As String) As String

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strText:** Character string used to generate the SQL name.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following example defines the default value of the SQL name of an object called "Label" in the AssetCenter database:

```
RetVal=AmGenSQLName([Label])
```

AmGetCatRef()

This function searches for a valid non-catalog reference for a given model (respecting the validity dates). The following rules are respected:

- `CatProduct.lModelId=lModelId`
- `CatProduct.lParentId=0`

The function returns a reference not created on the fly as its priority. If no references are found and the parameter **bCreate** is set to "1", a new non-catalog reference and a product are created (pointing to the model).

API syntax

long AmGetCatRef(long hApiCnxBase, long lModelId, long bCreate);

Internal BASIC syntax

Function AmGetCatRef(lModelId As Long, bCreate As Long) As Long

Field of application

Version: 4.1.0

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lModelId**: This parameter contains the ID of the model concerned by the operation.

- **bCreate**: This parameter enables you to specify if a non-catalog reference should be created in the case where the search returns no results.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

Note:

The function does not require you to specify a supplier. This is because the search is performed on non-catalog references regardless of the supplier.

AmGetCatRefFromCatProduct()

This function is identical to the function **amGetCatRef**, except that the search is performed for a given product.

API syntax

long AmGetCatRefFromCatProduct(long hApiCnxBase, long lCatProductId, long bCreate);

Internal BASIC syntax

Function AmGetCatRefFromCatProduct(lCatProductId As Long, bCreate As Long) As Long

Field of application

Version: 4.1.0

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lCatProductId**: This parameter contains the ID of the product concerned by the operation.
- **bCreate**: This parameter enables you to specify if a non-catalog reference should be created in the case where the search returns no results.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetComputeString()

This function returns the description string of a given record according to a template.

API syntax

long AmGetComputeString(long hApiCnxBase, char *strTableName, long lRecordId, char *strTemplate, char *pstrComputeString, long lComputeString);

Internal BASIC syntax

Function AmGetComputeString(strTableName As String, lRecordId As Long, strTemplate As String) As String

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strTableName**: This parameter contains the SQL name of the table of the record for which you want to recover the description string.
- **lRecordId**: This parameter contains the identifier of the record within the table.
- **strTemplate**: This parameter contains, in the form of a character string, the template used for the description string.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
RetVal = amGetComputeString("amEmplDept", [lEmplDeptId], "[Name], [FirstN
ame]")
```

AmGetCurrentNTDomain()

This function returns the name of the NT domain of the current login.

API syntax

long AmGetCurrentNTDomain(char *pstrDomain, long lDomain);

Internal BASIC syntax

Function AmGetCurrentNTDomain() As String

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
RetVal = amGetCurrentNTDomain()
```

AmGetCurrentNTUser()

This function enables you to get the login of the user connected to Windows (NT or 2000).

API syntax

long AmGetCurrentNTUser(char *pstrUser, long lUser);

Internal BASIC syntax

Function AmGetCurrentNTUser() As String

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓

	Available
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetFeat()

This function creates a feature object based on its name and returns the handle of the feature object created.

API syntax

long AmGetFeat(long hApiTable, long lPos);

Internal BASIC syntax

Function AmGetFeat(hApiTable As Long, lPos As Long) As Long

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓

	Available
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiTable**: This parameter contains a handle of a table.
- **lPos**: This parameter contains the position of the feature in the table.

AmGetFeatCount()

This function returns the number of features of the table specified in the **hApiTable** parameter.

API syntax

long AmGetFeatCount(long hApiTable);

Internal BASIC syntax

Function AmGetFeatCount(hApiTable As Long) As Long

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiTable**: This parameter contains a handle of a table.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetField()

This function creates a field object from the handle of a query, a record or a table and returns the handle of the field object created.

API syntax

long AmGetField(long hApiObject, long lPos);

Internal BASIC syntax

Function AmGetField(hApiObject As Long, lPos As Long) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓

	Available
FINISH.DO script of a wizard	✓

Input parameters

- **hApiObject**: This parameter contains a handle of a query, record, or table.
- **lPos**: This parameter contains the position of the field (its index) within the object.

AmGetFieldCount()

This function returns the number of fields contained in the current object.

API syntax

long AmGetFieldCount(long hApiObject);

Internal BASIC syntax

Function AmGetFieldCount(hApiObject As Long) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiObject**: This parameter contains a handle of a valid record, query or table.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetFieldDateValue()

This function returns the value of a field contained in the current object. This value is returned in "Date" format (from external tools, it is a Long).

API syntax

long AmGetFieldDateValue(long hApiObject, long lFieldPos);

Internal BASIC syntax

Function AmGetFieldDateValue(hApiObject As Long, lFieldPos As Long) As Date

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓

	Available
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiObject**: This parameter contains a handle of a query or record.
- **lFieldPos**: This parameter contains the number of the field inside the current object.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetFieldDescription()

This function returns, as a character string ("String" format), the long description of a field identified by a handle.

API syntax

long AmGetFieldDescription(long hApiField, char *pstrBuffer, long lBuffer);

Internal BASIC syntax

Function AmGetFieldDescription(hApiField As Long) As String

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: This parameter contains a valid handle of the field whose long description you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetFieldDoubleValue()

This function returns the value of a field contained in the current object. This value is returned in "Double" format.

API syntax

double AmGetFieldDoubleValue(long hApiObject, long lFieldPos);

Internal BASIC syntax

Function AmGetFieldDoubleValue(hApiObject As Long, lFieldPos As Long) As Double

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiObject**: This parameter contains a handle of a query or record.
- **lFieldPos**: This parameter contains the number of the field inside the current object.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetFieldFormat()

This function is useful when the value of the "UserType" of the field concerned (cf. "database.txt" file) is:

- System itemized list
- Itemized list
- Time span
- Table or field name

The function returns the format of the "UserType", i.e.:

UserType	Format returned by the function
System itemized list	List of system-itemized list entries.
Itemized list	Name of the itemized list associated to the field.
Time span	Display format.
Table or field name	SQL name of the field that stores the SQL name of the table.

API syntax

long AmGetFieldFormat(long hApiField, char *pstrBuffer, long lBuffer);

Internal BASIC syntax

Function AmGetFieldFormat(hApiField As Long) As String

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: This parameter contains a valid handle of the field whose "UserType" you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetFieldFormatFromName()

This function returns the "UserType" format of a field, from its name.

API syntax

long AmGetFieldFormatFromName(long hApiCnxBase, char *strTableName, char *strFieldName, char *pFieldFormat, long lpFieldFormat);

Internal BASIC syntax

Function AmGetFieldFormatFromName(strTableName As String, strFieldName As String) As String

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓

	Available
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strTableName**: This parameter contains the SQL name of the table containing the field concerned by the operation.
- **strFieldName**: This parameter contains the SQL name of the field.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetFieldFromName()

This function creates a field object based on its name and returns the handle of the field object created.

API syntax

long AmGetFieldFromName(long hApiObject, char *strName);

Internal BASIC syntax

Function AmGetFieldFromName(hApiObject As Long, strName As String) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiObject**: This parameter contains a handle of a query, record, or table.
- **strName**: This parameter contains the field name.

AmGetFieldLabel()

This function returns, as a character string ("String" format), the label of a field identified by a handle.

API syntax

long AmGetFieldLabel(long hApiField, char *pstrBuffer, long lBuffer);

Internal BASIC syntax

Function AmGetFieldLabel(hApiField As Long) As String

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: This parameter contains a valid handle of the field whose label you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetFieldLabelFromName()

This function returns the label of a field from its SQL name.

API syntax

long AmGetFieldLabelFromName(long hApiCnxBase, char *strTableName, char *strFieldName, char *pFieldLabel, long lpFieldLabel);

Internal BASIC syntax

Function AmGetFieldLabelFromName(strTableName As String, strFieldName As String) As String

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strTableName**: This parameter contains the SQL name of the table containing the field concerned by the operation.
- **strFieldName**: This parameter contains the SQL name of the field.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetFieldLongValue()

This function returns the value of a field contained in the current object.

API syntax

long AmGetFieldLongValue(long hApiObject, long lFieldPos);

Internal BASIC syntax

Function AmGetFieldLongValue(hApiObject As Long, lFieldPos As Long) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiObject**: This parameter contains a handle of a query or record.
- **lFieldPos**: This parameter contains the number of the field inside the current object.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

Note:

If you use this function to recover the value of a field of date, time or date+time type, the long integer returned by the function represents the number of seconds since 01/01/1970 at 00:00:00.

AmGetFieldName()

This function returns the name of a field contained in the current object.

API syntax

long AmGetFieldName(long hApiObject, long lFieldPos, char *pstrBuffer, long lBuffer);

Internal BASIC syntax

Function AmGetFieldName(hApiObject As Long, lFieldPos As Long) As String

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiObject**: This parameter contains a handle of a query, record, or table.
- **lFieldPos**: This parameter contains the number of the field within the current object. E.g., the value "0" indicates the first field.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetFieldRights()

This function returns the user rights for a field in the current object. These rights are returned as a character string containing three characters, which specify the read/insert/update rights:

- "r": indicates the right to read data.
- "i": indicates the right to insert data.
- "u": indicates the right to update data.

For example, for a read-only field, the function returns the value "r ".

API syntax

long AmGetFieldRights(long hApiObject, long lFieldPos, char *pstrBuffer, long lBuffer);

Internal BASIC syntax

Function AmGetFieldRights(hApiObject As Long, lFieldPos As Long) As String

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiObject**: This parameter contains a handle of a query, record, or table.
- **lFieldPos**: This parameter contains the number of the field within the current object.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetFieldSize()

This function returns the size of a field.

API syntax

long AmGetFieldSize(long hApiField);

Internal BASIC syntax

Function AmGetFieldSize(hApiField As Long) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: This parameter contains a handle of the field whose size you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetFieldSqlName()

This function returns, as a character string ("String" format), the SQL name of a field identified by a handle.

API syntax

long AmGetFieldSqlName(long hApiField, char *pstrBuffer, long lBuffer);

Internal BASIC syntax

Function AmGetFieldSqlName(hApiField As Long) As String

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: This parameter contains a valid handle of the field whose SQL name you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetFieldStrValue()

This function returns the value of a field contained in the current object. This value is returned in string format.

Warning: When this function is used via the AssetCenter APIs, it expects two extra parameters **pszBuffer** and **lBuffer**, which define a string used as a buffer to store the recovered string and the size of this buffer respectively. The **pszBuffer** string must be formatted (filled with characters) and be of the size defined by **lBuffer**. The following portion of code is incorrect, the string used as a buffer is not sized:

```
Dim strBuffer as String
Dim lRec as Long
Dim lBuffer as Long
lBuffer=20
lRec=AmGetFieldStrValue(1, 0, strBuffer, lBuffer)
```

Here is the corrected portion of code:

```
Dim strBuffer as String
Dim lRec as Long
Dim lBuffer as Long
strBuffer=String(21, " ") ' The buffer is set to 21 characters (" ")
lBuffer=20
lRec=AmGetFieldStrValue(1, 0, strBuffer, lBuffer)
```

When you format a buffer string using the "String" function, do not use "0" as a padding character. Size the buffer before calling the **AmGetFieldStrValue** function, particularly if this function is in a loop and always uses the same string as a buffer.

API syntax

long AmGetFieldStrValue(long hApiObject, long lFieldPos, char *pstrBuffer, long lBuffer);

Internal BASIC syntax

Function AmGetFieldStrValue(hApiObject As Long, lFieldPos As Long) As String

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiObject**: This parameter contains a handle of a query or record.
- **lFieldPos**: This parameter contains the number of the field inside the current object.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetFieldType()

This function returns the type of a field.

API syntax

long AmGetFieldType(long hApiField);

Internal BASIC syntax

Function AmGetFieldType(hApiField As Long) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: This parameter contains a handle of the field whose type you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

Note:

The following table lists the values returned by the **AmGetFieldType** function for each type of field:

Valeurs retournées	Type de champ correspondant
0	Non défini
1	Byte
2	Short
3	Long
4	Float
5	Double
6	String
7	Time stamp
8	Bin
9	Blob
10	Date
11	Time
12	Memo

AmGetFieldUserType()

This function returns the "UserType" of a field (cf. **database.txt** file) identified by a handle, in the form of a long integer. For a field, the valid return values are summarized below:

Stored value	Plain-text value
0	Default
1	Number
2	Yes/ No
3	Money
4	Date
5	Date+Time
7	System itemized list
8	Custom itemized list
10	Percentage

Stored value	Plain-text value
11	Time span
12	Table or field SQL name

For a link, the valid return values are summarized below:

Stored value	Plain-text value
0	Normal
1	Comment
2	Image
3	History
4	Feature value

Up until version 4.0.0, the function always returned 0 for a link. From AssetCenter version 4.1.0 onwards, the function returns one of the following values for a link:

- 0: Normal
- 1: Comments
- 2: Image
- 3: History
- 5: Script

API syntax

long AmGetFieldUserType(long hApiField);

Internal BASIC syntax

Function AmGetFieldUserType(hApiField As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓

	Available
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: This parameter contains a valid handle of the field whose "UserType" you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetForeignKey()

Recovers the handle of the foreign key of a link, itself identified by its handle.

API syntax

long AmGetForeignKey(long hApiField);

Internal BASIC syntax

Function AmGetForeignKey(hApiField As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: Handle of the link concerned by the operation.

AmGetIndex()

This function creates an index object from a handle of a query, record, or a table and returns the handle of the index object created.

API syntax

long AmGetIndex(long hApiTable, long lPos);

Internal BASIC syntax

Function AmGetIndex(hApiTable As Long, lPos As Long) As Long

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiTable**: This parameter contains a handle of a table.
- **lPos**: This parameter contains the position of the index in the table.

AmGetIndexCount()

This function returns the number of indexes contained in the table specified in the **hApiTable** parameter.

API syntax

long AmGetIndexCount(long hApiTable);

Internal BASIC syntax

Function AmGetIndexCount(hApiTable As Long) As Long

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiTable**: This parameter contains a handle of a table.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetIndexField()

This function returns a handle on a field identified by its position in the index (the lpos th field of the index).

API syntax

long AmGetIndexField(long hApiIndex, long lPos);

Internal BASIC syntax

Function AmGetIndexField(hApiIndex As Long, lPos As Long) As Long

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiIndex**: This parameter contains a valid handle on the index concerned by the operation.
- **lPos**: This parameter contains the position of the field in the index.

AmGetIndexFieldCount()

This function returns the number of fields making up an index.

API syntax

long AmGetIndexFieldCount(long hApiIndex);

Internal BASIC syntax

Function AmGetIndexFieldCount(hApiIndex As Long) As Long

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiIndex**: This parameter contains a valid handle on the index concerned by the operation.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetIndexFlags()

This function returns the parameters of an index.

API syntax

long AmGetIndexFlags(long hApiIndex);

Internal BASIC syntax

Function AmGetIndexFlags(hApiIndex As Long) As Long

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiIndex**: This parameter contains a valid handle on the index concerned by the operation.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

The value returned by the function results from a logical combination (OR) of the following values:

- 1: The index authorized non unique records,
- 2: The index authorizes the null value,
- 4: The index is not case sensitive.

Thus, if the function returns 3, you can deduce that the index accepts non unique records and the null value (1 OR 2 = 3).

AmGetIndexName()

This function returns the name of an index.

API syntax

long AmGetIndexName(long hApiIndex, char *pstrBuffer, long lBuffer);

Internal BASIC syntax

Function AmGetIndexName(hApiIndex As Long) As String

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiIndex**: This parameter contains a valid handle on the index whose name you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetLink()

This function creates a link object from the handle of a table and returns the handle of the link object created.

API syntax

long AmGetLink(long hApiTable, long lPos);

Internal BASIC syntax

Function AmGetLink(hApiTable As Long, lPos As Long) As Long

Field of application

Version: 3.02

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiTable**: This parameter contains a handle of a table.
- **lPos**: This parameter contains the position of the link (its index) inside the object.

AmGetLinkCardinality()

This function returns the cardinality of a link.

API syntax

long AmGetLinkCardinality(long hApiField);

Internal BASIC syntax

Function AmGetLinkCardinality(hApiField As Long) As Long

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: This parameter contains a handle of the link whose cardinality you want to know.

Output parameters

- 1: The cardinality of the link is 1-1.
- 2: The cardinality of the link is 1-n.

AmGetLinkCount()

This function returns the number of links contained in the current table.

API syntax

long AmGetLinkCount(long hApiTable);

Internal BASIC syntax

Function AmGetLinkCount(hApiTable As Long) As Long

Field of application

Version: 3.02

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiTable**: This parameter contains a handle of a valid table.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetLinkDstField()

This function returns the field (foreign key) to which the link defined by the **hApiField** parameter points.

API syntax

long AmGetLinkDstField(long hApiField);

Internal BASIC syntax

Function AmGetLinkDstField(hApiField As Long) As Long

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: This parameter contains a handle of the link concerned by the operation.

AmGetLinkFeatureValue()

Returns the value of a "Link" type feature.

API syntax

long AmGetLinkFeatureValue(long hApiObject, long lFieldPos, long lRecordId);

Internal BASIC syntax

Function AmGetLinkFeatureValue(hApiObject As Long, lFieldPos As Long, lRecordId As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiObject**: This parameter contains a handle of a query or record.
- **lFieldPos**: This parameter contains the position of the field inside the current object.
- **lRecordId**: This parameter contains the identifier of the record whose feature value you want to recover.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim q as String
q = "Select fv_link, lEmplDeptId From amEmplDept Where lEmplDeptId = " &
[lEmplDeptId]
Dim hq as Long
hq = amQueryCreate()
Dim lErr as Long
lErr = amQueryGet(hq, q)
Dim lId as Long
```

```
lId = amGetFieldLongValue(hq, 1)
amMsgBox("str: " & amGetFieldStrValue(hq, 0))
amMsgBox("int: " &
amGetFieldLongValue(hq,0))
amMsgBox("lnk: " & amGetLinkFeatureValue(hq,0,lId))
```

AmGetLinkFromName()

This function creates a link object from a name and returns the handle of the object created.

API syntax

long AmGetLinkFromName(long hApiTable, char *strName);

Internal BASIC syntax

Function AmGetLinkFromName(hApiTable As Long, strName As String) As Long

Field of application

Version: 3.02

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiTable**: This parameter contains a handle of a table.
- **strName**: This parameter contains the SQL name of the link.

AmGetLinkType()

This function returns the type of a link.

API syntax

long AmGetLinkType(long hApiField);

Internal BASIC syntax

Function AmGetLinkType(hApiField As Long) As Long

Field of application

Version: 3.02

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: This parameter contains a handle of the link whose type you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetMainField()

This function creates a field object corresponding to the main field in a given table. It returns a handle of the field thus created.

API syntax

long AmGetMainField(long hApiTable);

Internal BASIC syntax

Function AmGetMainField(hApiTable As Long) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiTable**: This parameter contains a handle of the table whose main field you want to know.

AmGetMemoField()

This function creates a field object that correspondes to a Memo-type field of a given table. It returns a handle on the field thus created.

API syntax

long AmGetMemoField(long hApiTable);

Internal BASIC syntax

Function AmGetMemoField(hApiTable As Long) As Long

Field of application

Version: 4.1.0

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiTable**: This parameter contains a handle on the table for which we are searching the Memo field.

AmGetNextAssetPin()

The AmGetNextAssetPin API finds the next available pin on a device (lAssetId). Its sequence number sorts the pins. Depending on the port direction

(iPinPortDir), the available pins are sorted in ascending (iPinPortDir = 0) or descending (iPinPortDir = 1) order.

API syntax

long AmGetNextAssetPin(long hApiCnxBase, long lAssetId, long bPinPortDir, long iPrevPinSeq);

Internal BASIC syntax

Function AmGetNextAssetPin(lAssetId As Long, bPinPortDir As Long, iPrevPinSeq As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lAssetId**: This parameter is the device ID.
- **bPinPortDir**: This parameter is the direction to search.
  - 0=ascending
  - 1=descending
- **iPrevPinSeq**

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetNextAssetPort()

The AmGetNextAssetPort API finds the next available port on a device (lAssetId) providing a given service (lDutyId) or no service at all. The status of the port must be "Available". Boolean flags are used to signify if the user side (bCheckUser) and/or the host side (bCheckHost) of the port should be checked. The function compares the user value (bUserAvail) and /or the hosts value (bHostAvail) if the corresponding Boolean flag is true. The ports are sorted by their sequence number. Depending on the port direction (bPinPortDir), the available ports are sorted in ascending (bPinPortDir = 0) or descending (bPinPortDir = 1) order.

API syntax

long AmGetNextAssetPort(long hApiCnxBase, long lAssetId, long lCabCnxTypeId, long lDutyId, long bCheckUser, long bCheckHost, long bUserAvail, long bHostAvail, long bPinPortDir, long iPrevPortSeq);

Internal BASIC syntax

Function AmGetNextAssetPort(lAssetId As Long, lCabCnxTypeId As Long, lDutyId As Long, bCheckUser As Long, bCheckHost As Long, bUserAvail As Long, bHostAvail As Long, bPinPortDir As Long, iPrevPortSeq As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lAssetId**: This parameter defines the device ID to search.
- **lCabCnxTypeId**: This parameter defines the cable connection type for the port.
- **lDutyId**: This parameter is the duty of the port.
- **bCheckUser**: This parameter is a flag to check the user side.
- **bCheckHost**: This parameter is a flag to check the host side.
- **bUserAvail**: This parameter defines the user side availability state to check.
- **bHostAvail**: This parameter defines the host side availability state to check.
- **bPinPortDir**: This parameter defines the pin direction to check.
  - 0=ascending
  - 1=descending
- **iPrevPortSeq**

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetNextCableBundle()

The AmGetNextCableBundle API finds the next available bundle on a cable (lCableId) providing a given service (lDutyId) or no service at all. The status of the bundle must be "Available". Boolean flags are used to signify if the user side (bCheckUser) and/or the host side (bCheckHost) of the bundle should be checked. The function compares the user value (bUserAvail) and/ or the host value (bHostAvail) if the corresponding Boolean flag is true.

API syntax

long AmGetNextCableBundle(long hApiCnxBase, long lCableId, long lDutyId, long bCheckUser, long bCheckHost, long bUserAvail, long bHostAvail);

Internal BASIC syntax

Function AmGetNextCableBundle(lCableId As Long, lDutyId As Long, bCheckUser As Long, bCheckHost As Long, bUserAvail As Long, bHostAvail As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lCableId**: This parameter is the ID of the cable to check.

- **lDutyId**: This parameter defines the duty to locate.
- **bCheckUser**: This parameter states to check the user side connection of the bundle.
- **bCheckHost**: This parameter states to check the host side connection of the bundle.
- **bUserAvail**: This parameter defines the user side connection state to locate.
- **bHostAvail**: This parameter defines the host side connection state to locate.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetNextCablePair()

The AmGetNextCablePair API finds the next available cable pair in a cable (lCableId) of a given type (lPairType). The pairs are sorted by cable's pair ID.

API syntax

long AmGetNextCablePair(long hApiCnxBase, long lCableId, long lPairTypeId, long iStartPairSeq);

Internal BASIC syntax

Function AmGetNextCablePair(lCableId As Long, lPairTypeId As Long, iStartPairSeq As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lCableId**: This parameter is the ID of the cable to search.
- **lPairTypeId**: This parameter defines the cable pair type to locate.
- **iStartPairSeq**

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetNTDomains()

This function enables you to get the domain of the user connected to the database.

API syntax

long AmGetNTDomains(char *pstrDomains, long lDomains);

Internal BASIC syntax

Function AmGetNTDomains() As String

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetNTMachinesInDomain()

This function enables you to get the list of computers in a domain as a column (computer names separated by commas). If the domain is empty, the function returns ERR_CANCEL(2), but the execution is not interrupted.

API syntax

long AmGetNTMachinesInDomain(char *strDomain, char *pstrMachines, long lMachines, long bUseDC);

Internal BASIC syntax

Function AmGetNTMachinesInDomain(strDomain As String, bUseDC As Long) As String

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strDomain**: This parameter contains the name of the domain to explore.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetNTUsersInDomain()

This function enables you to get the list of users of a domain. The list is returned as two columns (login,fullname). '|' is used as column separator, ',' as line separator.

API syntax

long AmGetNTUsersInDomain(char *strDomain, char *pstrUsers, long lUsers);

Internal BASIC syntax

Function AmGetNTUsersInDomain(strDomain As String) As String

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strDomain**: This parameter contains the name of the domain to explore.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetPOLinePrice()

This function enables you to calculate the price of an order line.

API syntax

double AmGetPOLinePrice(long hApiCnxBase, long lPOrdLineId);

Internal BASIC syntax

Function AmGetPOLinePrice(lPOrdLineId As Long) As Double

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lPOrdLineId**: This parameter contains the identifier of the order line.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetPOLinePriceCur()

This function enables you to find the currency code for the order line

API syntax

long AmGetPOLinePriceCur(long hApiCnxBase, long lPOrdLineId, char *pstrPrice, long lPrice);

Internal BASIC syntax

Function AmGetPOLinePriceCur(lPOrdLineId As Long) As String

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lPOrdLineId**: This parameter contains the identifier of the order line.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetPOLineReference()

This function enables you to get the catalog reference description corresponding to the purchase order line.

API syntax

long AmGetPOLineReference(long hApiCnxBase, long lPOrdLineId, char *pstrRef, long lRef);

Internal BASIC syntax

Function AmGetPOLineReference(lPOrdLineId As Long) As String

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lPOrdLineId**: This parameter contains the identifier of the order line.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetRecordFromMainId()

This function returns the ID number of a record identified by a value of the primary key of the table containing this record.

API syntax

long AmGetRecordFromMainId(long hApiCnxBase, char *strTable, long lId);

Internal BASIC syntax

Function AmGetRecordFromMainId(strTable As String, lId As Long) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **strTable**: This parameter contains the SQL name of the table containing the record concerned by the operation.

- **lId**: This parameter contains the value of the primary key of the table for this records.

Notes

This function systematically returns a record handle, except when the table is not found. If no record is found in the specified table, an error is raised at each execution of a function using the handle returned by this function.

AmGetRecordHandle()

This function returns the handle of a record that is the current result of a query identified by its handle. This record can be used to write in the database. This function only works if the query contains the primary key of the record.

API syntax

long AmGetRecordHandle(long hApiQuery);

Internal BASIC syntax

Function AmGetRecordHandle(hApiQuery As Long) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **hApiQuery**: This parameter contains a valid handle of a query object.

AmGetRecordId()

This function returns the ID number of a record identified by its handle. In the case of a record being inserted, this value is 0.

API syntax

long AmGetRecordId(long hApiRecord);

Internal BASIC syntax

Function AmGetRecordId(hApiRecord As Long) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **hApiRecord**: This parameter contains a valid handle of the record whose ID number you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetRelDstField()

This function returns a handle on the target field of a link.

API syntax

long AmGetRelDstField(long hApiField);

Internal BASIC syntax

Function AmGetRelDstField(hApiField As Long) As Long

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: This parameter contains a valid handle on the link concerned by the operation.

AmGetRelSrcField()

This function returns a handle on the source field of a link.

API syntax

long AmGetRelSrcField(long hApiField);

Internal BASIC syntax

Function AmGetRelSrcField(hApiField As Long) As Long

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: This parameter contains a valid handle on the link concerned by the operation.

AmGetRelTable()

This function returns a handle on the relation table of an N-N link.

API syntax

long AmGetRelTable(long hApiField);

Internal BASIC syntax

Function AmGetRelTable(hApiField As Long) As Long

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: This parameter contains a valid handle on the link concerned by the operation.

Output parameters

In case of error, this function returns a non-valid handle (zero).

AmGetReverseLink()

This function returns the handle of the reverse link specified by the handle contained in the **hApiField** parameter.

API syntax

long AmGetReverseLink(long hApiField);

Internal BASIC syntax

Function AmGetReverseLink(hApiField As Long) As Long

Field of application

Version: 3.02

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: This parameter contains a handle of the link whose reverse link you want to know.

AmGetSelfFromMainId()

Returns the description string for a record in a given table.

API syntax

long AmGetSelfFromMainId(long hApiCnxBase, char *strTableName, long lId, char *pstrRecordDesc, long lRecordDesc);

Internal BASIC syntax

Function AmGetSelfFromMainId(strTableName As String, lId As Long) As String

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strTableName**: This parameter contains the SQL name of the table containing record concerned by the operation.
- **lId**: This parameter contains the ID number concerned by the operation.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetSourceTable()

Returns the handle of the source table of the link indicted in the **hApiField** parameter.

API syntax

long AmGetSourceTable(long hApiField);

Internal BASIC syntax

Function AmGetSourceTable(hApiField As Long) As Long

Field of application

Version: 3.02

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: This parameter contains a valid handle of the link whose source table you want to know.

Output parameters

In case of error, this function returns a non-valid handle (zero).

AmGetTable()

This function returns the handle of a table identified by its position in the current connection.

API syntax

long AmGetTable(long hApiCnxBase, long lPos);

Internal BASIC syntax

Function AmGetTable(lPos As Long) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lPos**: This parameter contains the position of the table in the current connection. Its values are comprised between "0" and **AmGetTableCount**.

Output parameters

In case of error, this function returns a non-valid handle (zero).

AmGetTableCount()

This function returns the number of tables in the database concerned by the currency connection.

API syntax

long AmGetTableCount(long hApiCnxBase);

Internal BASIC syntax

Function AmGetTableCount() As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetTableDescription()

This function returns, as a character string ("String" format), the long description of a table identified by a handle.

API syntax

long AmGetTableDescription(long hApiTable, char *pstrDesc, long lDesc);

Internal BASIC syntax

Function AmGetTableDescription(hApiTable As Long) As String

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiTable:** This parameter contains a valid handle of the table whose long description you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetTableFromName()

This function returns the handle of a table identified by its SQL name in the current connection.

API syntax

long AmGetTableFromName(long hApiCnxBase, char *strName);

Internal BASIC syntax

Function AmGetTableFromName(strName As String) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strName:** This parameter contains the SQL name of the table whose handle you want to recover.

Output parameters

In case of error, this function returns a non-valid handle (zero).

AmGetTableLabel()

This function returns, as a character string ("String" format), the label of a table identified by a handle.

API syntax

long AmGetTableLabel(long hApiTable, char *pstrLabel, long lLabel);

Internal BASIC syntax

Function AmGetTableLabel(hApiTable As Long) As String

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiTable**: This parameter contains a valid handle of the table whose label you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetTableName()

Returns the SQL name of a table as a character string.

API syntax

long AmGetTableName(long hApiTable, char *pstrBuffer, long lBuffer);

Internal BASIC syntax

Function AmGetTableName(hApiTable As Long) As String

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiTable**: Valid handle of the table whose name you want to recover.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetTableRights()

This function returns, as a character string ("String" format), the users rights for a table given by a handle. The returned string consists of a maximum of two characters that indicate the status of creation and deletion rights:

- "c" indicates that the user has creation rights for the table.
- "d" indicates that the user has deletion rights on the table.

Thus, for example:

- " c" means that the user has creation rights for the table only.
- "cd" means that the user has both creation and deletion rights for the table.

API syntax

long AmGetTableRights(long hApiTable, char *pstrBuffer, long lBuffer);

Internal BASIC syntax

Function AmGetTableRights(hApiTable As Long) As String

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓

	Available
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiTable**: This parameter contains a valid handle of the table for which you want to know the user rights.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetTableSqlName()

This function returns, as a character string ("String" format), the SQL name of a table identified by a handle.

API syntax

long AmGetTableSqlName(long hApiTable, char *pstrBuffer, long lBuffer);

Internal BASIC syntax

Function AmGetTableSqlName(hApiTable As Long) As String

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiTable**: This parameter contains a valid handle of the table whose SQL name you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetTargetTable()

Returns the SQL name of the target table of a link.

API syntax

long AmGetTargetTable(long hApiField);

Internal BASIC syntax

Function AmGetTargetTable(hApiField As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: Handle of the link concerned by the operation.

Output parameters

In case of error, this function returns a non-valid handle (zero).

AmGetTrace()

The AmGetTrace API gets the trace between two nodes (lUserId, lHostId) in the cable link table. The trace direction (lTraceDir) identifies if the trace should be user-to-host (lTraceDir = 1) or host-to-user (lTraceDir = 0). The trace type (lTraceType) indicates if the trace is a connection (lTraceType = 1) or a disconnection (lTraceType = 2). The full trace indicator (bFullTrace) identifies if the trace include only modified nodes (bFullTrace = 0) or the entire trace (bFullTrace = 1).

API syntax

long AmGetTrace(long hApiCnxBase, long lUserId, long lHostId, long iTraceDir, long iTraceType, long bFullTrace, char *pstrTrace, long lTrace);

Internal BASIC syntax

Function AmGetTrace(lUserId As Long, lHostId As Long, iTraceDir As Long, iTraceType As Long, bFullTrace As Long) As String

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lUserId**: This parameter defines the starting connection link ID.
- **lHostId**: This parameter defines the ending connection link ID.
- **iTraceDir**: This parameter specifies the direction of the connection.
  - 0=host to user
  - 1=user to host
- **iTraceType**: This parameter defines the connection type.
  - 1=connection
  - 2=disconnection
- **bFullTrace**: This parameter specifies to ignore the partial trace and return the whole trace string.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetTraceFromHist()

The AmGetTraceFromHist API is for calculating a string from Trace History using Trace Operations to show new connectivity versus existing connectivity.

API syntax

long AmGetTraceFromHist(long hApiCnxBase, long lProjTraceOutId, long iTraceDir, char *strDelimiter, char *pstrTraceint, long lTraceint, long bUpdateFlag);

Internal BASIC syntax

Function AmGetTraceFromHist(lProjTraceOutId As Long, iTraceDir As Long, strDelimiter As String, bUpdateFlag As Long) As String

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lProjTraceOutId**: This parameter defines the project trace ID.
- **iTraceDir**: This parameter specifies the direction of the connection.
  - 0=host to user
  - 1=user to host
- **strDelimiter**: This parameter is the string delimiter to show existing connects and disconnects.
- **bUpdateFlag**: This parameter is an optional parameter to AmGetTraceHist API to update the amCabTraceOut.TraceString.
  - 0=false
  - 1=true

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmGetTypedLinkField()

Returns a handle of the field whose value is the SQL name of the target table of the typed link indicated in the **hApiField** parameter.

API syntax

long AmGetTypedLinkField(long hApiField);

Internal BASIC syntax

Function AmGetTypedLinkField(hApiField As Long) As Long

Field of application

Version: 3.02

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: This parameter contains a valid handle of the typed link at the origin of the operation.

AmGetVersion()

This function returns the build number of AssetCenter in the form of a character string.

API syntax

long AmGetVersion(char *pstrBuf, long lBuf);

Internal BASIC syntax

Function AmGetVersion() As String

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmHasAdminPrivilege()

This function returns "TRUE" (value other that 0) if the connected user has administration rights.

API syntax

long AmHasAdminPrivilege(long hApiCnxBase);

Internal BASIC syntax

Function AmHasAdminPrivilege() As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmHasRelTable()

This function enables you to test whether a link has a relation table or not.

API syntax

long AmHasRelTable(long hApiField);

Internal BASIC syntax

Function AmHasRelTable(hApiField As Long) As Long

Field of application

Version: 3.5

	Available
AssetCenter API	✓

	Available
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: This parameter contains a valid handle on the link concerned by the operation.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmImportDocument()

This function creates and imports a document from a file.

API syntax

long AmImportDocument(long hApiCnxBase, long lDocObjId, char *strTableName, char *strFileName, char *strCategory, char *strDesignation);

Internal BASIC syntax

Function AmImportDocument(lDocObjId As Long, strTableName As String, strFileName As String, strCategory As String, strDesignation As String) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lDocObjId**: This parameter contains the value that will be stored in the lDocObjId field of the amDocument table.
- **strTableName**: This parameter contains the value that will be stored in the DocObjTable field of the amDocument table. In practice, it is the SQL name of the table containing the record to which the document is attached.
- **strFileName**: This parameter contains the name of the file to import.
- **strCategory**: This parameter contains the category of the document, as is appears in AssetCenter.
- **strDesignation**: This parameter contains the name of the document, as it appears in AssetCenter.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmIncrementLogLevel()

This function displays the **strMsg** message in a history window and creates a node in the final page of a wizard.

All the following messages appear in this node.

Internal BASIC syntax

Function AmIncrementLogLevel(strMsg As String, iType As Long) As Long

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **strMsg**: This parameter contains the text of the message to be displayed.
- **iType**: This parameter defines the icon associated with the message. The possible values are "1" for an error, "2" for a warning, and "4" for information.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmInsertRecord()

This function inserts a record previously created in the database. Only those records created using the **AmCreateRecord** function can be inserted in the database. Records accessed using a query cannot be inserted.

API syntax

long AmInsertRecord(long hApiRecord);

Internal BASIC syntax

Function AmInsertRecord(hApiRecord As Long) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **hApiRecord**: This parameter contains a handle of the record you want to insert in the database.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmInstantiateReqLine()

This function enables you to directly instantiate a given request line.

API syntax

long AmInstantiateReqLine(long hApiCnxBase, long lRequestLineId, long bFinal, long lPOrderLineId, float fQty);

Internal BASIC syntax

Function AmInstantiateReqLine(lRequestLineId As Long, bFinal As Long, lPOrderLineId As Long, fQty As Single) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lRequestLineId**: This parameter contains the identifier of the request line.
- **bFinal**This parameter enables you to specify whether you want to finalize the assignment.

- **lPOrderLineId**: This parameter contains the identifier of the order line.
- **fQty**: This parameter contains quantity to instantiate.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Notes

The function enables you to create requested elements without going through the procurement cycle. If bFinal = FALSE, then the asset will be created with the status Awaiting receipt.

AmInstantiateRequest()

This function enables you to directly instantiate the full contents of a given request.

API syntax

long AmInstantiateRequest(long hApiCnxBase, long lRequestId, long lMulFactor);

Internal BASIC syntax

Function AmInstantiateRequest(lRequestId As Long, lMulFactor As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lRequestId**: This parameter contains the identifier of the request.
- **lMulFactor**: This parameter enables you to specify the number of instantiations to perform.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmIsConnected()

This function tests whether the current connection is valid.

API syntax

long AmIsConnected(long hApiCnxBase);

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	
Wizard script	

	Available
FINISH.DO script of a wizard	

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmIsFieldForeignKey()

This function enables you to determine whether a field is an foreign key in the database.

API syntax

long AmIsFieldForeignKey(long hApiField);

Internal BASIC syntax

Function AmIsFieldForeignKey(hApiField As Long) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓

	Available
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: This parameter contains a handle of the field to be identified.

Output parameters

- 1: The field is a foreign key.
- 0: The field is not a foreign key.

AmIsFieldIndexed()

This function enables you to determine whether a field is indexed or not.

API syntax

long AmIsFieldIndexed(long hApiField);

Internal BASIC syntax

Function AmIsFieldIndexed(hApiField As Long) As Long

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: This parameter contains a handle of the field to be identified.

Output parameters

- 1: The field is indexed.
- 0: The field is not indexed.

AmIsFieldPrimaryKey()

This function enables you to determine whether a field is an primary key in the database.

API syntax

long AmIsFieldPrimaryKey(long hApiField);

Internal BASIC syntax

Function AmIsFieldPrimaryKey(hApiField As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: This parameter contains a handle of the field to be identified.

Output parameters

- 1: The field is a primary key.
- 0: The field is not a primary key.

AmIsLink()

Determines whether the object identified by its handle is a link or a field.

API syntax

long AmIsLink(long hApiField);

Internal BASIC syntax

Function AmIsLink(hApiField As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: Handle of the object concerned by the operation.

Output parameters

- 1: The object is a link.
- 0: The object is a field.

AmIsTypedLink()

Determines if the object identified by its handle is a typed link or not.

API syntax

long AmIsTypedLink(long hApiField);

Internal BASIC syntax

Function AmIsTypedLink(hApiField As Long) As Long

Field of application

Version: 3.02

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiField**: Handle of the object concerned by the operation.

Output parameters

- 1: The object is a typed link.
- 0: The object is not a typed link.

AmLastError()

This function returns the last error code generated by the last function executed in the context of the corresponding connection.

API syntax

long AmLastError(long hApiCnxBase);

Internal BASIC syntax

Function AmLastError() As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmLastErrorMsg()

This function returns the last error message occurred in the current connection.

API syntax

long AmLastErrorMsg(long hApiCnxBase, char *pstrBuffer, long lBuffer);

Internal BASIC syntax

Function AmLastErrorMsg() As String

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmListToString()

This function converts the result of a character string obtained via the **AmDbGetList** function to a character string that can be displayed in the same way as the **AmDbGetString** function.

API syntax

long AmListToString(char *return, long lreturn, char *strSource, char *strColSep, char *strLineSep, char *strIdSep);

Internal BASIC syntax

Function AmListToString(strSource As String, strColSep As String, strLineSep As String, strIdSep As String) As String

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strSource**: This parameter contains the character string to be converted.
- **strColSep**: This parameter contains the character used as column separator in the string to be converted.
- **strLineSep**: This parameter contains the character used as line separator in the string to be converted.
- **strIdSep**: This parameter contains the character used as identifier separator in the string to be converted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmLog()

This function displays the **strMessage** message in a history window.

Internal BASIC syntax

Function AmLog(strMessage As String, iLogType As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	

	Available
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **strMessage**: This parameter contains the text of the message to be displayed.
- **iLogType**: This parameter defines the icon associated with the message. The possible values are "1" for an error, "2" for a warning, and "4" for information.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Example

```
AmLog("This is a message")
```

AmLoginId()

This function returns the identifier of the connected user.

API syntax

long AmLoginId(long hApiCnxBase);

Internal BASIC syntax

Function AmLoginId() As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following example defines the identifier of the connected user as the default value for a database field:

```
RetVal=AmLoginId()
```

AmLoginName()

This function returns the login name of the connected user.

API syntax

long AmLoginName(long hApiCnxBase, char *return, long lreturn);

Internal BASIC syntax

Function AmLoginName() As String

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following example defines the login name of the connected user as the default value for a database field:

```
RetVal=AmLoginName()
```

AmMapSubReqLineAgent()

This function enables you to establish the possible links between the sub-lines of a request line and those of an order line.

API syntax

long AmMapSubReqLineAgent(long hApiCnxBase, long lRequestLineId, long lPorderLineId);

Internal BASIC syntax

Function AmMapSubReqLineAgent(lRequestLineId As Long, lPorderLineId As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lRequestLineId**: This parameter contains the identifier of the request line.
- **lPorderLineId**: This parameter contains the identifier of the request line.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmMoveCable()

The AmMoveCable API moves a cable (lCableId) from its current location to a given destination location (lToLoc). If the project (lProjectId) and work

order (lWorkOrderId) have values, the cable is added to the project and work order with the comment contained in the given comment (strComment). This comment describes the action that will be performed on the cable (i.e. "Move cable from here to there").

API syntax

long AmMoveCable(long hApiCnxBase, long lCableId, long lToLocId, long lProjectId, long lWorkOrderId, char *strComment);

Internal BASIC syntax

Function AmMoveCable(lCableId As Long, lToLocId As Long, lProjectId As Long, lWorkOrderId As Long, strComment As String) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lCableId**: This parameter is the ID of the cable to move.
- **lToLocId**: This parameter defines the cable ID to move.
- **lProjectId**: This parameter is the project ID.
- **lWorkOrderId**: This parameter defines the work order ID.
- **strComment**: This parameter is the comment that will be used on the work order.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmMoveDevice()

The AmMoveDevice API moves a device (lAssetId) from its current location to a given destination location (lToLoc). If the project (lProjectId) and work order (lWorkOrderId) have values, the device is added to the project and work order with the given comment (strComment). This comment describes the action that will be performed on the device (i.e. "Move device from here to there").

API syntax

long AmMoveDevice(long hApiCnxBase, long lDeviceId, long lToLocationId, long lProjectId, long lWorkOrderId, char *strComment);

Internal BASIC syntax

Function AmMoveDevice(lDeviceId As Long, lToLocationId As Long, lProjectId As Long, lWorkOrderId As Long, strComment As String) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lDeviceId**: This parameter defines the device ID that will be moved.
- **lToLocationId**: This parameter defines the device's new location.
- **lProjectId**: This parameter is the project ID.
- **lWorkOrderId**: This parameter defines the work order ID.
- **strComment**: This parameter is the comment that will be used on the work order.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmMsgBox()

This function displays a dialog box containing a message.

Internal BASIC syntax

Function AmMsgBox(strMessage As String, lMode As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strMessage**: This parameter contains the message displayed in the dialog box.
- **lMode**: This parameter contains the displayed dialog box type (O for a simple dialog box with an OK button, 1 for a dialog box with OK and Cancel, 2 for a dialog box with just Cancel).

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Example

```
AmMsgBox("Move carried out")
```

AmOpenConnection()

Creates a session from an AC database name. **strDataSource** should be a valid AssetCenter data source name (these AC database connections are listed in the login box of AssetCenter).

You can open several connections, in the same database or on different databases.

API syntax

long AmOpenConnection(char *strDataSource, char *strUser, char *strPwd);

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	

Input parameters

- **strDataSource**: Name of the data source.
- **strUser**: User name for the connection.
- **strPwd**: Password of the specified user.

AmOpenScreen()

This function enables you to open a screen in AssetCenter.

Internal BASIC syntax

Function AmOpenScreen(strScreenId As String, strContext As String, strFilter As String, iMode As Long, strBindField As String) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strScreenId**: This parameter contains the SQL name of the system or user screen you want to open (in this order of priority).
- **strContext**: This optional parameter contains the list of identifiers of the records selected in the list on opening the screen.
- **strFilter**: This parameter contains an AQL filter applied on the list on opening the screen.
- **iMode**: This parameter contains the mode in which the screen is opened: consultation, edit, etc. The possible values are: 0 (No action in progress), 1 (No action in progress), 2 (Modification in progress), 3 (Creation in progress), 4 (Duplication in progress), 5 (Addition in progress), 6 (Selection in progress).
- **strBindField**: This parameter enables you to open a screen with a filter and a mode for opening a linked window. It uses the SQL name of the source field or the value CurrentSrcChoice to use the current context.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmPagePath()

This function returns a string containing the execution path of the wizard, i.e. the list of pages browsed. Backward jumps are ignored.

Internal BASIC syntax

Function AmPagePath() As String

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmProgress()

This function displays, in the final page of a wizard, a progress indicator representing a percentage.

Internal BASIC syntax

Function AmProgress(iProgress As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **iProgress**: This parameter contains the percentage of completion (between 0 and 100) used to define size of the progress indicator.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Example

```
AmProgress(85)
```

This function displays a progress indicator representing 85% completion.

AmQueryCreate()

This function creates a query object in the current connection. This object can then be used to send AQL statements to the database server.

API syntax

long AmQueryCreate(long hApiCnxBase);

Internal BASIC syntax

Function AmQueryCreate() As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓

	Available
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

AmQueryExec()

This function executes an AQL query. It returns the first result of the query. The next result can be obtained via the **AmQueryNext** function.

When the query sent by this function returns a "Memo" type field the result is limited to 255 characters.

API syntax

long AmQueryExec(long hApiQuery, char *strQueryCommand);

Internal BASIC syntax

Function AmQueryExec(hApiQuery As Long, strQueryCommand As String) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiQuery**: This parameter contains a valid handle of the object to which the AQL statements are sent.
- **strQueryCommand**: This parameter contains the body of the AQL query as a string.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmQueryGet()

This function executes an AQL query without a cursor (one single result). It only returns one single line of results.

API syntax

long AmQueryGet(long hApiQuery, char *strQueryCommand);

Internal BASIC syntax

Function AmQueryGet(hApiQuery As Long, strQueryCommand As String) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓

	Available
FINISH.DO script of a wizard	✓

Input parameters

- **hApiQuery**: This parameter contains a valid handle of the object to which the AQL statements are sent.
- **strQueryCommand**: This parameter contains the body of the AQL query as a string.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmQueryNext()

This function returns the result of a query executed beforehand using the **AmQueryExec** function.

API syntax

long AmQueryNext(long hApiQuery);

Internal BASIC syntax

Function AmQueryNext(hApiQuery As Long) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓

	Available
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiQuery**: This parameter contains a valid handle of the object to which the AQL statements are sent.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmQuerySetAddMainField()

This function enables you to send a query in a mode where the main field of the table is automatically added to the list of fields to be returned. This type of query never returns a null identifier record.

API syntax

long AmQuerySetAddMainField(long hApiQuery, long bAddMainField);

Internal BASIC syntax

Function AmQuerySetAddMainField(hApiQuery As Long, bAddMainField As Long) As Long

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiQuery**: This parameter contains a valid handle on a query object.
- **bAddMainField**: This parameter can have one of two values:
  - True: The main field of the table is added,
  - False: The main field of the table is not added.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmQuerySetFullMemo()

By default, when executing the **AmQueryExec** function, the query truncates Memo type fields to 254 characters. This function sends the query in a mode where Memo fields are recovered in full.

API syntax

long AmQuerySetFullMemo(long hApiQuery, long bFullMemo);

Internal BASIC syntax

Function AmQuerySetFullMemo(hApiQuery As Long, bFullMemo As Long) As Long

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiQuery**: This parameter contains a valid handle on a query object.
- **bFullMemo**: This parameter can have one of two values:
  - True: The query returns the Memo field in full,
  - False: The query truncates Memo fields to 254 characters.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmQueryStartTable()

This function returns a handle of the table concerned by a query identified by its handle.

API syntax

long AmQueryStartTable(long hApiQuery);

Internal BASIC syntax

Function AmQueryStartTable(hApiQuery As Long) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiQuery**: This parameter contains a valid handle of a query object.

Output parameters

In case of error, this function returns a non-valid handle (zero).

AmQueryStop()

This function interrupts the execution of a query identified by its handle. This query must have been launched beforehand using the **AmQueryExec** function.

API syntax

long AmQueryStop(long hApiQuery);

Internal BASIC syntax

Function AmQueryStop(hApiQuery As Long) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiQuery**: This parameter contains a valid handle of a query object.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmReceiveAllPOLines()

This function receives all the items on an order line (takes delivery in full).

Note:

Warning: Delivery lines are created by an agent when the transaction is committed. You cannot access them beforehand.

API syntax

long AmReceiveAllPOLines(long hApiCnxBase, long lPOrdId, long lDelivId);

Internal BASIC syntax

Function AmReceiveAllPOLines(lPOrdId As Long, lDelivId As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lPOrdId**: This parameter contains the identifier of the order line containing the items to be received.
- **lDelivId**: This parameter contains the identifier of the receiving slip used to receive all the items present on the order line.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmReceivePOLine()

This function takes delivery of a certain quantity of items on an order line (takes delivery in part) and returns the identifier of the delivery line.

Note:

Warning: The delivery lines are created by an agent as soon as the transaction is committed. You cannot access them until this is performed.

API syntax

long AmReceivePOLine(long hApiCnxBase, long lPOrdLineId, long lDelivId, float fQty);

Internal BASIC syntax

Function AmReceivePOLine(lPOrdLineId As Long, lDelivId As Long, fQty As Single) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lPOrdLineId**: This parameter contains the identifier of the purchase order line containing the items to be received.

- **lDelivId**: This parameter contains the identifier of the receiving slip used to receive a certain quantity of items present on the order line.
- **fQty**: This parameter contains the quantity of items on the order line to be received in the receiving slip.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmRefreshAllCaches()

This function refreshes the caches used in AssetCenter.

API syntax

long AmRefreshAllCaches(long hApiCnxBase);

Internal BASIC syntax

Function AmRefreshAllCaches() As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓

	Available
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmRefreshLabel()

The AmRefreshLabel API refreshes the label string of a given record (lMainId) in a given table (strTableName).

API syntax

long AmRefreshLabel(long hApiCnxBase, long lMainId, char *strTableName, char *pstrLabel, long lLabel);

Internal BASIC syntax

Function AmRefreshLabel(lMainId As Long, strTableName As String) As String

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lMainId**: This parameter defines the ID that will be refreshed.
- **strTableName**: This parameter defines the table name for the lMainId.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmRefreshProperty()

Reevaluates the value of a property identified by the **strVarName** parameter. If this property uses a script, the script is executed again.

Otherwise the tree of dependencies is updated.

Internal BASIC syntax

Function AmRefreshProperty(strVarName As String) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strVarName**: Name of the property (of the wizard) that you want to reevaluate.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmRefreshTraceHist()

The AmRefreshTraceHist API refreshes a complete project trace entry and also has an optional parameter to allow refreshing of "individual" trace history entries. If this parameter is not provided, the complete trace history will be refreshed.

API syntax

long AmRefreshTraceHist(long hApiCnxBase, long lCabTraceOutId, long lTraceHistId);

Internal BASIC syntax

Function AmRefreshTraceHist(lCabTraceOutId As Long, lTraceHistId As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓

	Available
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lCabTraceOutId**: This parameter is the cable trace output ID.
- **lTraceHistId**: This parameter is an optional parameter to allow refreshing of "individual" trace history entries.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmReleaseHandle()

This function frees the handle and sub-handles of an object.

API syntax

long AmReleaseHandle(long hApiObject);

Internal BASIC syntax

Function AmReleaseHandle(hApiObject As Long) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	✓

	Available
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **hApiObject**: This parameter contains a handle of the object concerned.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmRemoveCable()

The AmRemoveCable API removes a cable (lCableId) from its current location. The status of the cable is updated to "Unavailable". If the project (lProjectId) and work order (lWorkOrderId) have values, the cable is added to the project and work order with comment contained in the given comment (strComment). This comment describes the action that will be performed on the cable (i.e. "Remove cable from its current location").

API syntax

long AmRemoveCable(long hApiCnxBase, long lCableId, long lProjectId, long lWorkOrderId, char *strComment);

Internal BASIC syntax

Function AmRemoveCable(lCableId As Long, lProjectId As Long, lWorkOrderId As Long, strComment As String) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lCableId**: This parameter is the ID of the cable to remove.
- **lProjectId**: This parameter is the project ID.
- **lWorkOrderId**: This parameter defines the work order ID.
- **strComment**: This parameter is the comment that will be used on the work order.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmRemoveDevice()

The AmRemoveDevice API removes a device (lAssetId) from its current location. The status of the device is updated to "Unavailable". If the project (lProjectId) and work order (lWorkOrderId) have values, the device is added to the project and work order with the given comment (strComment). This comment describes the action that will be performed on the device (i.e. "Remove device from its current location").

API syntax

long AmRemoveDevice(long hApiCnxBase, long lDeviceId, long lProjectId, long lWorkOrderId, char *strComment);

Internal BASIC syntax

Function AmRemoveDevice(lDeviceId As Long, lProjectId As Long, lWorkOrderId As Long, strComment As String) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lDeviceId**: This parameter defines the device ID to remove.
- **lProjectId**: This parameter is the project ID.
- **lWorkOrderId**: This parameter defines the work order ID.
- **strComment**: This parameter is the comment that will be used on the work order.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmReturnAsset()

This function enables you to return an asset.

API syntax

long AmReturnAsset(long hApiCnxBase, long lAstId, long lReturnId, long bCanMerge);

Internal BASIC syntax

Function AmReturnAsset(lAstId As Long, lReturnId As Long, bCanMerge As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lAstId**: This parameter contains the identifier of the asset to return.
- **lReturnId**: This parameter contains the identifier of the return slip.
- **bCanMerge**: This parameter enables you to specify whether the return can be merged with an existing line in the return slip.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmReturnContract()

This function enables you to return a contract.

API syntax

long AmReturnContract(long hApiCnxBase, long lCntrId, long lReturnId, long bCanMerge);

Internal BASIC syntax

Function AmReturnContract(lCntrId As Long, lReturnId As Long, bCanMerge As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lCntrId**: This parameter contains the identifier of the contract to return.
- **lReturnId**: This parameter contains the identifier of the return slip.

- **bCanMerge**: This parameter enables you to specify whether the return can be merged with an existing line in the return slip.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmReturnPortfolioItem()

This function enables you to return a portfolio item.

API syntax

long AmReturnPortfolioItem(long hApiCnxBase, long lPfId, float fQty, long lFromRecptLineId, long lReturnId, long bCanMerge);

Internal BASIC syntax

Function AmReturnPortfolioItem(lPfId As Long, fQty As Single, lFromRecptLineId As Long, lReturnId As Long, bCanMerge As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓

	Available
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lPfId**: This parameter contains the identifier of the portfolio item to return.
- **fQty**: This parameter contains the quantity (in the unit of the model) to return.
- **lFromRecptLineId**: This parameter contains the identifier of the source receipt line.
- **lReturnId**: This parameter contains the identifier of the return slip.
- **bCanMerge**: This parameter enables you to specify whether the return can be merged with an existing line in the return slip.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmReturnTraining()

This function enables you to return a training.

API syntax

long AmReturnTraining(long hApiCnxBase, long lTrainingId, long lReturnId, long bCanMerge);

Internal BASIC syntax

Function AmReturnTraining(lTrainingId As Long, lReturnId As Long, bCanMerge As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lTrainingId**: This parameter contains the identifier of the training to return.
- **lReturnId**: This parameter contains the identifier of the return slip.
- **bCanMerge**: This parameter enables you to specify whether the return can be merged with an existing line in the return slip.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmReturnWorkOrder()

This function enables you to return a work order.

API syntax

long AmReturnWorkOrder(long hApiCnxBase, long lWOId, long lReturnId, long bCanMerge);

Internal BASIC syntax

Function AmReturnWorkOrder(lWOId As Long, lReturnId As Long, bCanMerge As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **lWOId**: This parameter contains the identifier of the work order to return.
- **lReturnId**: This parameter contains the identifier of the return slip.
- **bCanMerge**: This parameter enables you to specify whether the return can be merged with an existing line in the return slip.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmRevCryptPassword()

This function decrypts an encrypted password.

API syntax

long AmRevCryptPassword(long hApiCnxBase, char *return, long lreturn, char *strPassword);

Internal BASIC syntax

Function AmRevCryptPassword(strPassword As String) As String

Field of application

Version: 3.5

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strPassword**: This parameter contains the password to decrypt.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmRgbColor()

This function gives the RGB value of the color corresponding to the **strText** parameter.

API syntax

long AmRgbColor(char *strText);

Internal BASIC syntax

Function AmRgbColor(strText As String) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strText**: This parameter contains the name of the color:
  - White
  - ltGray
  - Gray

- Dkgray
- Black
- Red
- Green
- Blue
- Yellow
- Cyan
- Magenta
- Dkyellow
- Dkgreen
- Dkcyan
- Dkblue
- Dkmagenta
- Dkred

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmRollback()

This function cancels all modifications made before the declaration of the start of the transaction (performed via the **AmStartTransaction** function).

API syntax

long AmRollback(long hApiCnxBase);

Internal BASIC syntax

Function AmRollback() As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmSetFieldDateValue()

This function modifies a field in a record. This function does not update the database. The modification will be made when the record is updated or inserted, or when the transaction is committed.

API syntax

long AmSetFieldDateValue(long hApiRecord, char *strFieldName, long tmValue);

Internal BASIC syntax

Function AmSetFieldDateValue(hApiRecord As Long, strFieldName As String, tmValue As Date) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **hApiRecord**: This parameter contains the handle of the record containing the field to be modified.
- **strFieldName**: This parameter contains the SQL name of the field to be modified.
- **tmValue**: This parameter contains the new value of the field in "Date" format.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmSetFieldDoubleValue()

This function modifies a field in a record. This function does not update the database.

API syntax

long AmSetFieldDoubleValue(long hApiRecord, char *strFieldName, double dValue);

Internal BASIC syntax

Function AmSetFieldDoubleValue(hApiRecord As Long, strFieldName As String, dValue As Double) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **hApiRecord**: This parameter contains the handle of the record containing the field to be modified.
- **strFieldName**: This parameter contains the SQL name of the field to be modified.
- **dValue**: This parameter contains the new value of the field in "Double" format.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmSetFieldLongValue()

This function modifies a field in a record. This function does not update the database. To modify the value of a date, time or date+time date you must express the new value in terms of seconds elapsed since 01/01/1970 at 00:00:00.

API syntax

long AmSetFieldLongValue(long hApiRecord, char *strFieldName, long lValue);

Internal BASIC syntax

Function AmSetFieldLongValue(hApiRecord As Long, strFieldName As String, lValue As Long) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **hApiRecord**: This parameter contains the handle of the field that to be modified.
- **strFieldName**: This parameter contains the SQL name of the field to be modified.
- **lValue**: This parameter contains the new value of the field.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmSetFieldStrValue()

This function modifies a field in a record. This function does not update the database.

API syntax

long AmSetFieldStrValue(long hApiRecord, char *strFieldName, char *strValue);

Internal BASIC syntax

Function AmSetFieldStrValue(hApiRecord As Long, strFieldName As String, strValue As String) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **hApiRecord**: This parameter contains the handle of the record containing the field to be modified.
- **strFieldName**: This parameter contains the SQL name of the field to be modified.
- **strValue**: This parameter contains the new value of the field in "String" format.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmSetLinkFeatureValue()

This function sets the value of a link type feature for a given record.

API syntax

long AmSetLinkFeatureValue(long hApiRecord, char *strFeatSqlName, char *strDstSelfValue, long lDstId);

Internal BASIC syntax

Function AmSetLinkFeatureValue(hApiRecord As Long, strFeatSqlName As String, strDstSelfValue As String, lDstId As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **hApiRecord**: This parameter contains the identifier of the record to which the link type feature is associated.

- **strFeatSqlName**: This parameter contains the SQL name of the link type feature whose value you want to set. This SQL name is always preceded by "fv_".
- **strDstSelfValue**: This parameter contains the value of the feature as it will be displayed for the record. It is the "Self" value of the record with identifier **lDstId**. If you pass an invalid or non-existent value, you take the risk of corrupting the integrity of the database.
- **lDstId**: This parameter contains the identifier of the record to which the link type feature points.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmSetProperty()

This function sets the value of a property identified by its name. It also updates the tree of dependencies of this property.

Internal BASIC syntax

Function AmSetProperty(strVarName As String, vValue As Variant) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strVarName**: This parameter contains the name of the property whose value you want to set.
- **vValue**: This parameter contains the new value for the property.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmShowCableCrossConnect()

This function displays the cross-connections screen.

Internal BASIC syntax

Function AmShowCableCrossConnect(lCableId As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lCableId**: This parameter contains the identifier of the cable concerned by the operation.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmShowDeviceCrossConnect()

This function displays the cross-connections screen for a cable device.

Internal BASIC syntax

Function AmShowDeviceCrossConnect(lDeviceId As Long) As Long

Field of application

Version: 4.00

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lDeviceId**: This parameter contains the identifier of the cable device concerned by the operation.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmSqlTextConst()

This function transforms a string to be used in a query. The following operations are performed on the string:

- All single quotes (') are doubled,
- Single quotes are added at the start and end of the string.

API syntax

long AmSqlTextConst(char *return, long lreturn, char *str);

Internal BASIC syntax

Function AmSqlTextConst(str As String) As String

Field of application

Version: 4.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **str**: This parameter contains the character string to process.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim strReq as String
strReq="SELECT lEmplDeptId FROM amEmplDept WHERE Name=" & amSqlTextConst(
strName)
```

This query is valid, even if the strName variable contains single quotes.

AmStartTransaction()

This function starts a new transaction with the database associated with the connection. The next "Commit" or "Rollback" statement will validate or cancel all the modifications made to the database.

API syntax

long AmStartTransaction(long hApiCnxBase);

Internal BASIC syntax

Function AmStartTransaction() As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓

	Available
Wizard script	
FINISH.DO script of a wizard	✓

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmStartup()

This function must be applied before all other functions. It initializes calls to the AssetCenter library.

API syntax

void AmStartup();

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	

AmTableDesc()

This function generates a character string with the format "<Description of the table> (<SQL name of the table>)" from the SQL name of the table.

API syntax

long AmTableDesc(long hApiCnxBase, char *return, long lreturn, char *strSqlName);

Internal BASIC syntax

Function AmTableDesc(strSqlName As String) As String

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strSqlName**: SQL name of the table for which a description string is required. If this parameter contains an invalid SQL name, the function returns a question mark ("?").

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following example generates a description string for the table of assets (SQL name: amAsset):

```
AmTableDesc("amAsset")
```

The result is as follows:

```
Assets (amAsset)
```

AmTaxRate()

This function calculates a tax rate according to a tax type, tax jurisdiction and a date.

API syntax

double AmTaxRate(char *strTaxRateName, long lTaxLocId, long tmDate, double dValue);

Internal BASIC syntax

Function AmTaxRate(strTaxRateName As String, lTaxLocId As Long, tmDate As Date, dValue As Double) As Double

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strTaxRateName**: This parameter contains the SQL name of the tax type used to calculate the tax rate.
- **lTaxLocId**: This parameter contains the ID number of the tax jurisdiction concerned by the tax type.
- **tmDate**: This parameter contains the date for which you want to know the tax rate.
- **dValue**: Obsolete parameter, kept for compatibility reasons. Do not use.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

AmUpdateDetail()

This function is used in the data-entry wizards. The context (table for which a record is updated or populated or updated using the wizard) is therefore clearly defined. The function updates or populates fields or links of the context according to a value. This function not allowed in non-modal wizards.

Internal BASIC syntax

Function AmUpdateDetail(strFieldName As String, varValue As Variant) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **strFieldName**: This parameter contains the SQL name of the feature to be updated.
- **varValue**: This parameter contains the new value of the field.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmUpdateRecord()

This function enables you to update a record.

API syntax

long AmUpdateRecord(long hApiRecord);

Internal BASIC syntax

Function AmUpdateRecord(hApiRecord As Long) As Long

Field of application

Version: 2.52

	Available
AssetCenter API	✓
Configuration script of a field or link	
"Script" type action	✓
Wizard script	
FINISH.DO script of a wizard	✓

Input parameters

- **hApiRecord**: This parameter contains a handle of the record containing the field to be updated.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmValueOf()

Used in a wizard, this function returns the value of the property identified by the **strVarName** parameter.

Internal BASIC syntax

Function AmValueOf(strVarName As String) As Variant

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓

	Available
FINISH.DO script of a wizard	✓

Input parameters

- **strVarName**: This parameter contains the name of the property whose value we want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following example returns the value of the "Page1.Label" property:

```
AmValueOf("Page1.Label")
```

Use this function with care because it breaks the dependency string of the property being processed.

AmWizChain()

This function executes a wizard B, inside a wizard A. When wizard B has finished executing, wizard A takes over again.

Internal BASIC syntax

Function AmWizChain(strWizSqlName As String) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strWizSqlName**: SQL name of the wizard to be executed.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

AmWorkTimeSpanBetween()

This function returns the duration of working periods between two dates. This duration is expressed in seconds; it respects the information in a calendar of working periods.

API syntax

long AmWorkTimeSpanBetween(char *strCalendarSqlName, long tmEnd, long tmStart);

Internal BASIC syntax

Function AmWorkTimeSpanBetween(strCalendarSqlName As String, tmEnd As Date, tmStart As Date) As Date

Field of application

Version: 3.00

	Available
AssetCenter API	✓
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strCalendarSqlName**: This parameter contains the SQL name of the calendar of working periods used to calculate the duration of the working period between the two dates. If this parameter is omitted, the calculated duration does not take working periods into account.
- **tmEnd**: This parameter contains the end date for the period used in calculating the working period.
- **tmStart**: This parameter contains the start date for the period used in calculating the working period.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following example calculates the working period between 01/09/1998 at 8 a.m. and 24/09/1998 at 7 p.m. The calendar used, whose SQL name is "Calendar_Paris", defines the following working periods:

- From Monday to Thursday from 8 a.m. to 12 noon, then from 2 p.m. to 6 p.m.
- Fridays from 8 a.m. to 12 noon, then from 2 p.m. to 5 p.m.

```
AmWorkTimeSpanBetween("Calendar_Paris", "1998/09/24 19:00:00", "1998/09/0
1 08:00:00")
```

This example returns the value 507,600 which represents the number of working seconds between the two dates.

AppendOperand()

Concatenates a string according to the parameters passed to the function. The results are given as follows:

```
strExprstrOperatorstrOperand
```

Internal BASIC syntax

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

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strExpr**: Expression to be concatenated.
- **strOperator**: Operator to concatenate.

- **strOperand**: Operand to concatenate.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

Note:

If one of the **strExpr** or **strOperand** parameters is omitted, **strOperator** is not used in the concatenation.

ApplyNewVals()

Assigns identical values to identical cells in a "ListBox" control.

Internal BASIC syntax

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

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓

	Available
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strValues**: Source string containing the values of a "ListBox" control to be processed.
- **strNewVals**: New value to assign to the cells concerned.
- **strRows**: Identifiers of lines to be processed. The identifiers are separated by commas.
- **strRowFormat**: Formatting instructions for the sublist. Instructions are separated by the "|" character. Each instruction represents the number of the column containing the **strNewVals** parameter.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Asc()

Returns a numeric value that is the ASCII code for the first character in a string.

Internal BASIC syntax

Function Asc(strAsc As String)

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strAsc**: Character sting on which the function operates.

Example

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

Atn()

Returns the arc tangent of a number, expressed in radians.

Internal BASIC syntax

Function Atn(dValue As Double) As Double

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dValue**: Number for which you want to know the arc tangent.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

BasicToLocalDate()

This function converts a Basic format date to a string format date (as displayed in Windows Control Panel).

Internal BASIC syntax

Function BasicToLocalDate(strDateBasic As String) As String

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strDateBasic**: Date in Basic format to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

BasicToLocalTime()

This function converts a Basic format time to a string format time (as displayed in Windows Control Panel).

Internal BASIC syntax

Function BasicToLocalTime(strTimeBasic As String) As String

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strTimeBasic**: Time in Basic format to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

BasicToLocalTimeStamp()

This function converts a Date+Time in Basic format to a Date+Time in string format (as displayed in Windows Control Panel).

Internal BASIC syntax

Function BasicToLocalTimeStamp(strTSBasic As String) As String

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strTSBasic**: Date+Time in Basic format to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Beep()

Plays a beep on the machine.

Internal BASIC syntax

Function Beep()

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

CDbl()

Converts an expression to a "Double".

Internal BASIC syntax

Function CDbl(dValue As Double) As Double

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dValue**: Expression to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

ChDir()

Changes the current directory.

Internal BASIC syntax

Function ChDir(strDirectory As String)

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓

	Available
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strDirectory**: New current directory.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

ChDrive()

Changes the current drive.

Internal BASIC syntax

Function ChDrive(strDrive As String)

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓

	Available
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strDrive**: New drivename.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Chr()

Returns a string corresponding to the ASCII passed by the **iChr** parameter.

Internal BASIC syntax

Function Chr(iChr As Long) As String

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓

	Available
FINISH.DO script of a wizard	✓

Input parameters

- **iChr**: ASCII code of the character.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim iCount as Integer
Dim iIteration as Integer
Dim strMessage as String
Dim strLF as String
  strLF=Chr(10)
  For iIteration=1 To 2
    For iCount=Asc("A") To Asc("Z")
      strMessage=strMessage+Chr(iCount)
    Next iCount
    strMessage=strMessage+strLF
  Next iIteration
  RetVal=strMessage
```

CInt()

Converts any valid expression to an Integer.

Internal BASIC syntax

Function CInt(iValue As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **iValue**: Expression to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

CLng()

Converts any valid expression to a Long.

Internal BASIC syntax

Function CLng(lValue As Long) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lValue**: Expression to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

Cos()

Returns the cosine of a number, expressed in radians.

Internal BASIC syntax

Function Cos(dValue As Double) As Double

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dValue**: Number whose cosine you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

CountOccurences()

Counts the number of occurrences of a string inside another string.

Internal BASIC syntax

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

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strSearched**: Character string in which to perform to the search.
- **strPattern**: Character string to find inside the **strSearched** parameter.
- **strEscChar**: Escape character. If the function encounters this character inside the **strSearched** string, the search stops.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyStr
  MyStr=CountOccurences("you|me|you,me|you", "you", ",") :'Returns "2"
  MyStr=CountOccurences("you|me|you,me|you", "you", "|") :'Returns "1"
```

CountValues()

Counts the number of elements in a string, taking into account a separator and an escape character.

Internal BASIC syntax

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

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strSearched**: Character string to process.
- **strSeparator**: Separator used to delimit the elements.
- **strEscChar**: Escape character. If this character prefixes a separator, it will be ignored.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyStr
  MyStr=CountValues("you|me|you\|me|you", "|", "\") :'Returns 4
  MyStr=CountValues("you|me|you\|me|you", "|", "") :'Returns 5
```

CSng()

Converts any valid expression to a floating point number ("Float").

Internal BASIC syntax

Function CSng(fValue As Single) As Single

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **fValue**: Expression to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

CStr()

Converts any valid expression to a String.

Internal BASIC syntax

Function CStr(strValue As String) As String

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strValue**: Expression to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

CurDir()

Returns the current path.

Internal BASIC syntax

Function CurDir() As String

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

CVar()

Converts any valid expression to a Variant.

Internal BASIC syntax

Function CVar(vValue As Variant) As Variant

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **vValue**: Expression to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaContext()

This function returns the value of a context identified by its name.

Internal BASIC syntax

Function DaContext(strField As String) As Variant

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	

	Available
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strField**: This parameter contains the name of the context for which you want to find the value.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

This example verifies that the environment variables can be recovered.

```
' Check that environment variables can be retrieved - this also retrieve
all
' variables in one shot, improving performances
lRc = DaFirstEnv()

' Get useful environment variables and store them in context
if lRc <> 0 then
 ' Get environment variables
 strProcArch      = DaGetEnv( "PROCESSOR_ARCHITECTURE" )
 strProcId        = DaGetEnv( "PROCESSOR_IDENTIFIER" )
 strProcLevel     = DaGetEnv( "PROCESSOR_LEVEL" )
 strProcRevision  = DaGetEnv( "PROCESSOR_REVISION" )
 strProcCount     = DaGetEnv( "NUMBER_OF_PROCESSORS" )
 strOs            = DaGetEnv( "OS" )
 strComputer      = DaGetEnv( "COMPUTERNAME" )

 ' Check that computer name is the right one
 if strComputer <> DaContext( "Computer.Name" ) then
   print "Warning: computer name does not look like the right one !"
 end if
```

```
 ' Store them in context
 DaSetContext "EnvInfo.ProcArch", strProcArch
 DaSetContext "EnvInfo.ProcId", strProcId
 DaSetContext "EnvInfo.ProcLevel", strProcLevel
 DaSetContext "EnvInfo.ProcRevision", strProcRevision
 DaSetContext "EnvInfo.ProcCount", strProcCount
 DaSetContext "EnvInfo.Os", strOs
 DaSetContext "EnvInfo.Computer", strComputer

 ' Dump
 print DaDumpContext()
end if
```

DaCopy()

This function copies a file, a set of files or a folder, locally. This function works in one of two ways:

- It copies a file locally to the deployment server: The source and the destination of the copy are both on the deployment server.
- It copies a file locally to the deployment target: The source and the destination of the copy are both on the deployment target.

The way in which it works is determined by the value of the **file.on_server** option, set using the **DaSetOption()** function:

- If this option is set to 1, the copy concerns the deployment server.
- If this option is set to 0, the copy concerns the deployment target.

Note:

For further information, refer to the description of the **DaSetOption()** function.

Note:

This function carries out the same operations as the **Copy files** activity.

Internal BASIC syntax

Function DaCopy(strSource As String, strDest As String, strNameFilter As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strSource**: This parameter contains the destination path of the elements to copy. Depending on how the function operates, it is either a path relative to the file depot (in the case of a local copy to the deployment server) or an absolute path (in the case of a local copy to the deployment target).
- **strDest**: Destination path of the elements to copy. Depending on how the function operates, it is either a path relative to the file depot (in the case of a local copy to the deployment server) or an absolute path (in the case of a local copy to the deployment target).
- **strNameFilter**: This parameter contains wildcard characters, as used by DOS (* and ?), used to filter the file names.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
' copy user spreadsheet documents to a backup folder
dim l as long
```

```
l = DaSetOption( "file.recursive", 1 )
l = DaCopy( "c:\%USERPROFILE%", "c:\backup", "\*.xls" )
```

DaDbDeleteList()

This function deletes one or more records from the AssetCenter database.

Internal BASIC syntax

Function DaDbDeleteList(strTable As String, strList As String, strLineSep As String, strIdSep As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strTable**: This parameter contains the SQL name of the table in which the records are inserted or updated.
- **strList**: This parameter contains the character string describing the record to be updated or inserted.

Note:

This parameter can be populated by the list obtained by the **DbGetList** function, or if the **strIdStep** parameter is empty, by a list of record identifiers.

- **strLineSep**: This parameter contains the character used as line separator in the character string contained in the **strList** parameter.
- **strIdSep**: This parameter contains the identifier of the record to be updated or inserted. If this parameter is null, the function considers that it is an insertion. If the value is not null, the function considers that it is an update of the record having this value as the identifier.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaDbGetList()

This function returns, as a list, the result of an AQL query.

Note:

For further information on the AQL query language, refer to the chapter **AQL** in the manual entitled **Advanced use**.

Internal BASIC syntax

Function DaDbGetList(strQuery As String, strColSep As String, strLineSep As String, strIdSep As String) As String

Field of application

Version: 1.0

	Available
AssetCenter API	

	Available
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strQuery**: This parameter contains the AQL query you want to execute.
- **strColSep** :This parameter contains the character use as column separator in the result given by the function.

  A column is equivalent to a field in a record returned by the query.
- **strLineSep**: This parameter contains the character used as line separator in the result returned by the function.

  A line is equivalent to a record returned by the query.
- **strIdSep**: This parameter contains the character used to separate record identifiers in the results given by the function.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim strList As String
Dim lWorkgroup, lErr As Long

' Get list of computer ids in workgroup 'ToChange'
strList = DaDbGetList( "SELECT Workgroup FROM amComputer WHERE Workgroup=
'ToChange'", "|", "/", "@" )

' Replace all 'ToChange' workgroups by 'Changed'. InStr will return 0 if
not found
lWorkgroup = InStr( 0, strList, "ToChange" )
```

```
Do While lWorkgroup > 0
  strList = Left$( strList, lWorkgroup - 1 ) + "Changed" + Mid$( strList,
 lWorkgroup + Len("ToChange"), Len(strList) )

  lWorkgroup = InStr( 0, strList, "ToChange" )
Loop

' Update computers
lErr = DaDbSetList( "amComputer", "Workgroup", strList, "|", "/", "@" )
```

DaDbSetList()

This function updates or creates records in a table of the AssetCenter database.

Note:

This function is the opposite of the **DaDbGetList** function.

Internal BASIC syntax

Function DaDbSetList(strTable As String, strFields As String, strList As String, strColSep As String, strLineSep As String, strIdSep As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strTable**: SQL name of the table concerned by the operation.

- **strFields**: This parameter contains the SQL name of the fields to be modified (in the case of the modification of an existing record) or populated (in the case of the creation of a new record). The character defined in the **strColSep** parameter is used as a separator for the SQL names of the fields.
- **strList**: list of the records concerned by the operation. Each record is separated by the **strLineSep** parameter. In the case of a modification (update of a record) the identifier of the record to modify is specified by appending the character contained in the **strIdSep** parameter, followed by the record identifier.
- **strColSep**: This parameter contains the character used as column separator (a column represents a field).
- **strLineSep**: This parameter contains the character used as a record separator (line separator).
- **strIdSep**: This parameter contains the character used to specify the identifier of the record to modify.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
' Update name and workgroup of the computer '42' and insert a new compute
r named 'NewComputer' in computers table
' We could also use 'NewName|NewWorkgroup@0' to insert the new computer.

lErr = DaDbSetList( "amComputer", "Name|Workgroup", "UpdatedName|UpdatedW
orkgroup@42/NewName|NewWorkgroup","|", "/", "@" )
```

DaDelete()

This function copies a file, a set of files or a folder, locally. This function works in one of two ways:

- Delete from the deployment server (from the file depot).
- Delete from the deployment target.

The way in which it works is determined by the value of the **file.on_server** option, set using the **DaSetOption()** function:

- If this option is set to 1, the deletion concerns the deployment server.
- If this option is set to 0, the deletion concerns the deployment target.

Note:

For further information, refer to the description of the **DaSetOption()** function.

Note:

This function carries out the same operations as the **Delete files** activity.

Internal BASIC syntax

Function DaDelete(strToDelete As String, strNameFilter As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strToDelete**: This parameter contains the path of the files to delete. If you want to delete the elements from the deployment server, it is a relative path of the absolute path of the file depot. If you want to delete the elements from the deployment target, it is an absolute path.
- **strNameFilter**: This parameter contains wildcard characters, as used by DOS (* and ?), used to filter the file names.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
'Remove temporary files from server depot
dim l as long
l = DaSetOption( "file.recursive", 1 )
l = DaSetOption( "file.on_server", 1 )
l = DaDelete( "", "*.tmp" )
```

DaDownload()

This function copies a file, folder or set of files to the deployment server depot. The source files must be located on the deployment target.

You can also configure the behavior of this function by setting certain options using the **DaSetOption()** option. A full list of this option is included in the description of the **DaSetOption()** function.

Note:

This function carries out the same operations as the **Download files** activity.

Internal BASIC syntax

Function DaDownload(strSrcPath As String, strDstPath As String, strSrcNameFilter As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strSrcPath**: This parameter contains the path of the files to copy to the depot of the deployment server. It is an absolute path on the deployment target. For example:

  ```
  c:\program files\antivirus\update\file.exe
  ```

- **strDstPath**: This parameter contains the path of the file copied on the deployment server. This path is relative to the absolute path of the deployment server. For example, if the absolute path of the file depot is:

  ```
  c:\files\depot
  ```

  and you want to copy a file to this location:

  ```
  c:\files\depot\software\antivirus\update\file.exe
  ```

  then this parameter will have the following value:

  ```
  software\antivirus\update
  ```

- **strSrcNameFilter**: This parameter contains wildcard characters, as used by DOS (* and **?**), used to filter the file names.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
'get target specific path within file depot
dim path as string
path = "logs/" & DaContext( "Computer.Name" )

' retrieve a log file from target
dim l as long
l = DaDownload( "c:\program files\application\setup", path, "setup.log" )
```

DaDumpContext()

This function returns a string containing all the defined contexts and is mainly used in debug mode.

Internal BASIC syntax

Function DaDumpContext() As String

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓

	Available
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
' Check that environment variables can be retrieved - this also retrieve
all
' variables in one shot, improving performances
lRc = DaFirstEnv()

' Get usefull environment variables and store them in context
if lRc <> 0 then
 ' Get environment variables
 strProcArch      = DaGetEnv( "PROCESSOR_ARCHITECTURE" )
 strProcId        = DaGetEnv( "PROCESSOR_IDENTIFIER" )
 strProcLevel     = DaGetEnv( "PROCESSOR_LEVEL" )
 strProcRevision  = DaGetEnv( "PROCESSOR_REVISION" )
 strProcCount     = DaGetEnv( "NUMBER_OF_PROCESSORS" )
 strOs            = DaGetEnv( "OS" )
 strComputer      = DaGetEnv( "COMPUTERNAME" )

 ' Check that computer name is the right one
 if strComputer <> DaContext( "Computer.Name" ) then
   print "Warning: computer name does not look like the right one !"
 end if

 ' Store them in context
 DaSetContext "EnvInfo.ProcArch", strProcArch
 DaSetContext "EnvInfo.ProcId", strProcId
 DaSetContext "EnvInfo.ProcLevel", strProcLevel
 DaSetContext "EnvInfo.ProcRevision", strProcRevision
 DaSetContext "EnvInfo.ProcCount", strProcCount
 DaSetContext "EnvInfo.Os", strOs
 DaSetContext "EnvInfo.Computer", strComputer

 ' Dump
```

```
 print DaDumpContext()
end if
```

DaExec()

This function execute a program stored on the deployment target.

You can also configure the behavior of this function by setting certain options using the **DaSetOption()** option. A full list of this option is included in the description of the **DaSetOption()** function.

Note:

This function performs the same operations as the **Execute** activity.

Internal BASIC syntax

Function DaExec(strCmd As String, strPath As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strCmd**: This parameter contains the name of the program to be executed. For example:

```
notepad.exe
```

- **strPath**: This parameter contains the full path of the executable. In the case of the previous example:

```
c:\windows\system32
```

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
' Log ipconfig information
lErr = DaSetOption( "exec.log_output", 1 )
lErr = DaExec( "ipconfig /all", "" )
```

DaExecAction()

This function executes an action, identified by its name, on a table in the AssetCenter database.

Note:

This function supports a limited number of tables (workflows, agents, computers, etc.).

Internal BASIC syntax

Function DaExecAction(strAction As String, strTable As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strAction**: This parameter contains the SQL name of the action as defined in AssetCenter.
- **strTable**: This parameter contains the SQL name of the table on which the action is executed.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim lErr as Long
lErr = DaExecAction( "MyAction", "amComputer")
```

DaExecuteActionByName()

This function executes an action, identified by its name, on a table and a record in the AssetCenter database.

Internal BASIC syntax

Function DaExecuteActionByName(strAction As String, strTable As String, lRecord As Long) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strAction**: This parameter contains the SQL name of the action as defined in AssetCenter.
- **strTable**: This parameter contains the SQL name of the table on which the action is executed.
- **lRecord**: This parameter contains the identifier of the record in the table on which the action is executed.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

Note:

If no identifier is specified, the record number linked to the table is used.

Example

```
Dim lErr as Long
lErr = DaExecuteActionByName( "MyAction", "amComputer", 3)
```

DaFileATime()

This function returns the date and time at which the file or folder were accessed.

Internal BASIC syntax

Function DaFileATime(strPath As String) As Date

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strPath**: This parameter contains the path of the file or folder concerned by the operation. If the operation is carried out on the deployment server,

it is a path relative to the absolute path of the file depot. If the operation is carried out on the deployment target, it is an absolute path.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaFileCRC()

This function returns the CRC of a file (Cyclic Redundancy Check).

Internal BASIC syntax

Function DaFileCRC(strPath As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strPath**: This parameter contains the path of the file or folder concerned by the operation. If the operation is carried out on the deployment server,

it is a path relative to the absolute path of the file depot. If the operation is carried out on the deployment target, it is an absolute path.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaFileCTime()

This function returns the date and time of creation of the file or folder.

Internal BASIC syntax

Function DaFileCTime(strPath As String) As Date

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strPath**: This parameter contains the path of the file or folder concerned by the operation. If the operation is carried out on the deployment server,

it is a path relative to the absolute path of the file depot. If the operation is carried out on the deployment target, it is an absolute path.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaFileLanguage()

This function returns the language of the file (if available).

Internal BASIC syntax

Function DaFileLanguage(strPath As String) As String

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strPath**: This parameter contains the path of the file or folder concerned by the operation. If the operation is carried out on the deployment server,

it is a path relative to the absolute path of the file depot. If the operation is carried out on the deployment target, it is an absolute path.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaFileMTime()

This function returns the date and time at which the file or folder were modified.

Internal BASIC syntax

Function DaFileMTime(strPath As String) As Date

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strPath**: This parameter contains the path of the file or folder concerned by the operation. If the operation is carried out on the deployment server, it is a path relative to the absolute path of the file depot. If the operation is carried out on the deployment target, it is an absolute path.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaFileSize()

This function returns the size of a file in bits. If the size of the file is too large (i.e it exceeds a long), the returned value is -1.

Internal BASIC syntax

Function DaFileSize(strPath As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strPath**: This parameter contains the path of the file or folder concerned by the operation. If the operation is carried out on the deployment server, it is a path relative to the absolute path of the file depot. If the operation is carried out on the deployment target, it is an absolute path.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaFileType()

This function returns the file type concerned by the operation.

Internal BASIC syntax

Function DaFileType(strPath As String) As String

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strPath**: This parameter contains the path of the file or folder concerned by the operation. If the operation is carried out on the deployment server, it is a path relative to the absolute path of the file depot. If the operation is carried out on the deployment target, it is an absolute path.

Output parameters

The function returns one of the following values:

- "r" for a file,
- "d" for a directory,
- "D" for a physical drive.

DaFileVersion()

This function returns the version of a file (if available).

Internal BASIC syntax

Function DaFileVersion(strPath As String) As String

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strPath**: This parameter contains the path of the file or folder concerned by the operation. If the operation is carried out on the deployment server, it is a path relative to the absolute path of the file depot. If the operation is carried out on the deployment target, it is an absolute path.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaFind()

This function performs a recursive search of the files and folders and creates a list of the files it finds. The list is then scanned using the **DaFindNext()** function.

Internal BASIC syntax

Function DaFind(strPath As String, strNameFilter As String, lDepth As Long) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓

	Available
FINISH.DO script of a wizard	✓

Input parameters

- **strPath**: This parameter contains the path of the search folder.
- **strNameFilter**: This parameter contains wildcard characters, as used by DOS (* and ?), used to filter the search.
- **lDepth**: This parameter contains the depth of the recursive research, in number of sub-folders under the **strPath** folder.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
' Builds a list of all bmp files
Dim lErr as Long
lErr = DaFind( "install", "*.bmp", 3)
```

DaFindNext()

This function retrieves the name of the next file or folder in the list created by the **DaFind()** function. When the end of the list is reached, the function returns an empty character string.

Internal BASIC syntax

Function DaFindNext() As String

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim strFileName as String
strFileName = DaFindNext()
If strFileName <> 0 Then
  Print strFileName
Else
  Print "Reached end of file list"
End If
```

DaFirstEnv()

This function builds the list of environment variables available on the deployment target. This list can be browsed using the **DaNextEnv()** and **DaGetEnv()** functions.

Internal BASIC syntax

Function DaFirstEnv() As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
' Check that environment variables can be retrieved - this also retrieve
all
' variables in one shot, improving performances
lRc = DaFirstEnv()

' Get usefull environment variables and store them in context
if lRc <> 0 then
 ' Get environment variables
 strProcArch      = DaGetEnv( "PROCESSOR_ARCHITECTURE" )
 strProcId        = DaGetEnv( "PROCESSOR_IDENTIFIER" )
 strProcLevel     = DaGetEnv( "PROCESSOR_LEVEL" )
 strProcRevision  = DaGetEnv( "PROCESSOR_REVISION" )
 strProcCount     = DaGetEnv( "NUMBER_OF_PROCESSORS" )
 strOs            = DaGetEnv( "OS" )
 strComputer      = DaGetEnv( "COMPUTERNAME" )
```

```
 ' Check that computer name is the right one
 if strComputer <> DaContext( "Computer.Name" ) then
   print "Warning: computer name does not look like the right one !"
 end if

 ' Store them in context
 DaSetContext "EnvInfo.ProcArch", strProcArch
 DaSetContext "EnvInfo.ProcId", strProcId
 DaSetContext "EnvInfo.ProcLevel", strProcLevel
 DaSetContext "EnvInfo.ProcRevision", strProcRevision
 DaSetContext "EnvInfo.ProcCount", strProcCount
 DaSetContext "EnvInfo.Os", strOs
 DaSetContext "EnvInfo.Computer", strComputer

 ' Dump
 print DaDumpContext()
end if
```

DaGetEnv()

This function returns the value of an environment variable - contained in the list buit using the **DaFirstEnv()** function - identified by its name.

Internal BASIC syntax

Function DaGetEnv(strEnv As String) As String

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strEnv:** This parameter contains the name of the environment variable as it is displayed by the operating system.

Tip:

To display the list of environment variables declared on your computer:

1 Right-click your computer icon,
2 Select the **Advanced** tab,
3 Click **Environment variables**

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
' Check that environment variables can be retrieved - this also retrieve
all
' variables in one shot, improving performances
lRc = DaFirstEnv()

' Get useful environment variables and store them in context
if lRc <> 0 then
 ' Get environment variables
 strProcArch      = DaGetEnv( "PROCESSOR_ARCHITECTURE" )
 strProcId        = DaGetEnv( "PROCESSOR_IDENTIFIER" )
 strProcLevel     = DaGetEnv( "PROCESSOR_LEVEL" )
 strProcRevision  = DaGetEnv( "PROCESSOR_REVISION" )
 strProcCount     = DaGetEnv( "NUMBER_OF_PROCESSORS" )
 strOs            = DaGetEnv( "OS" )
 strComputer      = DaGetEnv( "COMPUTERNAME" )

 ' Check that computer name is the right one
 if strComputer <> DaContext( "Computer.Name" ) then
   print "Warning: computer name does not look like the right one !"
 end if
```

```
 ' Store them in context
 DaSetContext "EnvInfo.ProcArch", strProcArch
 DaSetContext "EnvInfo.ProcId", strProcId
 DaSetContext "EnvInfo.ProcLevel", strProcLevel
 DaSetContext "EnvInfo.ProcRevision", strProcRevision
 DaSetContext "EnvInfo.ProcCount", strProcCount
 DaSetContext "EnvInfo.Os", strOs
 DaSetContext "EnvInfo.Computer", strComputer

 ' Dump
 print DaDumpContext()
end if
```

DaGetFileInfo()

This function returns the properties of a file or folder. The properties in question can be recovered directly using the **DaFile*** functions. If works in one of two ways:

- Operation on the deployment server (in the file depot)
- Operation on the deployment target

The way in which it works is determined by the value of the **file.on_server** option, set using the **DaSetOption()** function:

- If this option is set to 1, the operation is on the deployment server
- If this option is set to 0, the operation is on the deployment target.

Note:

For further information, refer to the description of the **DaSetOption()** function.

Internal BASIC syntax

Function DaGetFileInfo(strPath As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strPath**: This parameter contains the path of the file or folder concerned by the operation. If the operation is carried out on the deployment server, it is a path relative to the absolute path of the file depot. If the operation is carried out on the deployment target, it is an absolute path.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaImpersonate()

This function enables you to impersonate a user on the deployment target by simulating that user. This function can be particularly useful to simulate an administrator connection in order to install certain software on the target, for example.

Internal BASIC syntax

Function DaImpersonate(strUser As String, strPassword As String, strDomain As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strUser**: This parameter contains the name of the user for which you want to simulate a connection to the target.
- **strPassword**: This parameter contains the password of the user for which you want to simulate a connection to the target.
- **strDomain**: This parameter contains the domain of the user for which you want to simulate a connection to the target.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
' Upload whole 'install' directory
lErr = DaUpload( "install", "c:/tmp/install", "" )

' Impersonate
if lErr = 0 then lErr = DaImpersonate( "user", "password", "domain " )

' Execute setup as user, synchronous, and log errors
```

```
if lErr = 0 then
  lDummyErr = DaSetOption( "exec.synchronous", 1 )
  lDummyErr = DaSetOption( "exec.log_output", 1 )
  lDummyErr = DaSetOption( "exec.log_error", 1 )
end if
if lErr = 0 then lErr = DaExec( "c:/tmp/install/setup -i", "c:/tmp/instal
l" )

' On error, raise 'error' event
if lErr <> 0 then DaSetReturnValue "ErrorEvent"
```

DaMkDir()

This function creates a folder. It works in one of two ways:

- Creation on the deployment server (in the file depot)
- creation on the deployment target

The way in which it works is determined by the value of the **file.on_server** option, set using the **DaSetOption()** function:

- If this option is set to 1, the creation is on the deployment server
- If this option is set to 0, the creation is on the deployment target.

Note:

For further information, refer to the description of the **DaSetOption()** function.

Internal BASIC syntax

Function DaMkDir(strDirectory As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓

	Available
FINISH.DO script of a wizard	✓

Input parameters

- **strDirectory**: This parameter contains the path of the folder to create. If the creation is carried out on the deployment server, it is the path relative to the absolute path of the file depot. If the creation is on the deployment target, it is an absolute path.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaMove()

This function moves a file, a set of files or a folder. This function works in one of two ways:

- It moves a file locally to the deployment server: The source and the destination of the move are both in the deployment server.
- It moves a file locally to the deployment target: The source and the destination of the move are both on the deployment target.

The way in which it works is determined by the value of the **file.on_server** option, set using the **DaSetOption()** function:

- If this option is set to 1, the move is on the deployment server.
- If this option is set to 0, the move is on the deployment target.

Note:

For further information, refer to the description of the **DaSetOption()** function.

Note:

This function carries out the same operations as the **Move files** activity.

Internal BASIC syntax

Function DaMove(strSource As String, strDest As String, strNameFilter As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strSource**: This parameter contains the source path of the elements to move. Depending on how the function operates, it is either a path relative to the file depot (in the case of a local move to the deployment server) or an absolute path (in the case of a local move to the deployment target).
- **strDest**: This parameter contains the destination path of the elements to move. Depending on how the function operates, it is either a path relative to the file depot (in the case of a local move to the deployment server) or an absolute path (in the case of a local move to the deployment target).
- **strNameFilter**: This parameter contains wildcard characters, as used by DOS (* and **?**), used to filter the file names.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
' move old report files before getting new ones
Dim l as Long
l = DaSetOption( "file.on_server", 1 );
l = DaMove( "reports/current", "reports/old", "" )
```

DaNetIpFromName()

This function resolves the name of the target using the DNS method and returns the IP address of the target.

Internal BASIC syntax

Function DaNetIpFromName(strHost As String, pstrIP As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strHost**: This parameter contains the name of the host (deployement target).
- **pstrIP**: IP address of the target

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaNetNBTName()

This function queries the deployment target for related network information. This function is equivalent to the DOS command:

```
nbstat -A <IP address of the target>
```

Internal BASIC syntax

Function DaNetNBTName(strHost As String, lTimeout As Long, pstrComputerName As String, pstrGroup As String, pstrMAC As String) As Long

Field of application

Version: 1.0

Available
AssetCenter API
Configuration script of a field or link
"Script" type action

	Available
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strHost**: This parameter contains the name of the host (deployement target).
- **lTimeout**: This parameter contains the duration after which the operation is considered to have failed, if the target has not responded. This duration is expressed in milliseconds.
- **pstrComputerName**: Name of the target
- **pstrGroup**: Network group of the target
- **pstrMAC**: MAC address of the target

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaNetPing()

This function pings the deployment target. This enables you to test whether a computer is responding on the network.

Internal BASIC syntax

Function DaNetPing(lTimeout As Long) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lTimeout**: This parameter contains the duration after which the operation is considered to have failed, if the target has not responded. This duration is expressed in milliseconds.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim lLive As Long
' Ping with 1000ms timeout
lLive = DaNetPing( 1000 )
If lLive = 0 Then
  DaSetReturnValue "NOK"
End If
```

DaNetWakeOnLan()

This function sends a wake on LAN signal to a deployment target.

Note:

This function performs the same operations as the **Wake on LAN** activity.

Internal BASIC syntax

Function DaNetWakeOnLan(strMAC As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strMAC**: This parameter contains the MAC address of the deployment target, in hexadecimal format. The MAC address is made up of six hexadecimal pairs, which may or may not be separated (with the "-" or ":" characters).

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNetWinAddressByName()

This function resolves the name of the target using the specified method. The function returns the IP address of the target.

Internal BASIC syntax

Function DaNetWinAddressByName(strHost As String, iNameSpace As Long, pstrIP As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strHost**: This parameter contains the name of the host (deployement target).
- **iNameSpace**: This parameter specifies the resolution method to use. The possible values are the following:
  - 0: Uses the default resolution mode of the operating system
  - 1: Uses DNS resolution
  - 2: Uses WINS resolution
- **pstrIP**: This parameter contains the name of the host (deployement target).

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaNextEnv()

This function returns the name of the next environment variable in the list built using the **DaFirstEnv()** function. If the end of the list is reached, the function returns an empty character string.

Internal BASIC syntax

Function DaNextEnv() As String

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
'
' retrieve and print all environment variables from target
'

dim lRc as long
dim VarName, VarValue as String

If DaFirstEnv() <> 0 Then
  Do
    ' get next variable
    VarName = DaNextEnv()
    If VarName <> "" Then
      VarValue = DaGetEnv( VarName )
      Print VarName & "=" & VarValue
    Else
     Exit Do
    End If
  Loop
End If
```

DaNTFileCopyTo()

This function copies a file from a source folder to a destination folder, both on the deployment target.

Internal BASIC syntax

Function DaNTFileCopyTo(strSrcPath As String, strSrc As String, strDst As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓

	Available
FINISH.DO script of a wizard	✓

Input parameters

- **strSrcPath**: This parameter contains the full path of the folder containing the file to copy.
- **strSrc**: This parameter contains the name of the file to copy.
- **strDst**: This parameter contains the full path of the destination folder of the file to copy.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNTFileCreateDir()

This function recursively creates a folder on the deployment target.

Internal BASIC syntax

Function DaNTFileCreateDir(strPath As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strPath**: This parameter contains the full path of the folder on the remote computer.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNTFileDelete()

This function delete files on a deployment target.

Internal BASIC syntax

Function DaNTFileDelete(strFile As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strFile**: Path of the file to be deleted.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNTFileDeleteDir()

This function deletes a folder and its contents on the deployment target.

Internal BASIC syntax

Function DaNTFileDeleteDir(strPathConst As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strPathConst**: Path of the folder to be deleted.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNTFileDirCopyTo()

This function copies a folder and its contents to a deployment target.

Internal BASIC syntax

Function DaNTFileDirCopyTo(strSrcPath As String, strDst As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strSrcPath**: This parameter contains the path of the folder on the server.
- **strDst**: This parameter contains the path of the destination folder on the deployment target.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNTFileDirDownload()

This function copies a folder and its contents from a deployment target to the deployment server.

Internal BASIC syntax

Function DaNTFileDirDownload(strSrcPath As String, strDst As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strSrcPath**: This parameter contains the path of the folder on the deployment server.
- **strDst**: This parameter contains the path of the destination folder on the server. This path is relative to the file depot path.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNTFileDirUpload()

This function copies a folder and its contents to a deployment target.

Internal BASIC syntax

Function DaNTFileDirUpload(strSrcPath As String, strDst As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strSrcPath**: This parameter contains the path of the folder on the deployment server. This path is relative to the path of the file depot.
- **strDst** : This parameter contains the path of the destination folder on the deployment target.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNTFileDownload()

This function downloads a file from the deployment target.

Internal BASIC syntax

Function DaNTFileDownload(strSrcPath As String, strSrc As String, strDst As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strSrcPath**: This parameter contains the full path of the folder (on the target) containing the file to be downloaded.
- **strSrc**: This parameter contains the name of the file to download.
- **strDst**: This parameter contains the full path of the folder (on the server) to which the file is downloaded.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNTFileUpload()

This function uploads a file to the deployment target.

Internal BASIC syntax

Function DaNTFileUpload(strSrcPath As String, strSrc As String, strDst As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strSrcPath**: This parameter contains the full path of the folder (on the server) containing the file to be uploaded.
- **strSrc**: This parameter contains the name of the file to copy.
- **strDst**: This parameter contains the full path of the folder (on the target) to which the file is uploaded.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNTRegistryLMAddStringValue()

This function adds a **REG_SZ** or **REG_EXPAND** type entry for a key.

Internal BASIC syntax

Function DaNTRegistryLMAddStringValue(strKey As String, strString As String, strValue As String, bExpand As Long) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strKey**: This parameter contains the full name of the key containing the entry to create.
- **strString**: This parameter contains the name of the entry to be created.
- **strValue**: This parameter contains the value of the entry to be created.
- **bExpand**: If this parameter has a value of 1, the function only processes **REG_EXPAND**-type entries.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNTRegistryLMCreateKey()

This function creates recursively a key in the **HKEY_LOCAL_MACHINE** hive of the registry.

Internal BASIC syntax

Function DaNTRegistryLMCreateKey(strKey As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strKey**: This parameter contains the full path of the key in the **HKEY_LOCAL_MACHINE** hive of the registry.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNTRegistryLMDeleteKey()

This function deletes a key fro the Registry of the deployment target. All the entries of the key are deleted.

Internal BASIC syntax

Function DaNTRegistryLMDeleteKey(strKey As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strKey**: This parameter contains the full name of the key to be deleted.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNTRegistryLMDeleteValue()

This function deletes a key entry fro the Registry of the deployment target.

Internal BASIC syntax

Function DaNTRegistryLMDeleteValue(strKey As String, strString As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strKey**: This parameter contains the full name of the key containing the entry to delete.
- **strString** : ce paramètre contient le nom de l'entrée à effacer.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNTRegistryLMGetLongValue()

This function returns the value of a **REG_DWORD** type entry of a registry key on the deployment target.

Internal BASIC syntax

Function DaNTRegistryLMGetLongValue(strKey As String, strString As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strKey**: This parameter contains the full name of the key containing the entry for which you want to recover the value.
- **strString**: This parameter contains the name of the entry for which you want to recover the value.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaNTRegistryLMGetStringValue()

This function returns the value of a **REG_DWORD** or **REG_EXPAND** type entry of a key from the Registry of the deployment target.

Internal BASIC syntax

Function DaNTRegistryLMGetStringValue(strKey As String, strString As String) As String

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strKey**: This parameter contains the full name of the key containing the entry for which you want to recover the value.
- **strString**: This parameter contains the name of the entry for which you want to recover the value.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaNTRegistryLMSetLongValue()

This function sets the value of a **REG_DWORD** type entry for a key. The entry must exist in the key.

Internal BASIC syntax

Function DaNTRegistryLMSetLongValue(strKey As String, strString As String, lValue As Long) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strKey**: This parameter contains the full name of the key containing the entry for which you want to set the value.
- **strString**: This parameter contains the name of the entry for which you want to set the value.
- **lValue**: This parameter contains the value of the entry for which you want to set the value.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNTRegistryLMSetStringValue()

This function sets the value of a **REG_SZ** or **REG_EXPAND** type entry for a key. The entry must exist in the key.

Internal BASIC syntax

Function DaNTRegistryLMSetStringValue(strKey As String, strString As String, strValue As String, bExpand As Long) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strKey**: This parameter contains the full name of the key containing the entry for which you want to set the value.
- **strString**: This parameter contains the name of the entry for which you want to set the value.
- **strValue**: This parameter contains the value of the entry for which you want to set the value.
- **bExpand**: If this parameter has a value of 1, the function only processes **REG_EXPAND**-type entries.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNTServiceInstall()

This function installs an NT service on the deployment target.

Internal BASIC syntax

Function DaNTServiceInstall(strServiceName As String, strDisplayName As String, strBinaryFullName As String, bAutoStart As Long, bInteractWithDesktop As Long) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strServiceName**: This parameter contains the internal name of the service, as it will be displayed in the properties of the service (accessible via the **Properties** shortcut menu).
- **strDisplayName**: This parameter contains the name of the service as it is displayed in the main list of services.
- **strBinaryFullName**: This parameter contains the full path of the service executable on the deployment target.
- **bAutoStart**: When this parameter has a value of 1, the service is started automatically.
- **bInteractWithDesktop**: When this parameter has a value of 1, the service can interact with the operating system's desktop (message sending and user dialog boxes).

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNTServiceStart()

This function starts a named NT service on a deployment target.

Internal BASIC syntax

Function DaNTServiceStart(strServiceName As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strServiceName**: This parameter contains the name of a service to start. The name used is the internal name of the service and not the name displayed in the operating system's services console. To locate the internal name:
  1. Right-click on the name of the service.
  2. Select the **Properties** menu.
  3. The **Service name** (internal name) is displayed in the **General** tab.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNTServiceStatus()

This function returns the status of an NT service on the deployment target.

Internal BASIC syntax

Function DaNTServiceStatus(strServiceName As String) As String

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strServiceName:** This parameter contains the internal name of the service, as it is displayed in the properties of the service (accessible via the **Properties** shortcut menu).

Output parameters

The function returns one of the following values, each one corresponding to a status for the service:

- NOT FOUND
- STOPPED
- START_PENDING
- STOP_PENDING
- RUNNING
- CONTINUE_PENDING

- PAUSE_PENDING
- PAUSED

DaNTServiceStop()

This function stops the execution of an NT service named on the deployment target.

Internal BASIC syntax

Function DaNTServiceStop(strServiceName As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strServiceName**: This parameter contains the name of the service to stop. The name used is the internal name of the service and not the name displayed in the operating system's services console. To locate the internal name:
  1. Right-click on the name of the service.
  2. Select the **Properties** menu.
  3. The **Service name** (internal name) is displayed in the **General** tab.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNTServiceUninstall()

This function uninstalls an NT service on the deployment target.

Internal BASIC syntax

Function DaNTServiceUninstall(strServiceName As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strServiceName**: This parameter contains the internal name of the service, as it is displayed in the properties of the service (accessible via the **Properties** shortcut menu).

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNTWMIExecMethod()

This function enables you to use a WMI method on an object or a class.

Internal BASIC syntax

Function DaNTWMIExecMethod(strObjectPath As String, strMethod As String, strInParams As String, pstrOutParam As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strObjectPath**: Name of the object or the class.
- **strMethod**Name of the method (CIMV2 namespace).
- **strInParams**List of (input) parameters of the method
- **pstrOutParam**: Description of the parameters (output) of the method.

  For example, the result of the **Create** method applied to the WIN32_Process process will look like:

```
instance of __PARAMETERS
{
 ProcessId = 400;
 ReturnValue = 0;
};
```

Note:

The character string that is returned contains line breaks.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
lCount = DaNTWMIGetInstanceCount( "Win32_Process" )
 For l = lCount - 1 to 0 Step -1
   strName = DaNTWMIGetPropertyValue( "Win32_Process.Name", l )
   If strName = "NOTEPAD.EXE" Then
     strPID = DaNTWMIGetPropertyValue( "Win32_Process.Handle", l )
     lErr = DaNTWMIExecMethod( "Win32_Process.Handle='" & strPID & "'", "
Terminate", strDummy )
   End If
 Next l
```

Ce script arrête l'ensemble des processus NOTEPAD.EXE grâce à la méthode **Terminate**.

DaNTWMIExecQuery()

This function enables you to enumerate the WMI by using a WMI query in WQL language.

Internal BASIC syntax

Function DaNTWMIExecQuery(strQuery As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	

	Available
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strQuery**: WQL query

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Example

```
lErr = DaNTWMIExecQuery( "select * from Win32_NTLogEvent WHERE Logfile='S
ecurity'" )
While DaNTWMINextItem()
 print DaNTWMIGetCurrentPropertyValue( "Message", i )
Wend
lErr = DaNTWMIResetEnumeration()
```

Ce script énumère l'ensemble des événements du journal d'événements de la sécurité NT et recherche l'événement spécifié.

DaNTWMIGetCurrentArrayValue()

This function is used with the **DANTWMIExecQuery** function. It recovers the values of properties (table type) of the enumerated object.

The function returns the list of properties of the value, separated by a character defined in the **strSeparator**.

Internal BASIC syntax

Function DaNTWMIGetCurrentArrayValue(strProperty As String, strSeparator As String) As String

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strProperty**: This parameter contains the name of property (of the WMI object) for which you want to recover the value.
- **strSeparator**: This parameter contains the character used as separator for the values of the property.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

Dim lErr As Long lErr = DaNTWMIExecQuery("SELECT * FROM Win32_NetworkAdapterConfiguration WHERE IPEnabled=TRUE") While

DaNTWMINextItem() print DaNTWMIGetCurrentArrayValue("IPAddress", ";") Wend

DaNTWMIGetCurrentPropertyValue()

This function is used in relation to the **DANTWMIExecQuery** function. It recovers the property values of the enumerated object.

Note:

Use the **DaNTWMIGetCurrentArrayValue()** function if you want to retrieve the values of an array property.

Internal BASIC syntax

Function DaNTWMIGetCurrentPropertyValue(strProperty As String) As Variant

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strProperty**: This parameter contains the name of property (of the WMI object) for which you want to recover the value.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaNTWMIGetInstanceCount()

This function returns the number of objects of a given class (for example, the number of hard drives, CPUs, etc. of a computer).

Internal BASIC syntax

Function DaNTWMIGetInstanceCount(strShortName As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strShortName**: This parameter contains the name of the object for which you want to count the number of instances. The available objects are listed in the table below:

Property
Win32_Processor
Win32_Processor
Win32_PhysicalMemory
Win32_OperatingSystem
Win32_DiskDrive
Win32_LogicalDisk
Win32_Operatingsystem
Win32_Operatingsystem.BuildNumber
Win32_NetworkAdapter
Win32_NetworkAdapterConfiguration

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Example

The following example returns the number of hard drives on the computer:

```
Dim iHDCount as Integer
iHDCount = DaNTWMIGetInstanceCount("Win32_LogicalDisk")
print "Number of hard drives = " & iHDCount
```

DaNTWMIGetPropertyValue()

This function recovers the value of a property from a WMI object.

Internal BASIC syntax

Function DaNTWMIGetPropertyValue(strShortName As String, idx As Long) As Variant

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strShortName**: This parameter contains the name of the property for which you want to recover the value. The syntax used is the following:

```
<object>.<property>
```

You can also use the aliases listed in the table below:

Alias	Referenced property
CPUType	Win32_Processor.Family
CPUInternal	Win32_Processor.Caption
CPUSpeed	Win32_Processor.CurrentClockSpeed
MemorySize	Win32_PhysicalMemory.Capacity
OSMemorySize	Win32_OperatingSystem.TotalVisibleMemorySize
TotalDiskSize	Wind32_DiskDrive.Size
DiskSize	Win32_LogicalDisk.Size
DiskFreeSpace	Win32_LogicalDisk.FreeSpace
OperatingSystem	Win32_Operatingsystem.Caption
OSServiceLevel	Win32_Operatingsystem.ServicePackMajorVersion
OSBuildNumber	Win32_Operatingsystem.BuildNumber
PhysAddress	Win32_NetworkAdapter.MACAddress
DNSHostName	Win32_NetworkAdapterConfiguration.DNSHostName
DNSDomain	Win32_NetworkAdapterConfiguration.DNSDomain

- **idx**: This parameter contains the index of the object concerned by the operation. If, for example, you have several hard drives, each of them will have an index number.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following example recovers the clock speed of the computer's CPU:

```
Dim vSpeed as Variant
vSpeed = DaNTWMIGetPropertyValue("CPUSPeed", 0)
print "Speed = " & vSpeed
```

DaNTWMIGetTotalPropertiesValue()

This function returns the sum of the values of a property for all objects with this property. In this way you can recover the disk space available for all the logical drives of the computer.

Internal BASIC syntax

Function DaNTWMIGetTotalPropertiesValue(strShortName As String) As Variant

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓

	Available
FINISH.DO script of a wizard	✓

Input parameters

- **strShortName:** This parameter contains the name of the property for which you want to recover the value. The syntax used is the following:

```
<object>.<property>
```

You can also use the aliases listed in the table below:

Alias	Referenced property
CPUType	Win32_Processor.Family
CPUInternal	Win32_Processor.Caption
CPUSpeed	Win32_Processor.CurrentClockSpeed
MemorySize	Win32_PhysicalMemory.Capacity
OSMemorySize	Win32_OperatingSystem.TotalVisibleMemorySize
TotalDiskSize	Wind32_DiskDrive.Size
DiskSize	Win32_LogicalDisk.Size
DiskFreeSpace	Win32_LogicalDisk.FreeSpace
OperatingSystem	Win32_Operatingsystem.Caption
OSServiceLevel	Win32_Operatingsystem.ServicePackMajorVersion
OSBuildNumber	Win32_Operatingsystem.BuildNumber
PhysAddress	Win32_NetworkAdapter.MACAddress
DNSHostName	Win32_NetworkAdapterConfiguration.DNSHostName
DNSDomain	Win32_NetworkAdapterConfiguration.DNSDomain

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following example returns the total capacity of all the logical drives of the computer:

```
Dim vCapacity as Variant
vCapacity = DaNTWMIGetTotalPropertiesValue("WIN32_PhysicalMemory.Capacity
")
print "Physical memory = " & vCapacity
```

DaNTWMINextItem()

This function is used in relation to the **DANTWMIExecQuery** function. It enumerates the set of elements returned by a WQL query.

Internal BASIC syntax

Function DaNTWMINextItem() As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaNTWMIResetEnumeration()

This function is used in relation to the **DANTWMIExecQuery** function. It reinitiates the enumeration being opened and frees up the corresponding memory.

Internal BASIC syntax

Function DaNTWMIResetEnumeration() As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

DaRegCreateKey()

This function creates a key in the Registry of the deployment target.

Internal BASIC syntax

Function DaRegCreateKey(strKey As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strKey**: This parameter contains the full name of the key to be created.

  Example:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Peregrine Systems\Automated Desktop Admini
stration\6.0.0
```

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaRegDeleteKey()

This function deletes a key from the Registry of the deployment target.

Internal BASIC syntax

Function DaRegDeleteKey(strKey As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strKey**: This parameter contains the full name of the key to be deleted.

```
HKEY_LOCAL_MACHINE\SOFTWARE\Peregrine Systems\Automated Desktop Admini
stration\6.0.0
```

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaRegExec()

This function executes several operations (defined in one script) on the Registry of the target. The script uses the same syntax as the **.reg** files of the operating system.

Note:

This function corresponds to the **Registry** type activity.

Internal BASIC syntax

Function DaRegExec(strScript As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strScript**: This parameters contains the script to execute on the Registry.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaRegGetValue()

This function returns the value of a registry key from the deployment target.

Internal BASIC syntax

Function DaRegGetValue(strKey As String, strValue As String) As Variant

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strKey**: This parameter contains the full name of the key concerned by the operation.
- **strValue**: This parameter contains the name of the entry of the key for which you want to recover the value.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaRegOutputValue()

This function recovers the value of the contents of **DaRegOutput**.

Internal BASIC syntax

Function DaRegOutputValue(strKey As String, strValue As String) As Variant

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strKey**: This parameter contains the full name of the key concerned by the operation. If this parameter is empty, any key will be used.
- **strValue**: This parameter contains the string to convert. If the parameter is empty, the default value will be used.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaRegSetValue()

This function defines the value for a key entry in the Registry of the deployment target.

Internal BASIC syntax

Function DaRegSetValue(strKey As String, strValue As String, vValue As Variant) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strKey**: This parameter contains the full name of the key for which you want to set the value.
- **strValue**: This parameter contains the name of the entry to be defined.
- **vValue**: This parameter contains the value to assign to the entry. If you do not specify a value for this parameter, the current value for the entry is deleted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaRegStrValue()

This function converts a value expressed as a variant to an expression compatible with the registry (as a string).

Internal BASIC syntax

Function DaRegStrValue(vValue As Variant) As String

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **vValue**: This parameter contains the value to convert

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaRegVarValue()

This function converts a string type value to a variant.

Internal BASIC syntax

Function DaRegVarValue(strValue As String) As Variant

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strValue**: This parameter contains the string to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaRename()

This function renames a file or a folder. It works in one of two ways:

- renaming on the deployment server (in the file depot)
- renaming of the deployment server

The way in which it works is determined by the value of the **file.on_server**option, set using the **DaSetOption()** function:

- If this option is set to 1, the renaming is on the deployment server.
- If this option is set to 0, the renaming is on the deployment target.

Note:

For further information, refer to the description of the **DaSetOption()** function.

Internal BASIC syntax

Function DaRename(strSource As String, strDest As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strSource**: This parameter contains the path of the source file or folder to rename. If the operation is carried out on the deployment server, it is a path relative to the absolute path of the file depot. If the operation is carried out on the deployment target, it is an absolute path.

- **strDest**: This parameter contains the new name of the file or folder if the form of a full path. If the operation is carried out on the deployment server, it is path relative to absolute path in the file depot. If the operation is carried out on the deployment target, it is an absolute path.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaReturnValue()

This function returns the name of the output event of the current activity of the deployment workflow.

Internal BASIC syntax

Function DaReturnValue() As String

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
' Set a return value if not already set
If DaReturnValue = "" Then
  SetDaReturnValue( "MyEvent" )
End If
```

DaRmDir()

This function deletes a folder and its contents. It works in one of two ways:

- Deletion on the deployment server (in the file depot)
- deletion on the deployment target

This way in which it works is determined by the value of the **file.on_server** option, set set using the **DaSetOption()** function:

- if this option is set to 1, the deletion is on the deployment server
- If this option is set to 0, the deletion is on the deployment target

Note:

For further information, refer to the description of the **DaSetOption()** function.

Internal BASIC syntax

Function DaRmDir(strDirectory As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strDirectory**: This parameter contains the path of the folder to delete. If the deletion is carried out on the deployment server, it is the path relative to the absolute path of the file depot. If the deletion is on the deployment target, it is an absolute path.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaSendMail()

This function sends an e-mail. Since the operation is performed by the deployment server, all the configuration parameters associated with sending an e-mail must be configured on the server.

The following protocols are supported:

- SMTP: The addresses are as follows:

```
SMTP:[name@address.domain]
```

- MAPI: The addresses are as following:

```
MAPI:[name of the mail box]
```

- VIM: The addresses are as follows:

```
VIM:[name/domain]
```

- AM: internal messaging system (corresponding to the employees referenced in the database). The addresses are as follows:

```
AM:[login of the recipient]
```

Note:

In the case of multiple recipients, each address is separated by a comma (",").

Internal BASIC syntax

Function DaSendMail(strTo As String, strCc As String, strBcc As String, strSubject As String, strMessage As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strTo**: This parameter contains the list of addresses of recipients of a message.
- **strCc**: This parameter contains the list of addresses of recipients to receive copies of a message.

- **strBcc**: This parameter contains the list of addresses of recipients to receive blind carbon copies (they do not appear in the list of recipients) of a message.
- **strSubject**: This parameter contains the title of the message.
- **strMessage**: This parameter contains the body of the message.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaSetContext()

This function defines a new context in the list of contexts available for a deployment workflow.

Internal BASIC syntax

Function DaSetContext(strField As String, vValue As Variant)

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strField**: This parameter contains the name of the context you define.
- **vValue**: This parameter contains the value of the context that you define.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
' Check that environment variables can be retrieved - this also retrieve
all
' variables in one shot, improving performances
lRc = DaFirstEnv()

' Get usefull environment variables and store them in context
if lRc <> 0 then
 ' Get environment variables
 strProcArch      = DaGetEnv( "PROCESSOR_ARCHITECTURE" )
 strProcId        = DaGetEnv( "PROCESSOR_IDENTIFIER" )
 strProcLevel     = DaGetEnv( "PROCESSOR_LEVEL" )
 strProcRevision  = DaGetEnv( "PROCESSOR_REVISION" )
 strProcCount     = DaGetEnv( "NUMBER_OF_PROCESSORS" )
 strOs            = DaGetEnv( "OS" )
 strComputer      = DaGetEnv( "COMPUTERNAME" )

 ' Check that computer name is the right one
 if strComputer <> DaContext( "Computer.Name" ) then
   print "Warning: computer name does not look like the right one !"
 end if

 ' Store them in context
 DaSetContext "EnvInfo.ProcArch", strProcArch
 DaSetContext "EnvInfo.ProcId", strProcId
 DaSetContext "EnvInfo.ProcLevel", strProcLevel
 DaSetContext "EnvInfo.ProcRevision", strProcRevision
 DaSetContext "EnvInfo.ProcCount", strProcCount
 DaSetContext "EnvInfo.Os", strOs
 DaSetContext "EnvInfo.Computer", strComputer
```

```
 ' Dump
 print DaDumpContext()
end if
```

DaSetOption()

This function enables you to define options for certain other functions. The functions concerned are:

- All **File management** functions
- All **Remote execution** functions

Internal BASIC syntax

Function DaSetOption(strOption As String, lValue As Long) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strOption**: This parameter contains the name of the option on which you want to intervene. The list of available options is as follows:
  - **find.showdir**: When you select this option, the **DaFind()** function includes the folders in the returned list.
  - **file.resume**: When you select this option, the functions that transfer files - **DaUpload()** and **DaDownload()** - attempt to continue the transfer from where it was when the transfer was interrupted.

- **file.mkdir**: When you select this option, the **File management** functions authorize you to create folders.
- **file.overwrite**: When you select this option, the **File management** functions authorize you to overwrite files with the same name in the target.
- **file.recursive**: When you select this option, the **File management** functions become recursive.
- **file.on_server**: When you select this option, the **DaMove()**, **DaCopy()** and **DaDelete()** functions operate on the file depot of the deployment server and not on the deployment target.
- **file.force**: When this option is selected, the file is forced. For example, when associated to the **DaDelete** function, this option enables you to delete a file in read-only.
- **exec.synchronous**: When you select this option, the execution of a program on the deployment target is synchronous. All later executions wait until the earlier executions have finished.
- **exec.log_output**: When you select this option, the messages sent by the program during its execution on the target are recorded in a log file.
- **exec.log_error**: When you select this option, the messages resulting from a program-execution error on the target are recorded in a log file.
- **exec.force_visibility**: When you execute a program with a graphical interface, the operating system decides whether or not to display this graphical interface. If this option is selected, the display is determined by the value of the **exec.visibility** option.
- **exec.visibility**: When you select this option, the execution of a program on the target causes the program's graphical interface to be displayed on the target. This option is used in conjunction with the **exec.force_visibility** option.

- **lValue** : This parameter can take one of two values:
  - 0: the option is not selected
  - 1: The option is selected

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
' Further invocations of DaUpload will not be recursive
lErr = DaSetOption( 'upload.recursive', 0 )
```

DaSetReturnValue()

This function enables you to change the event triggered at the end of the execution of a **Basic Script** type activity. The deployment workflow continues by using the transition associated with the event defined by the **strValue** parameter.

Note:

The activity's output event is referred to by its name and not by its title.

Internal BASIC syntax

Function DaSetReturnValue(strValue As String)

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strValue**: This parameter contains the name of the event triggered at the end of execution of the Basic script.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
dim errCode as long

' call a function that may return error code
errCode = DaFind( "c:\images", "*.gif" )

If errCode <> 0 Then
  DaSetReturnValue( "Error" )
End If
```

Date()

Returns the current system date.

Internal BASIC syntax

Function Date() As Date

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DateSerial()

This function returns a date formatted according to the **iYear**, **iMonth** and **iDay** parameters.

Internal BASIC syntax

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

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓

	Available
FINISH.DO script of a wizard	✓

Input parameters

- **iYear**: Year. If the value is between 0 and 99, this parameter describes the years from 1900 to 1999. For all other years, you must specify the four digits (e.g. 1800).
- **iMonth**: Month.
- **iDay**: Day.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

Each of these parameters can be set to a numeric expression representing a number of days, months or years. For example:

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

Returns the value:

```
1989/1/7
```

When the value of a parameter is out of the expected range (i.e. 1-31 for days, 1-12 for months, etc.), the function returns an empty date.

DateValue()

This function returns the date portions of a "Date+Time" value.

Internal BASIC syntax

Function DateValue(tmDate As Date) As Date

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **tmDate**: "Date+Time" format date.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following example:

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

Returns the value:

```
1999/09/24
```

DaTrackingDelete()

This function deletes a record in the tracking table.

Internal BASIC syntax

Function DaTrackingDelete(strDomain As String, strCategory As String, strSection As String, strName As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strDomain**: This parameter contains the domain of the record in the tracking table.
- **strCategory**: This parameter contains the category of the record in the tracking table.
- **strSection**: This parameter contains the section of the record in the tracking table.
- **strName**: This parameter contains the name of the record in the tracking table.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaTrackingGet()

This function recovers the value of a field from the tracking table.

Internal BASIC syntax

Function DaTrackingGet(strDomain As String, strCategory As String, strSection As String, strName As String, strField As String) As String

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strDomain**: This parameter contains the domain of the record in the tracking table.
- **strCategory**: This parameter contains the category of the record in the tracking table.
- **strSection**: This parameter contains the section of the record in the tracking table.

- **strName**: This parameter contains the name of the record in the tracking table.
- **strField**: This parameter contains the name of the field in the tracking table.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaTrackingSet()

This function populates a field in the tracking table. If the record does not exist, it is created.

Internal BASIC syntax

Function DaTrackingSet(strDomain As String, strCategory As String, strSection As String, strName As String, strField As String, strValue As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strDomain**: This parameter contains the domain of the record in the tracking table.
- **strCategory**: This parameter contains the category of the record in the tracking table.
- **strSection**: This parameter contains the section of the record in the tracking table.
- **strName**: This parameter contains the name of the record in the tracking table.
- **strField**: This parameter contains the name of the field in the tracking table.
- **strValue**: This parameter contains the value of the field in the tracking table.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaTrackingTest()

This function returns TRUE if there is a record with these properties in the tracking table (amDaTracking).

Internal BASIC syntax

Function DaTrackingTest(strDomain As String, strCategory As String, strSection As String, strName As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strDomain**: This parameter contains the domain of the record in the tracking table.
- **strCategory**: This parameter contains the category of the record in the tracking table.
- **strSection**: This parameter contains the section of the record in the tracking table.
- **strName**: This parameter contains the name of the record in the tracking table.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

DaUpload()

This function copies a file, folder or set of files to the deployment server. The source files must be stored in the file depot of the deployment server.

You can also configure the behavior of this function by setting certain options using the **DaSetOption()** option. A full list of this option is included in the description of the **DaSetOption()** function.

Note:

This function carries out the same operations as the **Upload files** activity.

Internal BASIC syntax

Function DaUpload(strSrcPath As String, strDstPath As String, strSrcNameFilter As String) As Long

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strSrcPath**: This parameter contains the path of the files to copy on the deployment target. It is a relative path depending on the path of the file depot on the deployment server. For example, if the absolute path of the depot is:

  ```
  c:\files\depot
  ```

  and you want to copy a file with the following absolute path:

  ```
  c:\files\depot\software\antivirus\update\file.exe
  ```

  then this parameter will have the following value:

  ```
  software\antivirus\update
  ```

- **strDstPath**: This parameter contains the absolute path of the files copied on the deployment target. In the previous example this is:

```
c:\program files\antivirus\update
```

- **strSrcNameFilter**: This parameter contains wildcard characters, as used by DOS (* and ?), used to filter the file names.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
' Upload whole 'install' directory
lErr = DaUpload( "install", "c:/tmp/install", "" )

' Impersonate
if lErr = 0 then lErr = DaImpersonate( "user", "password", "domain " )

' Execute setup as user, synchronous, and log errors
if lErr = 0 then
  lDummyErr = DaSetOption( "exec.synchronous", 1 )
  lDummyErr = DaSetOption( "exec.log_output", 1 )
  lDummyErr = DaSetOption( "exec.log_error", 1 )
end if
if lErr = 0 then lErr = DaExec( "c:/tmp/install/setup -i", "c:/tmp/instal
l" )

' On error, raise 'error' event
if lErr <> 0 then DaSetReturnValue "ErrorEvent"
```

DaWait()

This function pause a deployment workflow for a period of time expressed in seconds.

Internal BASIC syntax

Function DaWait(flSecDelay As Double)

Field of application

Version: 1.0

	Available
AssetCenter API	
Configuration script of a field or link	
"Script" type action	
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **flSecDelay**: This parameter defines the length of the pause in seconds.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
' Waits half a second, prints a message and waits 10 seconds
DaWait (0.5)
print "Waiting again..."
DaWait (10)
```

Day()

Returns the day contained in the **tmDate** parameter.

Internal BASIC syntax

Function Day(tmDate As Date) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **tmDate**: Parameter in Date+Time format to be processed.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

EscapeSeparators()

Prefixes one or more separator characters with an escape character.

Internal BASIC syntax

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

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strSource**: Character string to process.
- **strSeparators**: List of separators to be prefixed. If you want to declare several separators, you must separate them with the escape character (indicated in the **strEscChar** parameter.
- **strEscChar**: Escape character. It will be used to prefix all separators in **strSeparators**.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyStr
  MyStr=EscapeSeparators("you|me|you,me|you", "|\,", "\") :'Returns "you\
|me\|you\,me\|you"
```

ExeDir()

This function returns the full path of the executable.

Internal BASIC syntax

Function ExeDir() As String

Field of application

Version: 3.60

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

Exp()

Returns the exponent of a number.

Internal BASIC syntax

Function Exp(dValue As Double) As Double

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dValue**: Number whose exponent you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

ExtractValue()

Extracts from a string the values delimited by a separator. The recovered value is deleted from the source string. This operation takes into account a possible escape character. If the separator is not found in the source string, the whole string is returned and the source string is deleted in full.

Internal BASIC syntax

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

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **pstrData**: Source string to be processed.
- **strSeparator**: Character used as separator in the source string.
- **strEscChar**: Escape character. If this character prefixes the separator, it will be ignored.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyStr
  MyStr=ExtractValue("you,me", ",", "\") :'Returns "you" and leaves "me"
in the source string
  MyStr=ExtractValue(",you,me", ",", "\") :'Returns "" and leaves "you,me
" in the source string
  MyStr=ExtractValue("you", ",", "\") :'Returns "you" and leaves "" in th
e source string
  MyStr=ExtractValue("you\,me", ",", "\") :'Returns "you\,me" and leaves
"" in the source string
  MyStr=ExtractValue("you\,me", ",", "") :'Returns "you\" and leaves "me"
 in the source string
  RetVal=""
```

FileCopy()

Copies a file or a folder.

Internal BASIC syntax

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

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strSource**: Full path of the file or directory to copy.
- **strDest**: Full path of the target file or directory.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

FileDateTime()

Returns the time and date of a file as a Long.

Internal BASIC syntax

Function FileDateTime(strFileName As String) As Date

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strFileName**: Full path name of the file concerned by the operation.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

FileExists()

This function tests for the existence of a file.

Internal BASIC syntax

Function FileExists(strFileName As String) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strFileName**: This parameter contains the full path of the file you want to test for.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

FileLen()

Returns the size of a file.

Internal BASIC syntax

Function FileLen(strFileName As String) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strFileName**: Full path name of the file concerned by the operation.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Fix()

Returns the integer portion of a number (first greatest integer in the case of a negative number).

Internal BASIC syntax

Function Fix(dValue As Double) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dValue**: Number whose integer portion you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

FormatDate()

Formats a date according to the expression contained in the **strFormat** parameter.

Internal BASIC syntax

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

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **tmFormat**: Date to be formatted.
- **strFormat**: Expression containing the formatting instructions.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following example of code shows how to format a date:

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

FormatResString()

This function processes a source string, replacing the variable $1, $2, $3, $4, and $5 with the strings passed in the **strParamOne**, **strParamTwo**, **strParamThree**, **strParamFour**, and **strParamFive** parameters.

Internal BASIC syntax

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

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strResString**: Source string to be processed.
- **strParamOne**: Replacement string of variable $1.
- **strParamTwo**: Replacement string of variable $2.
- **strParamThree**: Replacement string of variable $3.
- **strParamFour**: Replacement string of variable $4.
- **strParamFive**: Replacement string of variable $5.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following example:

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

returns "Iyouheweyouthey".

FV()

This function returns the future amount of an annuity based on constant and periodic payments, with a set interest rate.

Internal BASIC syntax

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

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dblRate**: This parameter indicates the interest rate per date of payment. For example, the rate per date of payment for a loan with a 6% annual interest rate, paid back by monthly dates of payment, would be:

```
0.06/12=0.005 or 0.5%
```

- **iNper**: This parameter contains the total number of dates of payment for the financial operation.
- **dblPmt**: This parameter indicates the amount of the payment to be made at each date of payment. The payment generally includes both principal and interest.
- **dblPV**: This parameter contains the actual value (or overall sum) for a series of payments to be made in the future.
- **iType**: This parameter indicates the payment deadline. It can have one of the following values:
  - **0** if the payments are due in arrears (i.e. at the end of the period)
  - **1** if the payments are due in advance (i.e. at the start of the period)

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

Note:

- The **Rate** and **Nper** parameters must be calculated using payments expressed in the same units.
- Amounts paid (expressed in particular by the **Pmt** parameter) are represented by negative numbers. Sums received are represented by positive numbers.

GetListItem()

Returns the **lNb**th portion of a string delimited by separators.

Internal BASIC syntax

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

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strFrom**: Source string to be processed.
- **strSep**: Character used as separator in the source string.
- **lNb**: Position of the string to recover.
- **strEscChar**: Escape character. If this character prefixes a separator, it will be ignored.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following example:

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

returns "is".

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

returns "a".

Hex()

Returns the hexadecimal value of a decimal parameter.

Internal BASIC syntax

Function Hex(dValue As Double) As String

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dValue**: Decimal number whose hexadecimal value you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Hour()

Returns the hour value contained in the **tmTime** parameter.

Internal BASIC syntax

Function Hour(tmTime As Date) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **tmTime**: Parameter in Date+Time format to be processed.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

InStr()

Returns the character position of the first occurrence of a string within a string.

Internal BASIC syntax

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

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **iPosition**: Starting point of the search. This parameter is not optional and must be a valid positive integer no greater than 65,535.
- **strSource**: String in which the search is performed.
- **strPattern**: String to search.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

Int()

Returns the integer portion of a number (first lesser than integer in the case of a negative number).

Internal BASIC syntax

Function Int(dValue As Double) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dValue**: Number whose integer portion you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

IPMT()

This function returns the amount of interest for an given date of payment of an annuity.

Internal BASIC syntax

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

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dblRate**: This parameter indicates the interest rate per date of payment. For example, the rate per date of payment for a loan with a 6% annual interest rate, paid back by monthly dates of payment, would be:

```
0.06/12=0.005 or 0.5%
```

- **iPer**: This parameter indicates the period for the calculation, between 1 and the value of the **Nper** parameter.
- **iNper**: This parameter contains the total number of dates of payment for the financial operation.
- **dblPV**: This parameter contains the actual value (or overall sum) for a series of payments to be made in the future.
- **dblFV**: This parameter contains the future value or the balance that you want to obtain after having paid the final date of payment. In general, and particularly when reimbursing a loan, this parameter is set to "0". In effect, once you have made all the dates of payment, the value of the loan is nil.
- **iType**: This parameter indicates the payment deadline. It can have one of the following values:
  - **0** if the payments are due in arrears (i.e. at the end of the period)
  - **1** if the payments are due in advance (i.e. at the start of the period)

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

Note:

- The **Rate** and **Nper** parameters must be calculated using payments expressed in the same units.
- Amounts paid (expressed in particular by the **Pmt** parameter) are represented by negative numbers. Sums received are represented by positive numbers.

IsNumeric()

This function enables you to determine whether a character string contains a numeric value.

Internal BASIC syntax

Function IsNumeric(strString As String) As Long

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓

	Available
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strString**: This parameter contains the character string to analyze.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Kill()

Deletes a file.

Internal BASIC syntax

Function Kill(strKilledFile As String) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓

	Available
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strKilledFile**: Full path of the file concerned by the operation.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

LCase()

Returns a string in which all letters of the string parameter have been converted to lower case.

Internal BASIC syntax

Function LCase(strString As String) As String

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strString:** Character string to convert to lowercase.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
' This example uses the LTrim and RTrim functions to strip leading ' and
trailing spaces, respectively, from a string variable.
' It uses the Trim function alone to strip both types of spaces.
' LCase and UCase are also shown in this example as well as the use
' of nested function calls

Dim strString as String
Dim strTrimString as String
  strString = "  <-Trim->  " :' Initialize string.
  strTrimString = LTrim(strString) :' strTrimString = "<-Trim->  ".
  strTrimString = LCase(RTrim(strString)) :' strTrimString = "  <-trim->"
.
  strTrimString = LTrim(RTrim(strString)) :' strTrimString = "<-Trim->".
  ' Using the Trim function alone achieves the same result.
  strTrimString = UCase(Trim(strString)) :' strTrimString = "<-TRIM->".
  RetVal= "|" & strTrimString & "|"
```

Left()

Returns the left most iNumber characters of a string parameter.

Internal BASIC syntax

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

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strString**: Character string to process.
- **iNumber**: Number of characters to return.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim lWord, strMsg, rWord, iPos :' Declare variables.
  strMsg = "Left() Test."
  iPos = InStr(1, strMsg, " ") :' Find space.
  lWord = Left(strMsg, iPos - 1) :' Get left word.
  rWord = Right(strMsg, Len(strMsg) - iPos) :' Get right word.
  strMsg=rWord+lWord :' And swap them
  RetVal=strMsg
```

LeftPart()

Extracts the portion of a string to the left of the separator specified in the **strSep** parameter.

The search for the separator is performed from left to right.

The search can be made case sensitive using the **bCaseSensitive** parameter.

Internal BASIC syntax

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

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strFrom**: Source string to be processed.
- **strSep**: Character used as separator in the source string.
- **bCaseSensitive**: Depending on this parameter, the search is case sensitive (=1) or not (=0).

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

These examples illustrate use of the **LeftPart**, **LeftPartFromRight**, **RightPart**, and **RightPartFromLeft** functions on the same string: "This_is_a_test":

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

Returns "This".

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

Returns "This_is_a".

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

Returns "test".

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

Returns "is_a_test".

LeftPartFromRight()

Extracts the portion of a string to the left of the separator specified in the **strSep** parameter.

The search for the separator is performed from right to left.

The search can be made case sensitive using the **bCaseSensitive** parameter.

Internal BASIC syntax

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

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strFrom**: Source string to be processed.
- **strSep**: Character used as separator in the source string.
- **bCaseSensitive**: Depending on this parameter, the search is case sensitive (=1) or not (=0).

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

These examples illustrate use of the **LeftPart**, **LeftPartFromRight**, **RightPart**, and **RightPartFromLeft** functions on the same string: "This_is_a_test":

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

Returns "This".

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

Returns "This_is_a".

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

Returns "test".

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

Returns "is_a_test".

Len()

Returns the number of characters in a string or a variant.

Internal BASIC syntax

Function Len(vValue As Variant) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **vValue**: Variant concerned by the operation.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim strTest as String
Dim iLength as Integer
  strTest = "Peregrine Systems"
  iLength = Len(strTest) :'The value of iLength is 17
  RetVal=iLength
```

LocalToBasicDate()

This function converts a string format date (as displayed in Windows Control Panel) to a Basic format date .

Internal BASIC syntax

Function LocalToBasicDate(strDateLocal As String) As String

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strDateLocal**: Date as string to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

LocalToBasicTime()

This function converts a string format time (as displayed in Windows Control Panel) to a Basic format time.

Internal BASIC syntax

Function LocalToBasicTime(strTimeLocal As String) As String

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strTimeLocal**: Time in string format to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

LocalToBasicTimeStamp()

This function converts a Date+Time in string format (as displayed in Windows Control Panel) to a Date+Time in Basic format.

Internal BASIC syntax

Function LocalToBasicTimeStamp(strTSLocal As String) As String

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strTSLocal**: Date+Time in string format to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

LocalToUTCDate()

This function converts a date in "Date+Time" format to a UTC format date (time-zone independent).

Internal BASIC syntax

Function LocalToUTCDate(tmLocal As Date) As Date

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **tmLocal**: "Date+Time" format date.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Log()

Returns the natural log of a number.

Internal BASIC syntax

Function Log(dValue As Double) As Double

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dValue**: Number whose logarithm you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

LTrim()

Removes all leading spaces in a string.

Internal BASIC syntax

Function LTrim(strString As String) As String

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strString**: Character string to process.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
' This example uses the LTrim and RTrim functions to strip leading ' and
trailing spaces, respectively, from a string variable.
' It uses the Trim function alone to strip both types of spaces.
' LCase and UCase are also shown in this example as well as the use
' of nested function calls

Dim strString as String
Dim strTrimString as String
  strString = "  <-Trim->  " :' Initialize string.
  strTrimString = LTrim(strString) :' strTrimString = "<-Trim->  ".
  strTrimString = LCase(RTrim(strString)) :' strTrimString = "  <-trim->"
.
  strTrimString = LTrim(RTrim(strString)) :' strTrimString = "<-Trim->".
  ' Using the Trim function alone achieves the same result.
  strTrimString = UCase(Trim(strString)) :' strTrimString = "<-TRIM->".
  RetVal= "|" & strTrimString & "|"
```

MakeInvertBool()

This function returns an inverse Boolean; (0 becomes 1, all other numbers become 0).

Internal BASIC syntax

Function MakeInvertBool(lValue As Long) As Long

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lValue**: Number concerned by the operation.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyValue
  MyValue=MakeInvertBool(0) :'Returns 1
  MyValue=MakeInvertBool(1) :'Returns 0
  MyValue=MakeInvertBool(254) :'Returns 0
```

Mid()

Returns a substring within a string.

Internal BASIC syntax

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

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strString**: String concerned by the operation.
- **iStart**: Start position of the string to extract from within strString.
- **iLen**: Length of the string to extract.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim strTest as String
  strTest="One Two Three" :' Defines the test string
  strTest=Mid(strTest,5,3) :' strTest="Two"
  RetVal=strTest
```

Minute()

Returns the number of minutes contained in the time expressed in the **tmTime** parameter.

Internal BASIC syntax

Function Minute(tmTime As Date) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **tmTime**: Parameter in Date+Time format to be processed.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim strMinute
  strMinute=Minute(Date())
  RetVal=strMinute :'Returns the number of minutes elapsed in the current
 hour, for example "45" if the time is 15:45:30
```

MkDir()

Creates a new directory.

Internal BASIC syntax

Function MkDir(strMkDirectory As String) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strMkDirectory**: Full path of the directory to create.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Month()

Returns the month contained in the date expressed in the **tmDate** parameter.

Internal BASIC syntax

Function Month(tmDate As Date) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **tmDate**: Parameter in Date+Time format to be processed.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim strMonth
  strMonth=Month(Date())
  RetVal=strMonth :'Returns the current month
```

Name()

Changes the name of file.

Internal BASIC syntax

Function Name(strSource As String, strDest As String)

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strSource**: Full path of the file to rename.
- **strDest**: New file name.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Now()

Returns the current date and time.

Internal BASIC syntax

Function Now() As Date

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

NPER()

This function returns the number of payments of an annuity based on constant and periodic payments, and at a constant interest rate.

Internal BASIC syntax

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

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dblRate**: This parameter indicates the interest rate per date of payment. For example, the rate per date of payment for a loan with a 6% annual interest rate, paid back by monthly dates of payment, would be:

```
0.06/12=0.005 or 0.5%
```

- **dblPmt**: This parameter indicates the amount of the payment to be made at each date of payment. The payment generally includes both principal and interest.
- **dblPV**: This parameter contains the actual value (or overall sum) for a series of payments to be made in the future.
- **dblFV**: This parameter contains the future value or the balance that you want to obtain after having paid the final date of payment. In general, and

particularly when reimbursing a loan, this parameter is set to "0". In effect, once you have made all the dates of payment, the value of the loan is nil.

- **iType**: This parameter indicates the payment deadline. It can have one of the following values:
  - **0** if the payments are due in arrears (i.e. at the end of the period)
  - **1** if the payments are due in advance (i.e. at the start of the period)

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

Note:

Amounts paid (expressed in particular by the **Pmt** parameter) are represented by negative numbers. Sums received are represented by positive numbers.

Oct()

Returns the octal value of the decimal parameter.

Internal BASIC syntax

Function Oct(dValue As Double) As String

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dValue**: Number whose octal value you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

ParseDate()

This function converts a date expressed as a character string to a Basic date object.

Internal BASIC syntax

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

Field of application

Version: 3.60

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strDate**: Date in string format.
- **strFormat**: This parameter contains the format of the date contained in the character string. The possible values are the following:
  - DD/MM/YY
  - DD/MM/YYYY
  - MM/DD/YY
  - MM/DD/YYYY
  - YYYY/MM/DD
  - Date: date expressed according to the settings of the client computer.
  - DateInter: date expressed in the international format
- **strStep**: This optional parameter contains the date separator used in the character string. The authorized separators are "\" and "-".

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

ParseDMYDate()

This function returns a Date object (as understood in Basic) from a date formatted as follows:

```
dd/mm/yyyy
```

Internal BASIC syntax

Function ParseDMYDate(strDate As String) As Date

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strDate**: Date stored as a string.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

ParseMDYDate()

This function returns a Date object (as understood in Basic) from a date formatted as follows:

```
mm/dd/yyyy
```

Internal BASIC syntax

Function ParseMDYDate(strDate As String) As Date

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strDate**: Date stored as a string.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

ParseYMDDate()

This function converts a character string representing a date formatted as yyyy/mm/dd to a Basic Date type variable

Internal BASIC syntax

Function ParseYMDDate(strDate As String) As Date

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strDate:** Date stored as a string.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

PMT()

This function returns the amount of an annuity based on constant and periodic payments, and at a constant interest rate.

Internal BASIC syntax

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

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dblRate**: This parameter indicates the interest rate per date of payment. For example, the rate per date of payment for a loan with a 6% annual interest rate, paid back by monthly dates of payment, would be:

```
0.06/12=0.005 or 0.5%
```

- **iNper**: This parameter contains the total number of dates of payment for the financial operation.

- **dblPV**: This parameter contains the actual value (or overall sum) for a series of payments to be made in the future.
- **dblFV**: This parameter contains the future value or the balance that you want to obtain after having paid the final date of payment. In general, and particularly when reimbursing a loan, this parameter is set to "0". In effect, once you have made all the dates of payment, the value of the loan is nil.
- **iType**: This parameter indicates the payment deadline. It can have one of the following values:
  - **0** if the payments are due in arrears (i.e. at the end of the period)
  - **1** if the payments are due in advance (i.e. at the start of the period)

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

Note:

- The **Rate** and **Nper** parameters must be calculated using payments expressed in the same units.
- Amounts paid (expressed in particular by the **Pmt** parameter) are represented by negative numbers. Sums received are represented by positive numbers.

PPMT()

This function returns the amount of capital reimbursed for a given date of payment in an annuity based on constant and periodic payments and at a constant interest rate.

Internal BASIC syntax

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

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dblRate**: This parameter indicates the interest rate per date of payment. For example, the rate per date of payment for a loan with a 6% annual interest rate, paid back by monthly dates of payment, would be:

```
0.06/12=0.005 or 0.5%
```

- **iPer**: This parameter indicates the period for the calculation, between 1 and the value of the **Nper** parameter.
- **iNper**: This parameter contains the total number of dates of payment for the financial operation.
- **dblPV**: This parameter contains the actual value (or overall sum) for a series of payments to be made in the future.
- **dblFV**: This parameter contains the future value or the balance that you want to obtain after having paid the final date of payment. In general, and particularly when reimbursing a loan, this parameter is set to "0". In effect, once you have made all the dates of payment, the value of the loan is nil.
- **iType**: This parameter indicates the payment deadline. It can have one of the following values:
  - **0** if the payments are due in arrears (i.e. at the end of the period)
  - **1** if the payments are due in advance (i.e. at the start of the period)

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

Note:

- The **Rate** and **Nper** parameters must be calculated using payments expressed in the same units.
- Amounts paid (expressed in particular by the **Pmt** parameter) are represented by negative numbers. Sums received are represented by positive numbers.

PV()

This function returns the actual amount of an annuity based on constant and periodic future deadlines, and on a fixed interest rate.

Internal BASIC syntax

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

Field of application

Version: 3.00

	Available
AssetCenter API	

	Available
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dblRate**: This parameter indicates the interest rate per date of payment. For example, the rate per date of payment for a loan with a 6% annual interest rate, paid back by monthly dates of payment, would be:

```
0.06/12=0.005 or 0.5%
```

- **iNper**: This parameter contains the total number of dates of payment for the financial operation.
- **dblPmt**: This parameter indicates the amount of the payment made at each date of payment. The payment generally includes both principal and interest.
- **dblFV**: This parameter contains the future value or the balance that you want to obtain after having paid the final date of payment. In general, and particularly when reimbursing a loan, this parameter is set to "0". In effect, once you have made all the dates of payment, the value of the loan is nil.
- **iType**: This parameter indicates the payment deadline. It can have one of the following values:
  - **0** if the payments are due in arrears (i.e. at the end of the period)
  - **1** if the payments are due in advance (i.e. at the start of the period)

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

Note:

- The **Rate** and **Nper** parameters must be calculated using payments expressed in the same units.
- Amounts paid (expressed in particular by the **Pmt** parameter) are represented by negative numbers. Sums received are represented by positive numbers.

Randomize()

Initializes the random number generator.

Internal BASIC syntax

Function Randomize(lValue As Long)

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **lValue**: Optional parameter used to initialize the random-number generator of the **Rnd** function by specifying a new initial value. If this parameter is omitted , the value returned by the system clock is used as the initial value.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyNumber
  Randomize
  MyNumber= Int((10*Rnd)+1) :'Returns a random value between 1 and 10.
  RetVal=MyNumber
```

RATE()

This function returns the interest rate per date of payment for an annuity.

Internal BASIC syntax

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

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **iNper**: This parameter contains the total number of dates of payment for the financial operation.
- **dblPmt**: This parameter indicates the amount of the payment to be made at each date of payment. The payment generally includes both principal and interest.
- **dblFV**: This parameter contains the future value or the balance that you want to obtain after having paid the final date of payment. In general, and particularly when reimbursing a loan, this parameter is set to "0". In effect, once you have made all the dates of payment, the value of the loan is nil.
- **dblPV**: This parameter contains the actual value (or overall sum) for a series of payments to be made in the future.
- **iType**: This parameter indicates the payment deadline. It can have one of the following values:
  - **0** if the payments are due in arrears (i.e. at the end of the period)
  - **1** if the payments are due in advance (i.e. at the start of the period)
- **dblGuess**: This parameter contains the estimated value of the interest rate per date of payment.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

Note:

- Amounts paid (expressed in particular by the **Pmt** parameter) are represented by negative numbers. Sums received are represented by positive numbers.
- This function performs its calculation using iterations, starting with the value assigned in the **Guess** parameter. If no result is found after 20 iterations, the function fails.

RemoveRows()

Performs a deletion in a list of lines identified by the **strRowNames** parameter.

This function is useful when processing "ListBox" control type values. Values from this type of control are represented as arrays as described below:

- The "|" character is used as the column separator.
- The "," character is used as the line separator.
- Each line ends with a unique identifier at the right of the "=" sign.

Internal BASIC syntax

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

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strList**: Source string containing the values of a "ListBox" control to be processed.
- **strRowNames**: Identifiers of lines to be deleted. The identifiers are separated by commas.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyStr
  MyStr=RemoveRows("a1|a2=a0,b1|b2=b0", "a0,c0") :'Returns "b1|b2=b0"
  RetVal=MyStr
```

Replace()

Replaces all occurrences of the **strOldPattern** parameter with the **strNewPattern** parameter inside the character string contained in the **strData** parameter. The search for the **strOldPattern** parameter can be made case-sensitive using the value of the **bCaseSensitive** parameter.

Internal BASIC syntax

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

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strData**: Character string containing the occurrences to be replaced.
- **strOldPattern**: Occurrence to find in the string contained in the **strData** parameter.
- **strNewPattern**: Text replacing each occurrence found.
- **bCaseSensitive**: Depending on this parameter, the search is case sensitive (=1) or not (=0).

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyStr
  MyStr=Replace("youmeyoumeyou", "you", "me",0) :'Returns "mememememe"
  MyStr=Replace("youmeyoumeyou", "You", "me",1) :'Returns "youmeyoumeyou"
  MyStr=Replace("youmeYoumeyou", "You", "me",1) :'Returns "youmememeyou"
  RetVal=""
```

Right()

Returns the rights most iNumber characters of the string parameter.

Internal BASIC syntax

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

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strString**: Character string to process.
- **iNumber**: Number of characters to return.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim lWord, strMsg, rWord, iPos :' Declare variables.
  strMsg = "Left() Test."
  iPos = InStr(1, strMsg, " ") :' Find space.
  lWord = Left(strMsg, iPos - 1) :' Get left word.
  rWord = Right(strMsg, Len(strMsg) - iPos) :' Get right word.
  strMsg=rWord+lWord :' And swap them
  RetVal=strMsg
```

RightPart()

Extracts the portion of a string to the right of the separator specified in the **strSep** parameter.

The search for the separator is performed from right to left.

The search can be made case sensitive using the **bCaseSensitive** parameter.

Internal BASIC syntax

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

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strFrom**: Source string to be processed.

- **strSep**: Character used as separator in the source string.
- **bCaseSensitive**: Depending on this parameter, the search is case sensitive (=1) or not (=0).

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

These examples illustrate use of the **LeftPart**, **LeftPartFromRight**, **RightPart**, and **RightPartFromLeft** functions on the same string: "This_is_a_test":

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

Returns "This".

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

Returns "This_is_a".

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

Returns "test".

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

Returns "is_a_test".

RightPartFromLeft()

Extracts the portion of a string to the right of the separator specified in the **strSep** parameter.

The search for the separator is performed from left to right.

The search can be made case sensitive using the **bCaseSensitive** parameter.

Internal BASIC syntax

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

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strFrom**: Source string to be processed.
- **strSep**: Character used as separator in the source string.
- **bCaseSensitive**: Depending on this parameter, the search is case sensitive (=1) or not (=0).

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

These examples illustrate use of the **LeftPart**, **LeftPartFromRight**, **RightPart**, and **RightPartFromLeft** functions on the same string: "This_is_a_test":

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

Returns "This".

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

Returns "This_is_a".

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

Returns "test".

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

Returns "is_a_test".

RmDir()

Removes an existing directory.

Internal BASIC syntax

Function RmDir(strRmDirectory As String) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strRmDirectory**: Full path of the directory to be removed.

Output parameters

- 0: Normal execution.
- Other than zero: Error code.

Rnd()

Returns a value containing a random number.

Internal BASIC syntax

Function Rnd(dValue As Double) As Double

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dValue**: Optional parameter whose value defines the mode of execution of the function:
  - Less than zero: The same number is generated each time.
  - Greater than zero: Next random number in the series.
  - Equal to zero: Last random number generated.
  - Omitted: Next random number in the series.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

Note:

Before calling this function, you must use the **Randomize** function, without parameters, to initialize the random number generator.

Example

```
Dim MyNumber
  Randomize
  MyNumber= Int((10*Rnd)+1) :'Returns a random value between 1 and 10.
  RetVal=MyNumber
```

RTrim()

Removes all trailing spaces in a string.

Internal BASIC syntax

Function RTrim(strString As String) As String

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strString**: String to process.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
' This example uses the LTrim and RTrim functions to strip leading ' and
trailing spaces, respectively, from a string variable.
' It uses the Trim function alone to strip both types of spaces.
' LCase and UCase are also shown in this example as well as the use
' of nested function calls

Dim strString as String
Dim strTrimString as String
  strString = "  <-Trim->  " :' Initialize string.
  strTrimString = LTrim(strString) :' strTrimString = "<-Trim->  ".
  strTrimString = LCase(RTrim(strString)) :' strTrimString = "  <-trim->"
.
  strTrimString = LTrim(RTrim(strString)) :' strTrimString = "<-Trim->".
  ' Using the Trim function alone achieves the same result.
  strTrimString = UCase(Trim(strString)) :' strTrimString = "<-TRIM->".
  RetVal= "|" & strTrimString & "|"
```

Second()

Returns the number of seconds contained in the time expressed by the **tmTime** parameter.

Internal BASIC syntax

Function Second(tmTime As Date) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **tmTime**: Parameter in Date+Time format to be processed.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim strSecond
  strSecond=Second(Date())
  RetVal=strSecond :'Returns the number of seconds elapsed in the current
 hour, for example "30" if the time is 15:45:30
```

SetSubList()

Defines the values of a sublist for a "ListBox" control.

Internal BASIC syntax

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

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strValues**: Source string containing the values of a "ListBox" control to be processed.
- **strRows**: List of values to add to or replace the characters contained in the string in the **strValues** parameter. The values are separated by the "|" character. The lines that are processed are identified by their identifier, situated to the right of the "=" sign. Unknown lines are not processed.

- **strRowFormat**: Formatting instructions for the sublist. Instructions are separated by the "|" character. This parameter has the following characteristics:
  - "1" represents the information contained in the first column of the sublist.
  - "i-j" can be used to define a group of columns.
  - "-" takes all columns into account.
  - An unknown column does not return a value.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyStr
  MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "A2|A1=a0, B2|B1=b
0", "2|1") :'Returns "A1|A2|a3=a0,B1|B2|b3=b0,c1|c2|c3=c0"
  MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "Z2=*,B2=b0", "2")
 :'Returns "a1|Z2|a3=a0,b1|B2|b3=b0,c1|Z2|c3=c0"
  MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "B5|B6|B7=b0,C5|C6
,C7=c0", "5-7") :'Returns "a1|a2|a3=a0,b1|b2|b3||B5|B6|B7=b0,c1|c2|c3||C5
|C6|C7=c0"
  MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "B1|B2|B3|B4=b0",
"-") :'Returns "a1|a2|a3=a0,B1|B2|B3|B4=b0,c1|c2|c3=c0"
  MyStr=SubList("A|B|C,D|E|F", "X=*", "2") :'Returns "A|X|C,D|X|F"
  RetVal=""
```

Sgn()

Returns a value indicating the sign of a number.

Internal BASIC syntax

Function Sgn(dValue As Double) As Double

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dValue**: Number whose sign you want know.

Output parameters

The function can return one of the following values:

- 1: The number is greater than zero.
- 0: The number is equal to zero
- -1: The number is less than zero.

Example

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

Shell()

Launches an executable program.

Internal BASIC syntax

Function Shell(strExec As String) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strExec**: Full path of the executable to be launched.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

Sin()

Returns the sine of an number that is expressed in radians.

Internal BASIC syntax

Function Sin(dValue As Double) As Double

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dValue**: Number whose sine you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

Space()

Creates a string including the number of spaces indicated by the **iSpace** parameter.

Internal BASIC syntax

Function Space(iSpace As Long) As String

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **iSpace**: Number of spaces to be inserted into the string.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Notes

Note:

This function can be used to format strings or to delete date in fixed length strings.

Example

```
Dim MyString
' Returns a string of 10 spaces.
  MyString = Space(10)
 ' Inserts 10 spaces between two strings.
  MyString = "Space" & Space(10) &  "inserted"
 RetVal=MyString
```

Sqr()

Returns the square root of a number.

Internal BASIC syntax

Function Sqr(dValue As Double) As Double

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dValue**: Number whose square root you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

Str()

Converts a number to a string.

Internal BASIC syntax

Function Str(strValue As String) As String

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓

	Available
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strValue**: Number to convert to a string.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

StrComp()

Compares two strings.

Internal BASIC syntax

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

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strString1**: First string.
- **strString2**: Second string.
- **iOptionCompare**: Comparison type. This parameter can be set to "0" for a binary comparison, or "1" for a text comparison.

Output parameters

- -1: **strString1** is greater than **strString2**.
- 0: **strString1** is equal to **strString2**.
- 1: **strString1** is less than **strString2**.

String()

String returns a string consisting of the **strString** character repeated over and over **iCount** times.

Internal BASIC syntax

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

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **iCount**: Number of occurrences of the character **strString**.
- **strString**: Character used to compose the string.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

SubList()

Returns a sublist of a list of values contained in a string representing the values of a "ListBox" control.

Internal BASIC syntax

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

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strValues**: Source string containing the values of a "ListBox" control to be processed.
- **strRows**: Identifiers of lines to be included in the sublist. The identifiers are separated by commas. Certain wildcard characters can be used:
  - "*" includes all identifiers in the sublist.
  - An unknown identifier returns an empty value for the sublist.
- **strRowFormat**: Formatting instructions for the sublist. Instructions are separated by the "|" character. This parameter has the following characteristics:
  - "1" represents the information contained in the first column of the list from which we are extracting a sublist.
  - "0" represents the identifier of the line in the list from which we are extracting a sublist.
  - "*" represents the information contained in all the columns (except the line identifier).
  - An unknown column does not return a value.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyStr
  MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "a0,b0,a0", "3|2|3
") :'Returns "a3|a2|a3,b3|b2|b3,a3|a2|a3"
  MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "*", "*|0") :'Retu
rns "a1|a2|a3|a0,b1|b2|b3|b0,c1|c2|c3|c0"
  MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "*", "*=0") :'Retu
rns "a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0"
  MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "*", "999=0") :'Re
turns "=a0,=b0,=c0"
  MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "z0", "*=0") :'Ret
urns ""
  MyStr=SubList("a1|a2|a3=a0,b1|b2|b3=b0,c1|c2|c3=c0", "*", "=1") :'Retur
ns "=a1,=b1,=c1"
  MyStr=SubList("A|B|C,D|E|F", "*", "2=0") :'Returns "B,E"
  RetVal=""
```

Tan()

Returns the tangent of a number expressed in radians.

Internal BASIC syntax

Function Tan(dValue As Double) As Double

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **dValue**: Number whose tangent you want to know.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

Time()

Returns the current time.

Internal BASIC syntax

Function Time() As Date

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Timer()

Returns the number of seconds elapsed since 12:00 AM.

Internal BASIC syntax

Function Timer() As Double

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓

	Available
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

TimeSerial()

This function returns a time formatted according to the **iHour**, **iMinute** and **iSecond parameters**.

Internal BASIC syntax

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

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **iHour**: Hour.
- **iMinute**: Minutes.
- **iSecond**: Seconds.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

Each of these parameters can be set to a numeric expression representing a number of hours, minutes or seconds. Thus in the following example:

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

Returns the value:

```
3:50:00
```

When the value of a parameter is out of the expected range (i.e. 0-59 for minutes and seconds and 0-23 for hours), it is converted to the parameter the next up. Thus, if you enter "75" for the **iMinute** parameter, it will be interpreted as 1 hour and 15 minutes.

The following example:

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

Returns the value:

```
16:50:45
```

TimeValue()

This function returns the time portion of a "Date+Time" value.

Internal BASIC syntax

Function TimeValue(tmTime As Date) As Date

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **tmTime**: "Date+Time" format date.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

The following example:

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

Returns the value:

```
15:00:00
```

ToSmart()

This function reformats a source string by capitalizing the first letter of each word.

Internal BASIC syntax

Function ToSmart(strString As String) As String

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strString**: Source string to reformat.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Trim()

Returns a copy a string with the leading and trailing spaces removed.

Internal BASIC syntax

Function Trim(strString As String) As String

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strString**: String to process.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
' This example uses the LTrim and RTrim functions to strip leading ' and
trailing spaces, respectively, from a string variable.
' It uses the Trim function alone to strip both types of spaces.
' LCase and UCase are also shown in this example as well as the use
' of nested function calls

Dim strString as String
Dim strTrimString as String
  strString = "  <-Trim->  " :' Initialize string.
  strTrimString = LTrim(strString) :' strTrimString = "<-Trim->  ".
  strTrimString = LCase(RTrim(strString)) :' strTrimString = "  <-trim->"
.
  strTrimString = LTrim(RTrim(strString)) :' strTrimString = "<-Trim->".
  ' Using the Trim function alone achieves the same result.
  strTrimString = UCase(Trim(strString)) :' strTrimString = "<-TRIM->".
  RetVal= "|" & strTrimString & "|"
```

UCase()

Returns a copy of a sting in which all lowercase characters are converted to uppercase.

Internal BASIC syntax

Function UCase(strString As String) As String

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strString:** Character string to convert to uppercase.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
' This example uses the LTrim and RTrim functions to strip leading ' and
trailing spaces, respectively, from a string variable.
' It uses the Trim function alone to strip both types of spaces.
' LCase and UCase are also shown in this example as well as the use
' of nested function calls

Dim strString as String
Dim strTrimString as String
  strString = "  <-Trim->  " :' Initialize string.
  strTrimString = LTrim(strString) :' strTrimString = "<-Trim->  ".
  strTrimString = LCase(RTrim(strString)) :' strTrimString = "  <-trim->"
.
  strTrimString = LTrim(RTrim(strString)) :' strTrimString = "<-Trim->".
  ' Using the Trim function alone achieves the same result.
  strTrimString = UCase(Trim(strString)) :' strTrimString = "<-TRIM->".
  RetVal= "|" & strTrimString & "|"
```

UnEscapeSeparators()

Deletes all the escape characters from a string.

Internal BASIC syntax

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

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strSource**: Character string to process.
- **strEscChar**: Escape character to be deleted.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyStr
  MyStr=UnEscapeSeparators("you\|me\|you\|", "\") :'Returns "you|me|you|"
 RetVal=""
```

Union()

Merges two strings delimited by separators. Duplicates are deleted.

Internal BASIC syntax

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

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strListOne**: First string.
- **strListTwo**: Second string.
- **strSeparator**: Separator used to delimit the elements contained in the strings.
- **strEscChar**: Escape character. If this character prefixes the separator, it will be ignored.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

```
Dim MyStr
  MyStr=Union("a1|a2,b1|b2", "a1|a3,b1|b2", ",", "\") :'Returns "a1|a2,b1
|b2,a1|a3"
  MyStr=Union("a1|a2,b1|b2", "a1|a3\,b1|b2", ",", "\") :'Returns "a1|a2,b
1|b2,a1|a3\,b1|b2"
  RetVal=""
```

UTCToLocalDate()

This function converts a date in UTC format (time-zone independent) to a "Date+Time" format date.

Internal BASIC syntax

Function UTCToLocalDate(tmUTC As Date) As Date

Field of application

Version: 3.5

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **tmUTC**: Date in UTC format.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Val()

Converts a string representing a number to a double.

Internal BASIC syntax

Function Val(strString As String) As Double

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **strString**: Character string to convert.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.

- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

Example

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

WeekDay()

Returns the day of the week contained in the date expressed by the **tmDate** parameter.

Internal BASIC syntax

Function WeekDay(tmDate As Date) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **tmDate**: Parameter in Date+Time format to be processed.

Output parameters

The number returned corresponds to a day of the week where "1" represents Sunday, "2" Tuesday, ..., "7" Saturday.

Example

```
Dim strWeekDay
  strWeekDay=WeekDay(Date())
  RetVal=strWeekDay :'Returns the day of the week
```

Year()

Returns the year contained in the value expressed by the **tmDate** parameter.

Internal BASIC syntax

Function Year(tmDate As Date) As Long

Field of application

Version: 3.00

	Available
AssetCenter API	
Configuration script of a field or link	✓
"Script" type action	✓
Wizard script	✓
FINISH.DO script of a wizard	✓

Input parameters

- **tmDate**: Parameter in Date+Time format to be processed.

Output parameters

In case of error, there are two possibilities:

- In AssetCenter, the script containing the function is suspended and an error message issued to the user.
- If calling from an external program, you must call the AmLastError() function (and optionally the AmLastErrorMsg() function) to find out if an error occurred (and obtain its associated message).

IV Index

PART

Available functions - Domain: All

INDEX

- Abs
- AmActionDde
- AmActionExec
- AmActionMail
- AmActionPrint
- AmActionPrintPreview
- AmActionPrintTo
- AmAddAllPOLinesToInv
- AmAddCatRefAndCompositionToPOrder
- AmAddCatRefToPOrder
- AmAddEstimLinesToPO
- AmAddEstimLineToPO
- AmAddPOLineToInv
- AmAddPOrderLineToReceipt
- AmAddReceiptLineToInvoice
- AmAddReqLinesToEstim
- AmAddReqLinesToPO
- AmAddReqLineToEstim
- AmAddReqLineToPO
- AmAddRequestLineToPOrder
- AmAddTemplateToPOrder
- AmAddTemplateToRequest
- AmBusinessSecondsInDay
- AmCalcConsolidatedFeature
- AmCalcDepr
- AmCbkReplayEvent
- AmCheckTraceDone
- AmCleanup
- AmClearLastError
- AmCloseAllChildren
- AmCloseConnection
- AmCommit
- AmComputeAllLicAndInstallCounts
- AmComputeLicAndInstallCounts
- 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
- AmCreatePOFromEstim
- AmCreatePOFromReq
- AmCreatePOrderFromRequest
- AmCreatePOrdersFromRequest
- AmCreatePOsFromAllReqLines
- AmCreateProjectCable
- AmCreateProjectDevice
- AmCreateProjectTrace
- AmCreateReceiptFromPOrder
- AmCreateRecord
- AmCreateRequestToInvoice
- AmCreateRequestToPOrder
- AmCreateRequestToReceipt
- AmCreateReturnFromReceipt
- AmCreateTraceHist
- AmCryptPassword
- AmCurrentDate
- AmCurrentIsoLang
- AmCurrentLanguage
- AmCurrentServerDate
- AmDateAdd
- AmDateAddLogical
- AmDateDiff
- AmDbGetDate
- AmDbGetDouble
- AmDbGetList
- AmDbGetListEx
- AmDbGetLong
- AmDbGetPk
- AmDbGetString
- AmDbGetStringEx
- AmDeadLine
- AmDecrementLogLevel
- AmDefAssignee
- AmDefaultCurrency
- AmDefEscalationScheme
- AmDefGroup
- AmDeleteLink
- AmDeleteRecord
- AmDisconnectTrace
- AmDuplicateRecord
- AmEndOfNthBusinessDay
- AmEnumValList
- AmEvalScript
- AmExecTransition
- AmExecuteActionById
- AmExecuteActionByName
- AmExportDocument
- AmFindCable
- AmFindDevice
- AmFindRootLink
- AmFindTermDevice
- AmFindTermField
- AmGenSqlName
- AmGetCatRef

- AmGetCatRefFromCatProduct
- AmGetComputeString
- AmGetCurrentNTDomain
- AmGetCurrentNTUser
- AmGetFeat
- AmGetFeatCount
- AmGetField
- AmGetFieldCount
- 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
- AmGetPOLinePrice
- AmGetPOLinePriceCur
- AmGetPOLineReference
- AmGetRecordFromMainId
- AmGetRecordHandle
- AmGetRecordId
- AmGetRelDstField
- AmGetRelSrcField
- AmGetRelTable
- AmGetReverseLink
- AmGetSelfFromMainId
- AmGetSourceTable
- AmGetTable
- AmGetTableCount
- AmGetTableDescription
- AmGetTableFromName
- AmGetTableLabel
- AmGetTableName
- AmGetTableRights
- AmGetTableSqlName
- AmGetTargetTable

- AmGetTrace
- AmGetTraceFromHist
- AmGetTypedLinkField
- AmGetVersion
- AmHasAdminPrivilege
- AmHasRelTable
- AmImportDocument
- AmIncrementLogLevel
- AmInsertRecord
- AmInstantiateReqLine
- AmInstantiateRequest
- AmIsConnected
- AmIsFieldForeignKey
- AmIsFieldIndexed
- AmIsFieldPrimaryKey
- AmIsLink
- AmIsTypedLink
- AmLastError
- AmLastErrorMsg
- AmListToString
- AmLog
- AmLoginId
- AmLoginName
- AmMapSubReqLineAgent
- AmMoveCable
- AmMoveDevice
- AmMsgBox
- AmOpenConnection
- AmOpenScreen
- AmPagePath
- AmProgress
- AmQueryCreate
- AmQueryExec
- AmQueryGet
- AmQueryNext
- AmQuerySetAddMainField
- AmQuerySetFullMemo
- AmQueryStartTable
- AmQueryStop
- AmReceiveAllPOLines
- AmReceivePOLine
- AmRefreshAllCaches
- AmRefreshLabel
- AmRefreshProperty
- AmRefreshTraceHist
- AmReleaseHandle
- AmRemoveCable
- AmRemoveDevice
- AmReturnAsset
- AmReturnContract
- AmReturnPortfolioItem
- AmReturnTraining
- AmReturnWorkOrder
- AmRevCryptPassword
- AmRgbColor
- AmRollback
- AmSetFieldDateValue
- AmSetFieldDoubleValue
- AmSetFieldLongValue
- AmSetFieldStrValue
- AmSetLinkFeatureValue
- AmSetProperty
- AmShowCableCrossConnect
- AmShowDeviceCrossConnect
- AmSqlTextConst
- AmStartTransaction
- AmStartup
- AmTableDesc

- AmTaxRate
- AmUpdateDetail
- AmUpdateRecord
- AmValueOf
- AmWizChain
- AmWorkTimeSpanBetween
- AppendOperand
- ApplyNewVals
- Asc
- Atn
- BasicToLocalDate
- BasicToLocalTime
- BasicToLocalTimeStamp
- Beep
- CDbl
- ChDir
- ChDrive
- Chr
- CInt
- CLng
- Cos
- CountOccurences
- CountValues
- CSng
- CStr
- CurDir
- CVar
- DaContext
- DaCopy
- DaDbDeleteList
- DaDbGetList
- DaDbSetList
- DaDelete
- DaDownload
- DaDumpContext
- DaExec
- DaExecAction
- DaExecuteActionByName
- DaFileATime
- DaFileCRC
- DaFileCTime
- DaFileLanguage
- DaFileMTime
- DaFileSize
- DaFileType
- DaFileVersion
- DaFind
- DaFindNext
- DaFirstEnv
- DaGetEnv
- DaGetFileInfo
- DaImpersonate
- DaMkDir
- DaMove
- DaNetIpFromName
- DaNetNBTName
- DaNetPing
- DaNetWakeOnLan
- DaNetWinAddressByName
- DaNextEnv
- DaNTFileCopyTo
- DaNTFileCreateDir
- DaNTFileDelete
- DaNTFileDeleteDir
- DaNTFileDirCopyTo
- DaNTFileDirDownload
- DaNTFileDirUpload
- DaNTFileDownload

- DaNTFileUpload
- DaNTRegistryLMAddStringValue
- DaNTRegistryLMCreateKey
- DaNTRegistryLMDeleteKey
- DaNTRegistryLMDeleteValue
- DaNTRegistryLMGetLongValue
- DaNTRegistryLMGetStringValue
- DaNTRegistryLMSetLongValue
- DaNTRegistryLMSetStringValue
- DaNTServiceInstall
- DaNTServiceStart
- DaNTServiceStatus
- DaNTServiceStop
- DaNTServiceUninstall
- DaNTWMIExecMethod
- DaNTWMIExecQuery
- DaNTWMIGetCurrentArrayValue
- DaNTWMIGetCurrentPropertyValue
- DaNTWMIGetInstanceCount
- DaNTWMIGetPropertyValue
- DaNTWMIGetTotalPropertiesValue
- DaNTWMINextItem
- DaNTWMIResetEnumeration
- DaRegCreateKey
- DaRegDeleteKey
- DaRegExec
- DaRegGetValue
- DaRegOutputValue
- DaRegSetValue
- DaRegStrValue
- DaRegVarValue
- DaRename
- DaReturnValue
- DaRmDir
- DaSendMail
- DaSetContext
- DaSetOption
- DaSetReturnValue
- Date
- DateSerial
- DateValue
- DaTrackingDelete
- DaTrackingGet
- DaTrackingSet
- DaTrackingTest
- DaUpload
- DaWait
- Day
- EscapeSeparators
- ExeDir
- Exp
- ExtractValue
- FileCopy
- FileDateTime
- FileExists
- FileLen
- Fix
- FormatDate
- FormatResString
- FV
- GetListItem
- Hex
- Hour
- InStr
- Int
- IPMT
- IsNumeric
- Kill

- LCase
- Left
- LeftPart
- LeftPartFromRight
- Len
- LocalToBasicDate
- LocalToBasicTime
- LocalToBasicTimeStamp
- LocalToUTCDate
- Log
- LTrim
- MakeInvertBool
- Mid
- Minute
- MkDir
- Month
- Name
- Now
- NPER
- Oct
- ParseDate
- ParseDMYDate
- ParseMDYDate
- ParseYMDDate
- PMT
- PPMT
- PV
- Randomize
- RATE
- RemoveRows
- Replace
- Right
- RightPart
- RightPartFromLeft
- RmDir
- Rnd
- RTrim
- Second
- SetSubList
- Sgn
- Shell
- Sin
- Space
- Sqr
- Str
- StrComp
- String
- SubList
- Tan
- Time
- Timer
- TimeSerial
- TimeValue
- ToSmart
- Trim
- UCase
- UnEscapeSeparators
- Union
- UTCToLocalDate
- Val
- WeekDay
- Year

Available functions - Domain: Procurement

INDEX

- AmAddAllPOLinesToInv
- AmAddCatRefAndCompositionToPOrder
- AmAddCatRefToPOrder
- AmAddEstimLinesToPO
- AmAddEstimLineToPO
- AmAddPOLineToInv
- AmAddPOrderLineToReceipt
- AmAddReceiptLineToInvoice
- AmAddReqLinesToEstim
- AmAddReqLinesToPO
- AmAddReqLineToEstim
- AmAddReqLineToPO
- AmAddRequestLineToPOrder
- AmAddTemplateToPOrder
- AmAddTemplateToRequest
- AmCreateAssetsAwaitingDelivery
- AmCreateDelivFromPO
- AmCreateEstimFromReq
- AmCreateEstimsFromAllReqLines
- AmCreateInvFromPO
- 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

Available functions - Domain: Cable

INDEX

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

Available functions - Domain: Actions

INDEX

- AmActionDde
- AmActionExec
- AmActionMail
- AmActionPrint
- AmActionPrintPreview
- AmActionPrintTo
- AmExecuteActionById
- AmExecuteActionByName

Available functions - Domain: Builtin

INDEX

- Abs
- AppendOperand
- ApplyNewVals
- Asc
- Atn
- BasicToLocalDate
- BasicToLocalTime
- BasicToLocalTimeStamp
- Beep
- CDbl
- ChDir
- ChDrive
- Chr
- CInt
- CLng
- Cos
- CountOccurences
- CountValues
- CSng
- CStr
- CurDir
- CVar
- Date
- DateSerial
- DateValue
- Day
- EscapeSeparators
- ExeDir
- Exp
- ExtractValue
- FileCopy
- FileDateTime
- FileExists
- FileLen
- Fix
- FormatDate
- FormatResString
- FV
- GetListItem
- Hex
- Hour
- InStr
- Int
- IPMT
- IsNumeric
- Kill

- LCase
- Left
- LeftPart
- LeftPartFromRight
- Len
- LocalToBasicDate
- LocalToBasicTime
- LocalToBasicTimeStamp
- LocalToUTCDate
- Log
- LTrim
- MakeInvertBool
- Mid
- Minute
- MkDir
- Month
- Name
- Now
- NPER
- Oct
- ParseDate
- ParseDMYDate
- ParseMDYDate
- ParseYMDDate
- PMT
- PPMT
- PV
- Randomize
- RATE
- RemoveRows
- Replace
- Right
- RightPart
- RightPartFromLeft
- RmDir
- Rnd
- RTrim
- Second
- SetSubList
- Sgn
- Shell
- Sin
- Space
- Sqr
- Str
- StrComp
- String
- SubList
- Tan
- Time
- Timer
- TimeSerial
- TimeValue
- ToSmart
- Trim
- UCase
- UnEscapeSeparators
- Union
- UTCToLocalDate
- Val
- WeekDay
- Year

Available functions - Domain: Wizards

INDEX

- AmDecrementLogLevel
- AmExecTransition
- AmIncrementLogLevel
- AmLog
- AmPagePath
- AmProgress
- AmRefreshProperty
- AmSetProperty
- AmUpdateDetail
- AmValueOf
- AmWizChain

Available functions - Domain: Authentication

INDEX

- DaImpersonate

Available functions - Domain: Remote execution

INDEX

- DaExec

Available functions - Domain: File management

INDEX

- DaCopy
- DaDelete
- DaDownload
- DaFileATime
- DaFileCRC
- DaFileCTime
- DaFileLanguage
- DaFileMTime
- DaFileSize
- DaFileType
- DaFileVersion
- DaFind
- DaFindNext
- DaGetFileInfo
- DaMkDir
- DaMove
- DaRename
- DaRmDir
- DaUpload

Available functions - Domain: Context

INDEX

- DaContext
- DaDumpContext
- DaSetContext

Available functions - Domain: Registry

INDEX

- DaRegCreateKey
- DaRegDeleteKey
- DaRegExec
- DaRegGetValue
- DaRegOutputValue
- DaRegSetValue
- DaRegStrValue
- DaRegVarValue

Available functions - Domain: Database

INDEX

- DaDbDeleteList
- DaDbGetList
- DaDbSetList
- DaExecAction
- DaExecuteActionByName

Available functions - Domain: Tracking

INDEX

- DaTrackingDelete
- DaTrackingGet
- DaTrackingSet
- DaTrackingTest

Available functions - Domain: WMI

INDEX

- DaNTWMIExecMethod
- DaNTWMIExecQuery
- DaNTWMIGetCurrentArrayValue
- DaNTWMIGetCurrentPropertyValue
- DaNTWMIGetInstanceCount
- DaNTWMIGetPropertyValue
- DaNTWMIGetTotalPropertiesValue
- DaNTWMINextItem
- DaNTWMIResetEnumeration

Available functions - Domain: Network

INDEX

- DaNetIpFromName
- DaNetNBTName
- DaNetPing
- DaNetWakeOnLan
- DaNetWinAddressByName

Peregrine SYSTEMS®

October 10, 2002